ON ADAPTIVE FILTER STRUCTURE AND PERFORMANCE

Bernard Mulgrew B.Sc. A.M.I.E.E.

Thesis submitted for the degree

of Doctor of Philosophy to the

Faculty of Science,

University of Edinburgh.

1987

UNIVERSITY OF EDINBURGH

ABSTRACT OF THESIS (Regulation 7.9)

Name of Candidate Bernard Mulgrew

Address [redacted]

Degree Ph.D. *Date* Febuary 1987

Title of Thesis On Adaptive Filter Structure and Performance

No. of words in the main text of Thesis 45,000

The subject of this thesis is the design, performance and structure of algorithms for discrete time adaptive filtering. Both finite impulse response (FIR) and infinite impulse response (IIR) filters are considered. However, while it has been possible to look at FIR filters in a general, application - independent, manner, IIR filters have only been examined in the specific application of linear equalisation of digital communications channels.

A broad selection of adaptive FIR filter algorithms are examined to assess relative convergence performance (as indicated by currently available theoretical results) and computational requirements. From this examination a classification system evolves in which the available algorithms are grouped into three classes according to performance and complexity. Of particular note is the unified approach to block least mean squares (BLMS) adaptive filtering which simplifies the application of efficient convolution algorithms other than the fast Fourier transform (FFT) to the construction of computationally efficient adaptive filters.

The classification system is confirmed through the use of computer simulation. The convergence performance of the various algorithms is compared in the specific application areas of system identification and channel equalisation. It is believed that such a comparison has not previously been attempted even in the recent textbooks on the subject.

A new adaptive FIR filter algorithm is presented. Analytic and experimental results confirm that this so-called self orthogonalised block adaptive filter (SOBAF) provides a unique combination of robust convergence performance and computational efficiency.

A closed form expression for the optimum IIR equalising filter is derived using Wiener filtering theory. The closed form solution highlights the structure of the optimum IIR equaliser and the difficulties incurred in developing an adaptive IIR equaliser. A comparison of the mean-square error (MSE) performance of FIR and IIR equalisers illustrates the inherent order advantage in using an IIR filter in this application.

The minimum phase spectral factorisation, which is an integral part of the Wiener formulation of the IIR filter is circumvented through the use of the Kalman equaliser of Lawrence and Kaufman. The Kalman equaliser is then made adaptive by combining it with a least mean squares (LMS) system identification algorithm and a novel technique, which both estimates the channel noise variance and compensates the Kalman filter for uncertainty in the channel impulse response. Comparisons of the computational load and convergence performance of this adaptive Kalman equaliser with a conventional linear equaliser are provided. Further a method for improving the convergence performance of the adaptive Kalman equaliser is described which involves replacing the LMS system identification algorithm with a recursive least squares (RLS) counterpart.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Colin F.N. Cowan, for inspiration and technical guidance, my second supervisor, Dr. Peter M. Grant, for his enthusiasim, and Dr. Ganapati Panda for many noisy but fruitful technical discussions. Finally thanks to my wife, Fiona, for her encouragement and for tolerating my long imprisonment in the Department of Electrical Engineering while I completed this tome.

To my parents, Elizabeth and Edward, and to Helen Madden.

For encouraging my education.

CONTENTS

Abstract ii

Declaration of Originality iii

Acknowledgements iv

Dedication v

Contents vi

List of Abreviations ix

List of Principal Symbols xi

Chapter 1 INTRODUCTION

1.1 Adaptive Filters 1

1.2 Application and Modes of Operation 4

1.3 Thesis Layout 10

Chapter 2 ADAPTIVE FIR FILTER ALGORITHMS

2.1 Introduction 13

2.2 Optimum Linear Estimation 14

2.2.1 The Optimum FIR Filter 17

2.2.2 FIR System Identification 21

2.3 Sampled Matrix Inversion 24

2.4 Least Squares Estimation 27

2.4.1 Recursive Least Squares 31

2.4.2 Data Windows 33

2.4.3 Fast Algorithms 34

2.4.4 Properties of the Least Squares Estimate 38

2.5 Stochastic Gradient Algorithms 40

2.5.1 The Least Mean Squares Algorithm 41
2.5.2 The Block Least Mean Squares Algorithm 45
2.6 Transform Domain Algorithms 51
2.6.1 The Sliding DFT Adaptive Filter 54
2.7 Algorithm Summary and Complexity Comparison 56

Chapter 3 PERFORMANCE COMPARISONS

3.1 Introduction 59
3.2 System Identification 60
3.3 Channel Equalisation 72
3.4 Summary and Conclusions 79

Chapter 4 A SELF ORTHOGONALISED BLOCK ADAPTIVE FILTER

4.1 Introduction 80
4.2 Theory 83
4.2.1 Comparison of Theory with Simulation 86
4.3 A Practical Algorithm 89
4.4 Computational Complexity 94
4.5 Simulation Results 99
4.6 Conclusions 102

Chapter 5 THE IIR LINEAR EQUALISER

5.1 Introduction 103
5.2 The Linear Equaliser 104
5.2.1 Structure of an IIR Equaliser 110
5.3 Comparison of FIR and IIR Equaliser Performance 114
5.4 System Identification 120

5.4.1 Adaptive IIR Solutions 121

5.5 Conclusiona 124

Chapter 6 AN ADAPTIVE IIR EQUALISER

6.1 Introduction 125

6.2 The Kalman Filter 127

6.3 The Kalman Filter as an IIR Equaliser 129

6.4 An Adaptive Kalman Equaliser 133

6.4.1 System Identification 135

6.4.2 Model Uncertainty 137

6.4.3 Verification of Compensation Technique 142

6.4.4 Comparison with an RLS FIR Equaliser 144

6.4.5 Computational Complexity 152

6.5 RLS System Identification 157

6.6 Conclusions 163

Chapter 7 CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

7.1 General Remarks 164

7.2 Specific Achievements 164

7.3 Limitations and Further Work 167

Appendix A THE FAST KALMAN ALGORITHM 168

Appendix B CIRCULAR AND LINEAR CONVOLUTION 175

Appendix C RELEVENT PUBLICATIONS 179

REFERENCES 180

LIST OF ABREVIATIONS

AR	Autoregressive
ARMA	Autoregressive Moving Average
ARMAX	Autoregressive Moving Average eXogeneous
BLMS	Block Least Mean Squares
BLUE	Best Linear Unbiased Estimate
CORDIC	COordinate Rotation Digital Computer
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
EKF	Extended Kalman Filter
FAEST	Fast A posteriori Error Sequential Technique
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FTF	Fast Transversal Filter
HF	High Frequency
IIR	Infinite Impulse Response
KLT	Karhunen Loeve Transform
LMS	Least Mean Squares
LS	Least Squares
LSI	Large Scale Integration
MA	Moving Average
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
MVUE	Minimum Variance Unbiased Estimate
RIV	Recursive Instrumental Variable
RLS	Recursive Least Squares
RT	Rectangular Transform
SM	Sampled Matrix

SMI	Sampled Matrix Inversion
SOBAF	Self Orthogonalising Block Adaptive Filter
WT	Walsh Transform

LIST OF PRINCIPAL SYMBOLS

Variables and Constants

d	estimation lag of IIR equaliser
$\{ e(n) \}$	error sequence $e(0), e(1), \cdots e(n), \cdots$
F_N	$(N \times N)$ complex DFT matrix
$G(z)$	z transform of Wiener optimum IIR equaliser
$\{ h_n \}$	impulse response sequence $h_0, h_1, \cdots h_n, \cdots$
$\underline{h}(k)$	impulse response vector of a FIR filter at time k
$\underline{h}_{opt}$	impulse response vector of Wiener optimum FIR filter
I_N	$(N \times N)$ identity matrix
$\underline{K}(k)$	Kalman gain matrix at time k (chapter 6)
N	number of taps in transversal filter
O_N	$(N \times N)$ zero matrix
$\underline{r}_{xx}(k)$	least squares "autocorrelation" matrix at time k
$\underline{r}_{xy}(k)$	least squares " cross correlation" vector at time k
T_N	$(N \times N)$ time reversal matrix
$\{ x(n) \}$	input data sequence $x(0), x(1), \cdots x(n), \cdots$
$\underline{x}(k)$	data vector at time k
$\{ y(n) \}$	output data sequence $y(0), y(1), \cdots y(n), \cdots$
λ	memory factor in exponentially windowed RLS algorithm
λ_i	eigenvalue of autocorrelation matrix
μ	step size used in LMS algorithm

μ_b	step size used in BLMS algorithm
ξ	mean square error cost function
ξ_{opt}	minimum mean square error
ρ	norm, measure of performance of system identification algorithm
Φ_{xx}	autocorrelation matrix
Φ_{xy}	cross correlation vector
∇	gradient of mean square error cost function
$\hat{\nabla}(k)$	estimate of the gradient at time k

Operators

$E[\ .\ ]$	statistical expectation operator
$tr[\ .\ ]$	trace of a matrix
z^{-1}	unit sample delay
Σ	summation
Π	product
$\hat{}$	denotes an estimate

Vectors and Matrices

All vectors are specified as column vectors. The matrix transpose operation is denoted by the superscript T.

Chapter 1

INTRODUCTION

1.1 ADAPTIVE FILTERS

This thesis is primarily concerned with the design of algorithms for adaptive filtering. The key words which require further explanation are: filter, adaptive and algorithm. By filter is meant a linear discrete time filter which operates on an input sequence of data samples $\{ x(n) \}$ to produce an output sequence of data samples $\{ y(n) \}$. The filter, illustrated in Figure 1.1, is characterised by the impulse response sequence $\{ h_n \}$ [1]. Such filters find application in many situations where it is necessary to reconstruct a signal which has been corrupted by additive noise and possibly linear distortion. Design rules for the calculation of the impulse response sequence may be obtained from the work of Wiener [2] or Kalman [3]. The former is optimal in a minimum mean-square error (MMSE) [2] sense and may be applied in stationary or non time varying environments. The latter is optimal in a minimum variance sense and may also be applied to non stationary or time varying environments [4]. Both techniques require explicit a priori knowledge of the environment either in the form of auto- and cross- spectral densities for the Wiener filter or a state space model for the Kalman filter [5]. When the environment is unknown or poorly defined these optimal filters cannot be designed and an adaptive filter must be considered.

An adaptive filter differs from an non adaptive filter in that the a priori information required to design an optimal non adaptive filter is replaced by a second input sequence, known as a training or desired input (Figure 1.2). The training signal is in some sense close to or approximates the output of an optimal filter. Such an input is more readily available in a practical situation than specific knowledge of the environment in the form of spectral densities and/or a state space model. The impulse response of the adaptive filter is then altered as more of the input and training

Figure 1.1 A LINEAR DISCRETE TIME FILTER

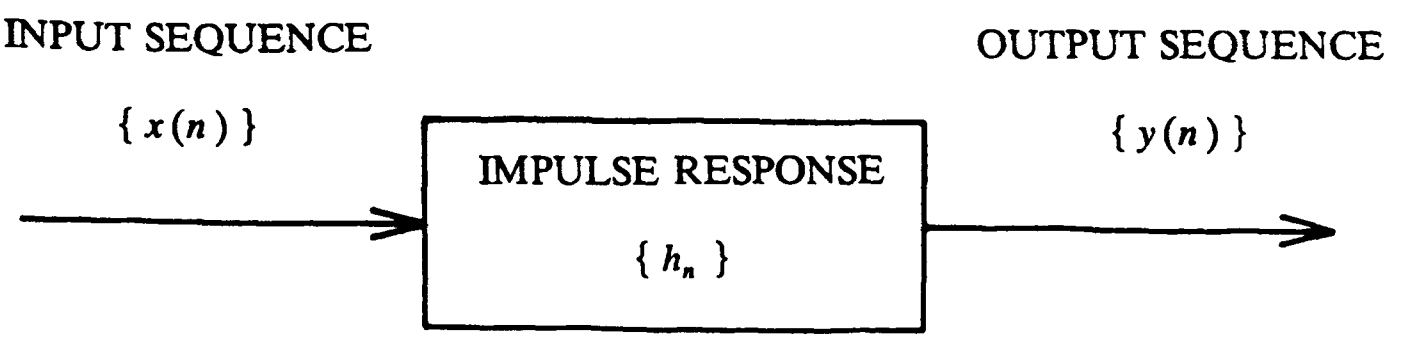
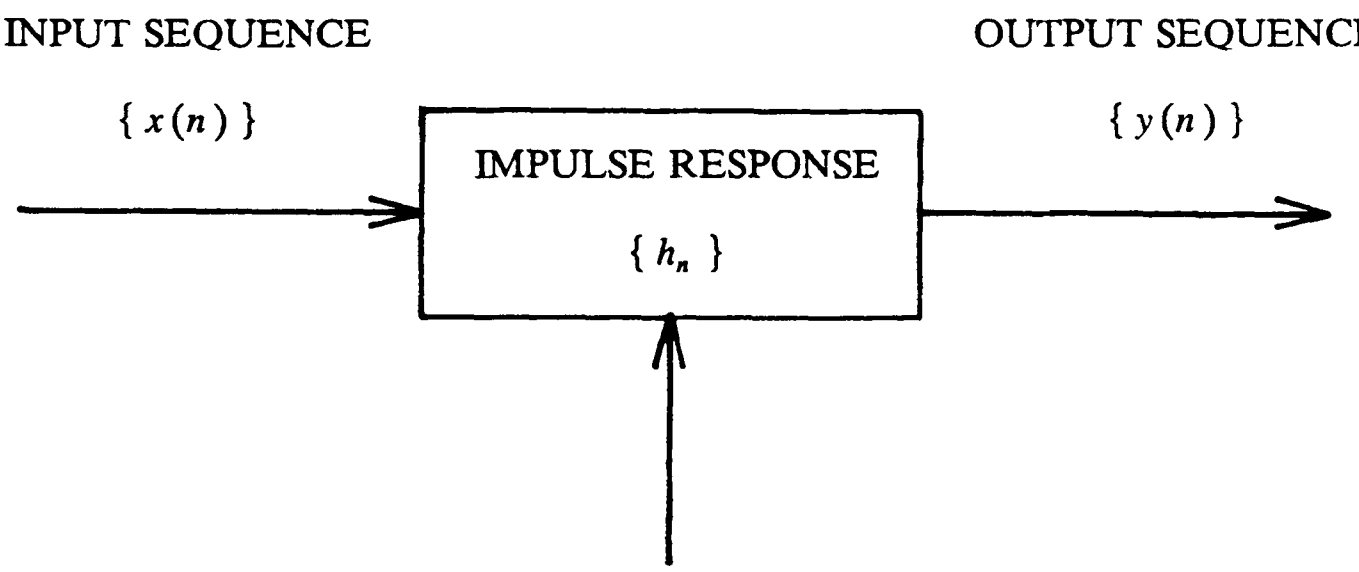


Figure 1.2 AN ADAPTIVE FILTER



sequence become available so that the output y gets closer in mean-square error (MSE) sense to the training sequence and hence to the output of the optimal filter. The strategy by which the impulse response of the adaptive filter is altered is the adaptive filter algorithm.

An adaptive filter is thus a time varying filter whose impulse response at a particular time is dependent on the input sequence, the training sequence and the adaptive filter algorithm. The time varying nature of an adaptive filter gives rise to the concept of convergence. In a stationary environment, the convergence performance is a measure of how many data samples are required for the impulse response of the adaptive filter to come within a specified distance from the impulse response of the Wiener filter. In a non-stationary environment the convergence performance is also a measure of how closely the impulse response of the adaptive filter follows the the time varying impulse response of some optimal filter, which can be identified as the Kalman filter if the underlying process is Markov [5].

1.2 APPLICATION AND MODES OF OPERATION

One of the major modes of operation of an adaptive filter is in system identification (Figure 1.3). Given an input sequence $\{ x(n) \}$, and an output sequence $\{ y(n) \}$ associated with an unknown system, the function of the adaptive filter is to estimate the impulse response sequence $\{ h_n \}$ that relates the output sequence to the input sequence. The function is effected in the adaptive filter by processing the data pairs, $(x(n), y(n))$, serially one pair at a time (or in blocks of several consequetative pairs at a time). As each new data pair (or block of data pairs) becomes available, the impulse response of the adaptive filter is updated so as to reduce the size of the error, e(n), which is the difference between the system output, y(n), and the output of the adaptive filter, $\hat{y}(n)$. In this mode, the optimal filter to which the adaptive filter aspires is the unknown system itself.

A practical example of this mode of operation is echo cancellation across the hybrid transformer used in telephone networks [6]. The hybrid transformer of Figure 1.4 performs the conversion from the two wire section, where transmission of information occurs in both directions on a single pair of wires, to the four wire section, where transmission only occurs in one direction on a pair of wires. Talker echo is the leakage of the signal from the transmitter across the hybrid into the receiver. One method of reducing the talker echo is to construct a filter in parallel with the hybrid which models the echo path across the hybrid. The echo can then be cancelled by subtracting the output of the filter, $\hat{y}(n)$, from the output of the hybrid, $y(n)$. The error sequence, $\{ e(n) \}$, is then used as the input to the receiver. Because the impulse response of echo path across the hybrid is unknown a priori and time varying, an adaptive filter is usually employed.

A second major mode of operation of an adaptive filter is in inverse system modelling or deconvolution, where a sequence, $\{ y'(n) \}$, is subjected to linear distortion (filtering) and additive noise in an unknown system to produce a second

Figure 1.3 SYSTEM IDENTIFICATION

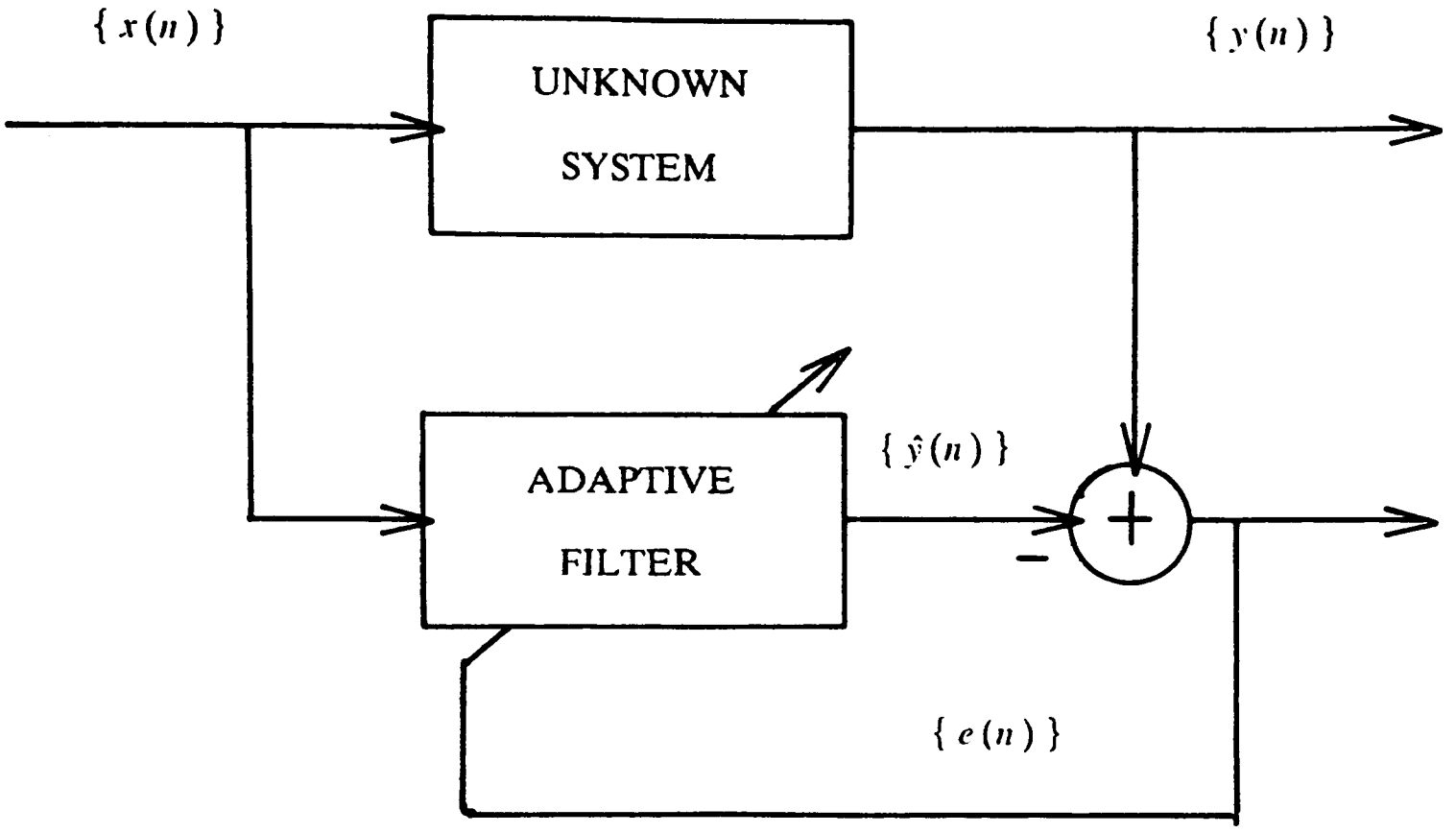
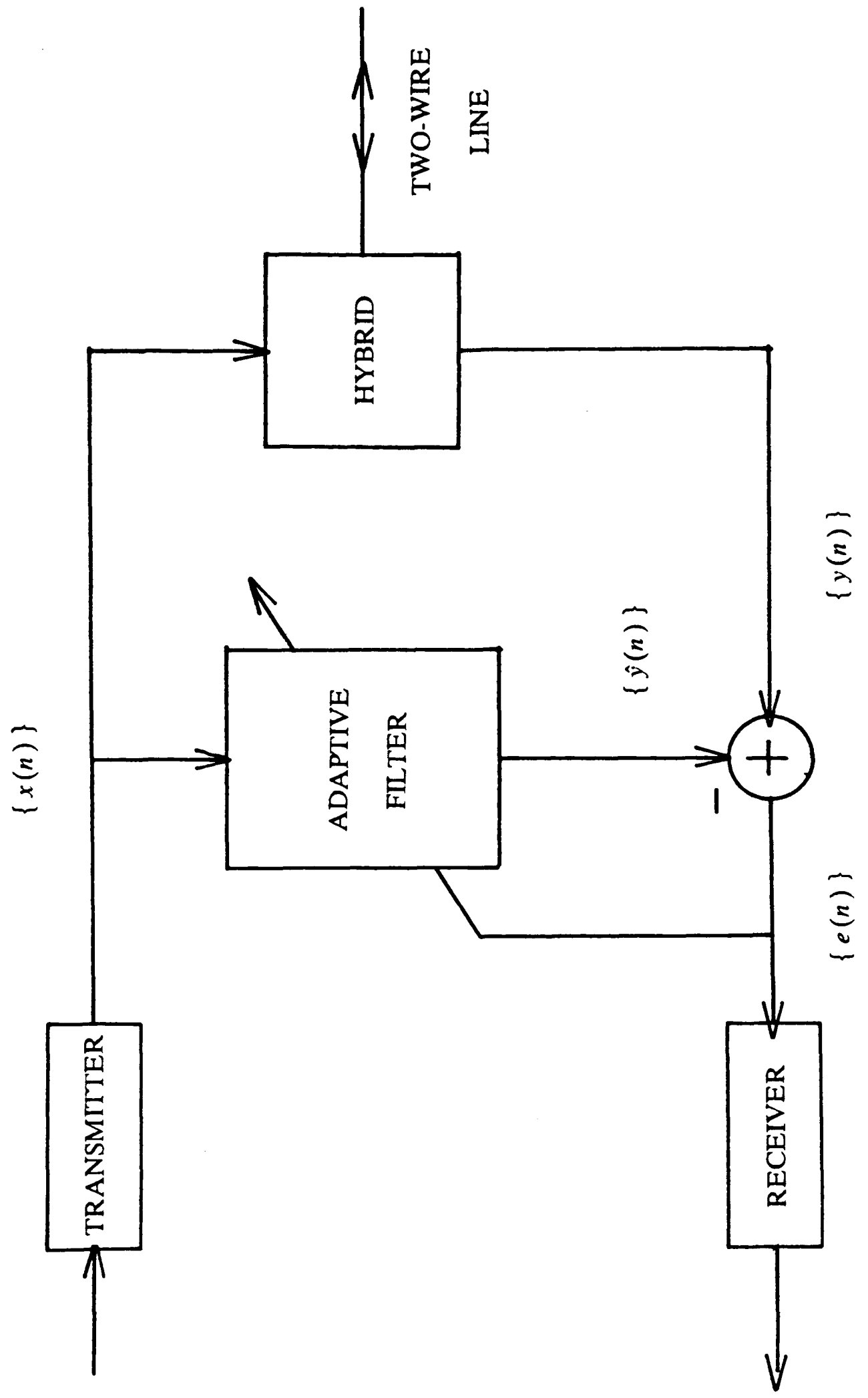


Figure 1.4 ECHO CANCELLATION



sequence, $\{ x(n) \}$ (Figure 1.5). If the linear distortion and additive noise could be characterised then the techniques developed by Wiener and Kalman could be applied to design an optimal filter. Application of the corrupted sequence, $\{ x(n) \}$, to the optimal filter would produce a third sequence which would be close in a minimum mean-square error (MMSE) sense to the original sequence. In a simple case where the unknown system consists of a minimum phase filter alone [7], the optimal filter is the inverse of the minimum phase filter and the output of the optimal filter is the original sequence ie. perfect reconstruction. When the system which causes the distortion is unknown and hence an optimal filter cannot be designed a priori, an adaptive filter solution is possible if the original uncorrupted sequence is accessible for a limited period, the convergence time.

A practical example of this second mode of operation is the equalisation of intersymbol interference on a digital communications channel [8]. Such a channel may be modelled by an equivalent discrete time transversal filter with additive white noise [9]. The digital signal which is applied to the channel is a sequence of symbols taken randomly from a finite alphabet. If the impulse response of the transversal filter consists of anything other than a single impulse, the elements of the output sequence will contain contributions from several symbols as well as noise ie. intersymbol interference. The function of the adaptive filter is to reconstruct the transmitted symbol sequence in a MMSE sense from the received sequence before a final decision is made as to which symbol was transmitted (Figure 1.6). A training sequence for the adaptive filter is obtained by transmitting a predetermined sequence, known to the receiver, as a precursor to actual data. Subsequent to this training period it is still possible to track slow variations in the channel characteristics by using the output of the decision circuit as a training sequence for the adaptive filter. This is known as a decision directed equaliser [8].

The final mode of operation of an adaptive filter that will be considered in this section is linear prediction [10]. This mode differs from the previous two is that the

Figure 1.5 INVERSE SYSTEM MODELLING

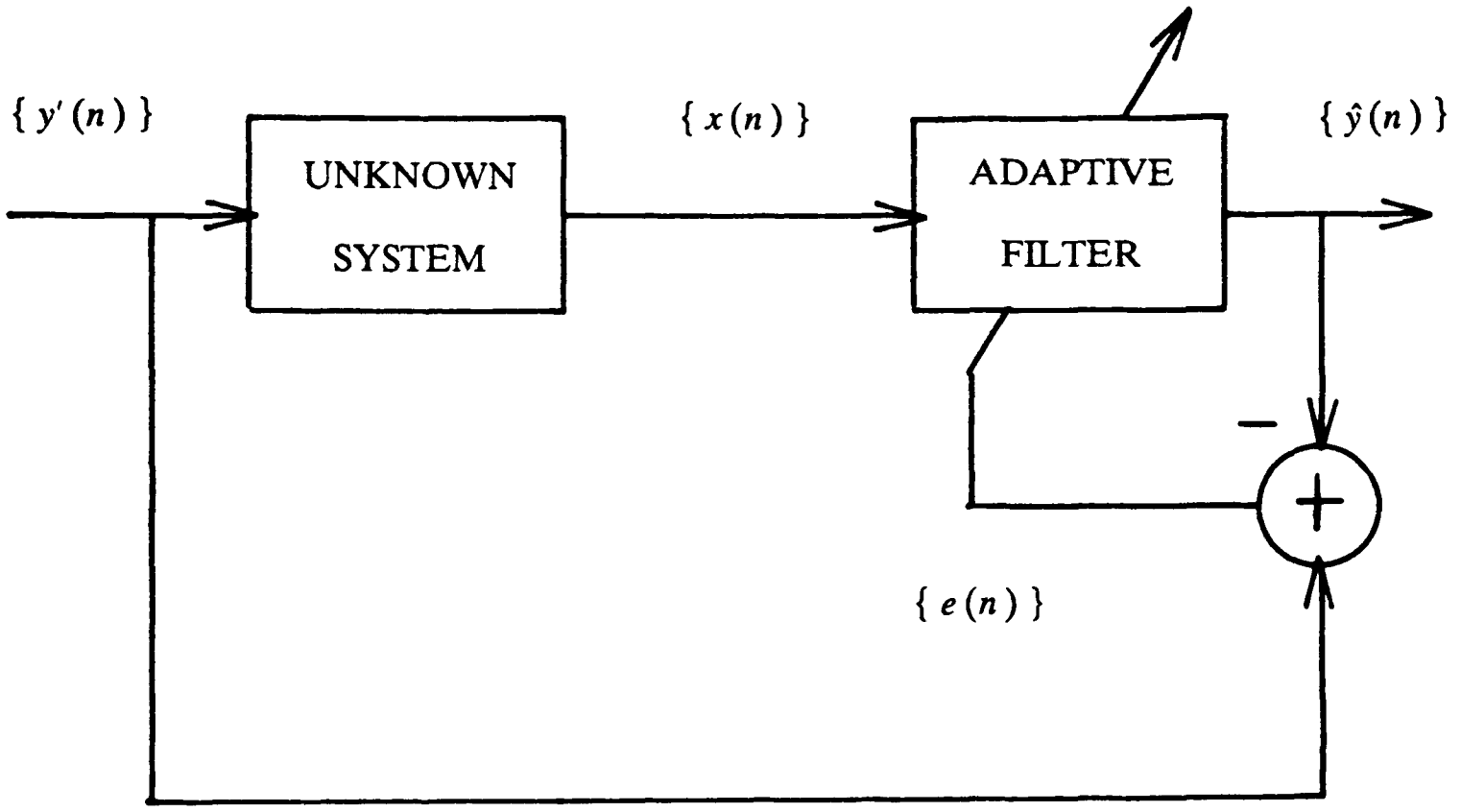
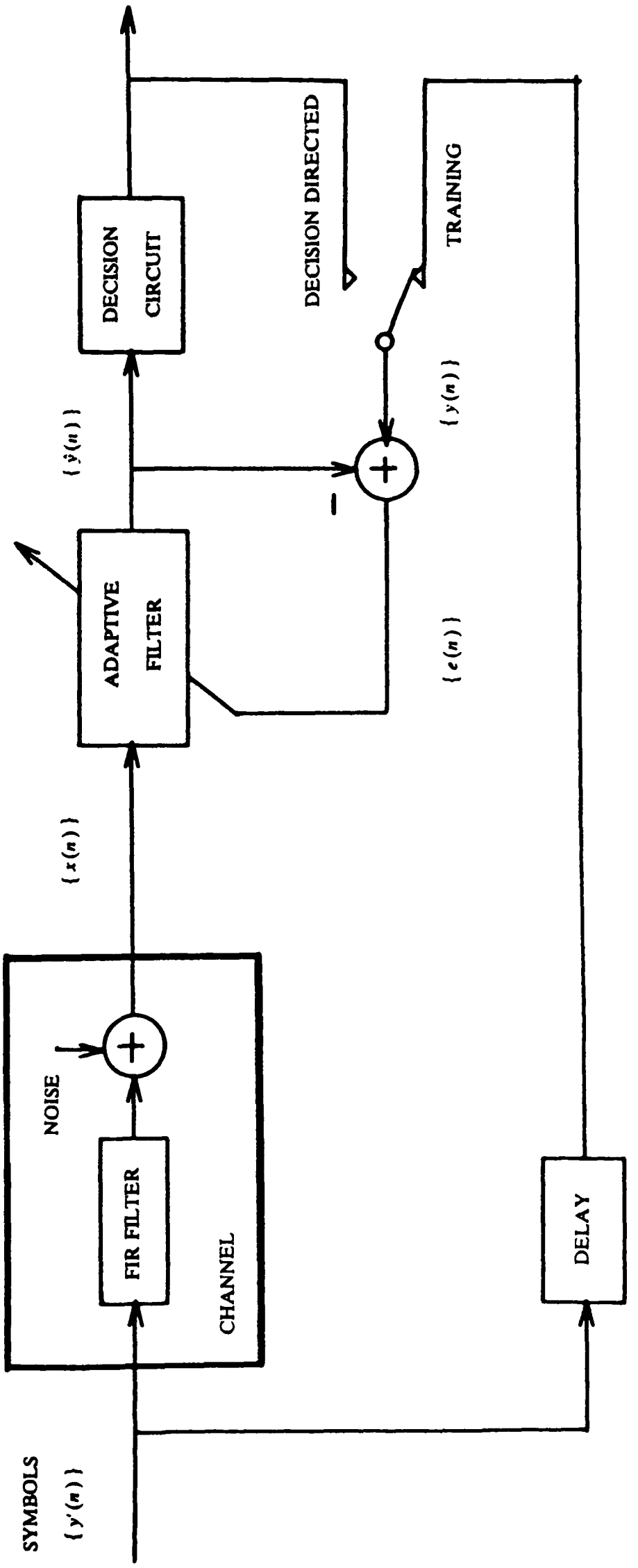


Figure 1.6 CHANNEL EQUALISATION



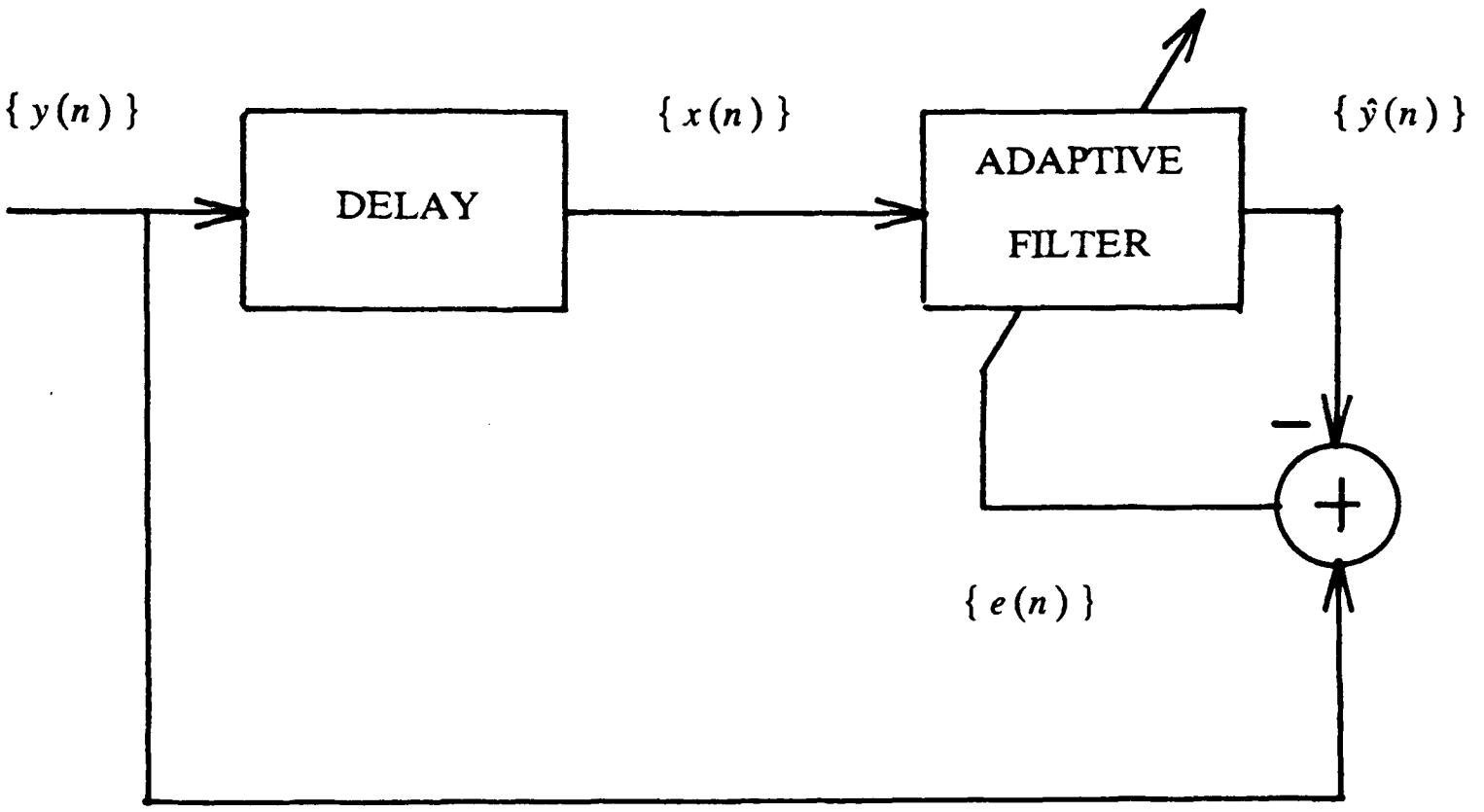
adaptive filter operates on a single sequence rather than two sequences. The function of the adaptive filter in this mode is to either characterise the sequence or to separate the correlated, coloured or predictable part of the sequence from the uncorrelated, white unpredictable part. This arrangement is illustrated in Figure 1.7. The optimal filter for this mode of operation is obtained by the minimum phase factorisation of the spectral density of the sequence $\{ y(n) \}$. The inverse of the minimum phase filter is a realisable whitening filter [5]. Practical examples of linear prediction can be found in spectral estimation [11], linear predictive coding of speech [12] and automatic enhancement of sinsusoids in noise [13].

1.3 THESIS LAYOUT

To return to the theme enunciated at the start of the Chapter, this thesis is primarily concerned with the design of algorithms for discrete time adaptive filtering. The foregoing two sections are included for the purposes of definition and to indicate the need for, and briefly discuss practical applications of, adaptive filtering. The thesis is divided into two halves: the first, containing Chapters 2, 3 and 4, is devoted solely to adaptive finite impulse response (FIR) filter algorithms; the second, containing chapters 5 and 6, documents the development of an adaptive infinite impulse response (IIR) linear equaliser for digital communications channels.

In Chapter 2, a broad selection of adaptive finite impulse response (FIR) filter algorithms are examined to assess relative convergence performance (as indicated by currently available theoretical results) and computational requirements. From this examination a classification system evolves in which the available algorithms are grouped into three classes according to performance and complexity. Of particular note is the unified approach to block least mean squares (BLMS) adaptive filtering [14] which simplifies the application of efficient convolution algorithms other than the fast Fourier transform (FFT) [1] to the construction of computationally efficient adaptive

Figure 1.7 LINEAR PREDICTION



filters.

The classification system is confirmed in Chapter 3 where the convergence performance of the various algorithms is compared by computer simulation in the specific application areas of system identification and channel equalisation. It is believed that such a comparison has not previously been attempted even in the recent textbooks on the subject [15, 16, 17].

A new adaptive FIR filter algorithm is presented in Chapter 4. Analytic and experimental results confirm that this so-called self orthogonalised block adaptive filter (SOBAF) provides a unique combination of convergence performance and computational efficiency.

In Chapter 5 a closed form solution to the MMSE linear equaliser problem is derived using discrete time Wiener filtering theory. This formulation highlights the structure of the optimum IIR equaliser and the difficulties incurred in developing an adaptive IIR equaliser.

Central to Chapter 6 is the recognition that the optimum IIR equaliser can be realised as a particular case of the Kalman equaliser of [18]. To make the Kalman equaliser adaptive, an adaptive FIR filter is operated in parallel with the equaliser to estimate the impulse response of the unknown channel. Combining two algorithms in this manner leads to problems of interaction which are overcome through the development of a novel compensation technique. Comparisons of the performance of this adaptive Kalman equaliser with a conventional linear equaliser are provided.

Finally Chapter 7 summarises the conclusions that have been drawn and provides suggestions for further investigation.

Chapter 2

ADAPTIVE FIR FILTER ALGORITHMS

2.1 INTRODUCTION

The aims of this chapter are threefold; (i) to describe and define a broad selection of adaptive FIR filter algorithms, (ii) to give an indication of the convergence performance that currently available theoretical results would predict for these algorithms, and (iii) to provide a comparison of the computational requirements of the algorithms. The function of an adaptive FIR filter algorithm was identified in the seminal work of Widrow [19, 20, 21], and that is to find the optimum FIR filter from available data rather than from the second order statistics of the data. Widrow used the Wiener minimum mean-square error (MMSE) definition of optimum [2]. Thus in section 2.2 the MMSE cost function is defined and an expression for the optimum MMSE FIR filter is given in terms of autocorrelation and cross-correlation functions [22]. To illustrate the role of the Wiener FIR filter in the design of adaptive filter systems, the important problem of system identification is examined.

Application of the Wiener FIR filter to a signal estimation, prediction or smoothing problem requires explicit knowledge of an autocorrelation function and a cross-correlation function. In practice, these statistical functions may be unknown or time-varying. The heuristic sampled matrix (SM) solution, presented in section 2.3, is to estimate the necessary terms in the auto- and cross- correlation functions from the available data and proceed to the Wiener solution as if the estimates were exact [23]. Alternatively in the least squares (LS) approach of section 2.4, the statistical MSE cost function is replaced with the data dependent LS error cost function [24]. The solution to the LS minimisation problem is in terms of the available data and may be updated using a time recursion as new data appears. This LS estimate will converge to the optimum Wiener FIR filter as the amount of data increases provided the random

processes are stationary. Although the recursive least squares (RLS) algorithm exhibits consistent convergence properties it is computationally expensive to implement even with the availability of the fast algorithms such as [25]. The stochastic gradient least-mean-squares (LMS) [21] and block least-mean-squares (BLMS) [14] algorithms of section 2.5 provide computationally less expensive alternatives to the RLS algorithm. However most of the available theoretical results [21, 26, 27, 28] and practical experiment [26] indicates that the convergence properties of the stochastic gradient algorithms is highly dependent on the autocorrelation function associated with the input signal to the FIR filter. These observations lead to the transform domain or quasi-orthogonalising adaptive filter algorithms of section 2.6. The philosophy behind these algorithms is to approximately whiten the input signal before applying an LMS algorithm. This has the effect of reducing the sensitivity of the LMS algorithm to the autocorrelation function associated with the input signal [29]. Finally in section 2.7 the computational requirements of the algorithms discussed in this chapter are compared.

2.2 OPTIMUM LINEAR ESTIMATION

The structure of a typical linear signal estimation problem is illustrated in Figure 2.1. Given an observed random sequence $\{ x(n) \}$ which is a distorted version of a signal or information-bearing random sequence $\{ y(n) \}$, find a linear filter which operates on $\{ x(n) \}$ to yield an estimate, $\{ \hat{y}(n) \}$, of $\{ y(n) \}$. The quality of the estimate is a function, f(.), of the error $\{ e(n) \}$, which is the difference between the information-bearing sequence and the estimated sequence.

$$e(n) = y(n) - \hat{y}(n) \tag{2.2.1}$$

The loss function $f(e(n))$ assigns a price or penalty incurred when the estimate is incorrect [4]. Clearly the loss function should be: (i) positive

$$f(\,e(n)\,) \geq 0$$

and (ii) non-decreasing.

$$f(\,0\,) = 0$$

$$f(\,e_2\,) \geq f(\,e_1\,) \quad if \;\; e_2 \geq e_1 = 0$$

Examples of loss functions with these properties are: e^2, e^4, and $|\,e\,|$. Since both $\{\,x(n)\,\}$ and $\{\,y(n)\,\}$ are random sequences, $\{\,e(n)\,\}$ is also a random sequence. Hence an optimal or best choice for the linear filter is that which minimises a cost function $l(.)$, which is the expected value of the loss function $f(.)$.

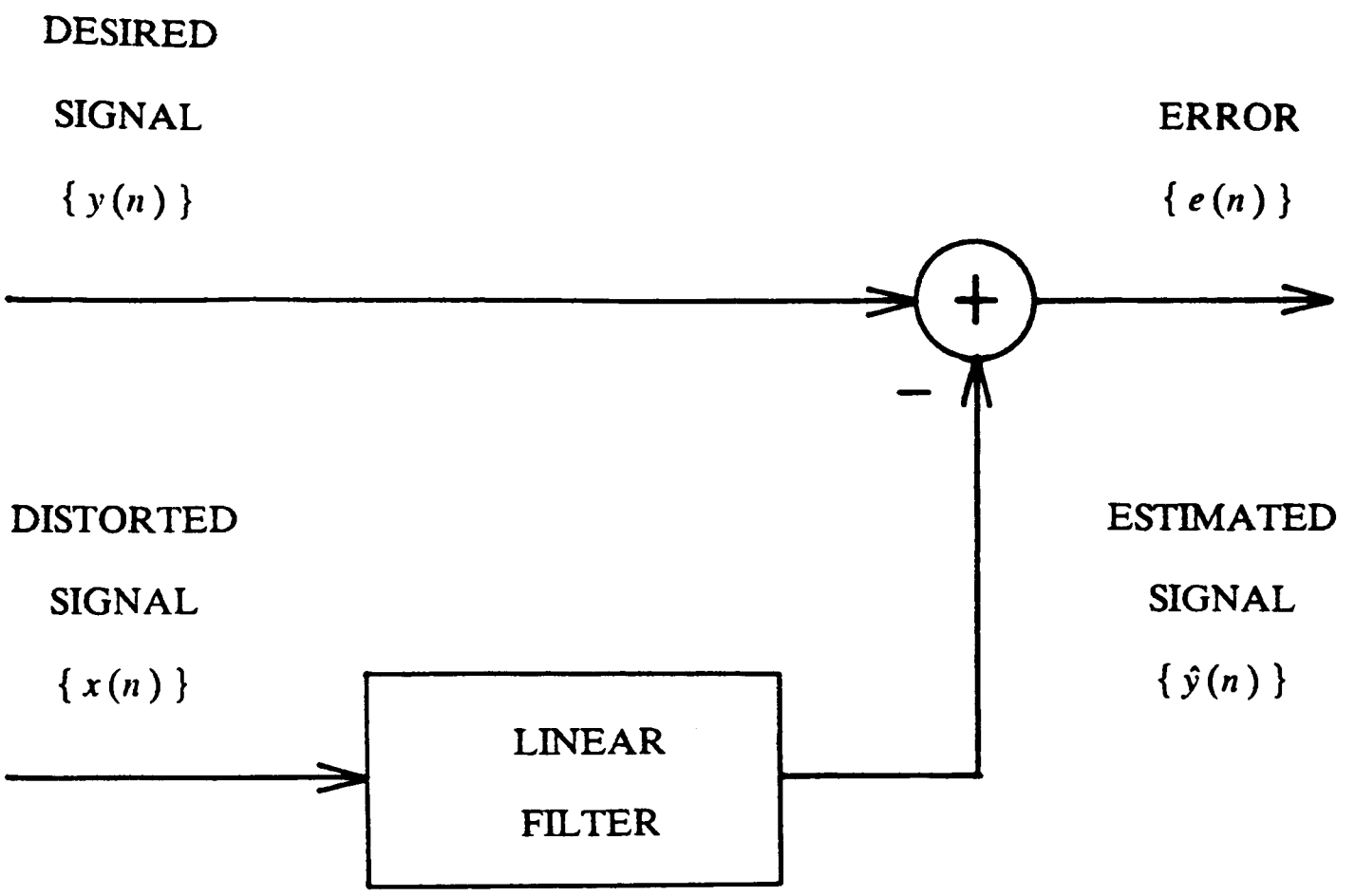
$$l(\,e(n)\,) = E[\,f(\,e(n)\,)\,]$$

The most commonly used cost function, and the one adopted here, is the mean-square error [2] (MSE) $\xi(n)$.

$$\xi(n) = E[\,e^2(n)\,] \tag{2.2.2}$$

Thus the optimal filter is defined as that filter of of the set of all possible linear filters which minimises the MSE.

Figure 2.1 OPTIMUM LINEAR ESTIMATION



2.2.1 The Optimum FIR Filter

The output $\hat{y}(n)$ of a causal linear filter may be written as the convolution of the input sequence $\{ x(n) \}$ and the impulse response sequence $\{ h_n \}$.

$$\hat{y}(n) = \sum_{i=0}^{+\infty} h_i \, x(n-i)$$

This is by definition an IIR filter since it includes terms to $h_\infty x(n-\infty)$. In this section only FIR filters will be discussed leaving consideration of IIR filters until Chapter 4. The output of a FIR filter of order N-1 may be written as a finite summation of N products since

$$h_n = 0 \quad for \quad n \geq N, n < 0$$

Thus

$$\hat{y}(n) = \sum_{i=0}^{N-1} h_i \, x(n-i)$$

This finite sum of products may be written more compactly as a vector inner product.

$$\hat{y}(n) = \underline{h}^T \underline{x}(n) \tag{2.2.3}$$

where $\underline{h}$ is a column vector containing the N non-zero elements of the impulse response sequence $\{ h_n \}$

$$\underline{h} = [\, h_0 \, h_1 \, \cdots \, h_{N-1} \,]^T$$

and $\underline{x}(n)$ is a column vector containing the last N elements of the input sequence $\{ x(n) \}$.

$$\underline{x}(n) = [\, x(n) \, x(n-1) \, \cdots \, x(n-N+1) \,]^T$$

The superscript T denotes vector or matrix transposition. The structure of a FIR filter is illustrated in Figure 2.2.

If the sequences $\{ x(n) \}$ and $\{ y(n) \}$ are wide sense stationary then substitution of (2.2.1) and (2.2.3) into (2.2.2) yields an expression for the MSE cost function

$$\xi = E[y^2] + \underline{h}^T \Phi_{xx} \underline{h} - 2 \underline{h}^T \Phi_{xy} \tag{2.2.4}$$

where Φ_{xx} is an $(N \times N)$ autocorrelation matrix

$$\Phi_{xx} = E[\underline{x}(n) \underline{x}^T(n)] \tag{2.2.5}$$

and Φ_{xy} is an N element cross-correlation vector.

$$\Phi_{xy} = E[\underline{x}(n) y(n)] \tag{2.2.6}$$

Thus for a FIR filter the MSE cost function has a quadratic form in the impulse response vector $\underline{h}$ and the minimum can be obtained by setting the gradient N-vector $\underline{\nabla}$ to zero [22].

$$\underline{\nabla} = \frac{\partial \xi}{\partial \underline{h}}$$

$$= [\frac{\partial \xi}{\partial h_0} \frac{\partial \xi}{\partial h_1} \cdots \frac{\partial \xi}{\partial h_{N-1}}]^T$$

$$= 2 \Phi_{xx} \underline{h} - 2 \Phi_{xy} = 0 \tag{2.2.7}$$

The optimum impulse response $\underline{h}_{opt}$ which minimises the MSE is thus the solution to a set of N simultaneous linear equations.

$$\Phi_{xx} \underline{h}_{opt} = \Phi_{xy} \tag{2.2.8}$$

If the power spectral density of the input sequence $\{ x(n) \}$ has no nulls ie. frequencies where it is zero, then the autocorrelation matrix Φ_{xx} is positive definite and hence is nonsingular. Under this condition, the optimum impulse response is unique and is is given by

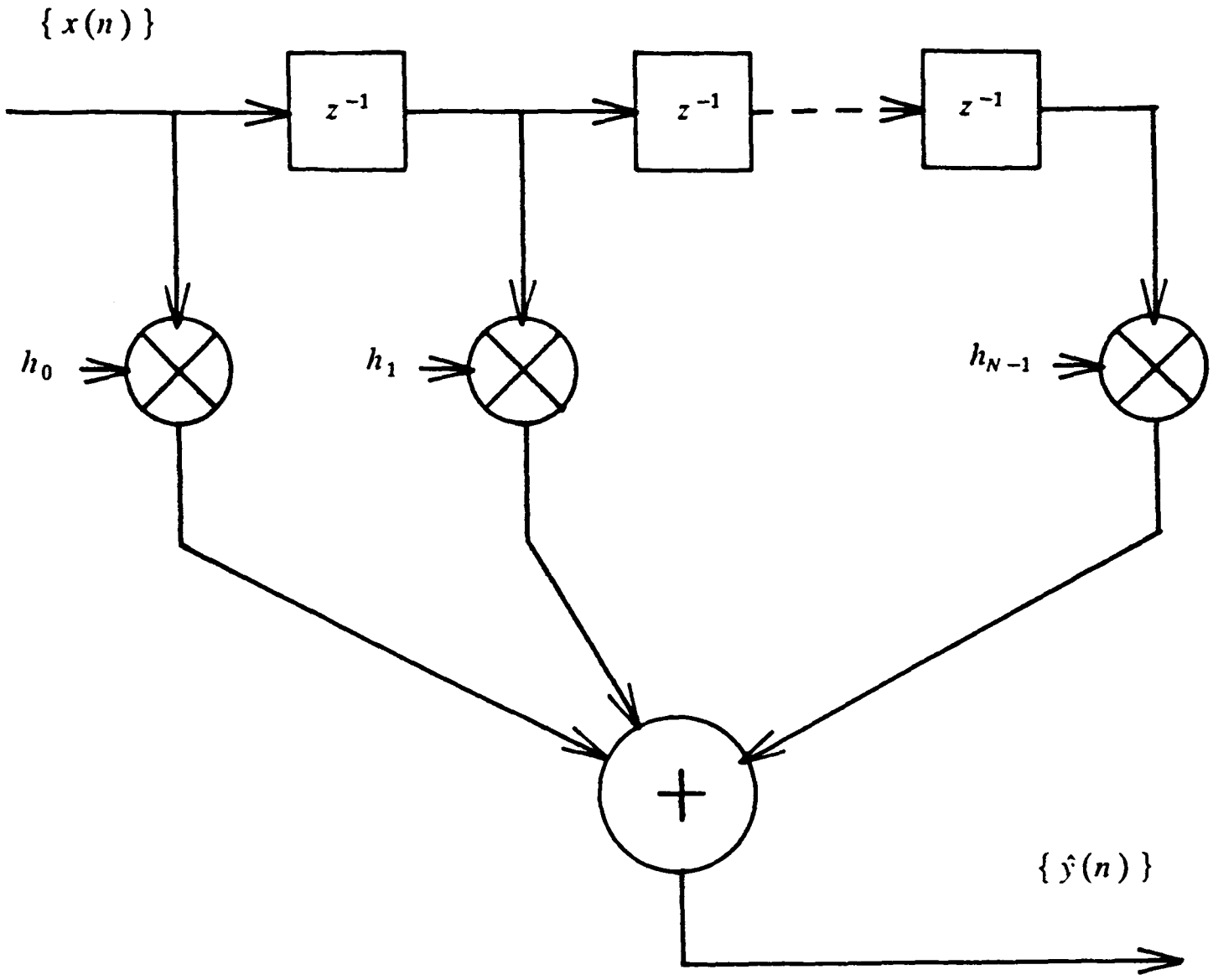
$$\underline{h}_{opt} = \Phi_{xx}^{-1} \Phi_{xy} \tag{2.2.9}$$

The filter defined by (2.2.9) is the Wiener FIR filter or Levinson filter. The minimum MSE, ξ_{opt}, is obtained by substitution of (2.2.9) in (2.2.4).

$$\xi_{opt} = E[\, y^2 \,] - \underline{h}_{opt}^T \, \underline{\Phi}_{xy} \tag{2.2.10}$$

Equation (2.2.9) provides a means for designing optimum linear FIR filters. However in order to calculate the impulse response of the optimum filter precise knowledge of the autocorrelation matrix and the cross correlation vector is required. In practice it is the data sequences rather then their second order statistics that are directly available. Determining the optimal filter from the data rather then the second order statistics is the function of an adaptive FIR filter [21]. An adaptive FIR filter can be defined as an algorithm which operates on the sequences $\{ x(n) \}$ and $\{ y(n) \}$ to form a time-varying impulse response vector $\underline{h}(k)$ which converges in the mean as $k \rightarrow \infty$ to the optimum impulse response $\underline{h}_{opt}$. The Wiener FIR filter is thus the goal of adaptive FIR filtering and can provide insight into how the adaptive filter should be applied and what the performance might be once the algorithm has converged. In order to highlight the role of the Wiener FIR filter in the design of adaptive filters, the important problem of FIR system identification is examined.

Figure 2.2 A FINITE IMPULSE RESPONSE FILTER



2.2.2 FIR System Identification

Consider the system identification problem illustrated in Figure 2.3. An unknown FIR system with N-point impulse response vector $\underline{h}_s$ has an input sequence $\{ \alpha(n) \}$ and an output sequence $\{ \beta(n) \}$. They are related by a vector inner product expression similar to (2.2.3).

$$\beta(n) = \underline{h}_s^T \underline{\alpha}(n)$$

where

$$\underline{\alpha}(n) = [\, \alpha(n)\ \alpha(n-1)\ \cdots\ \alpha(n-N+1)\,]^T$$

In forming a model, denoted by the N-point impulse response vector $\underline{h}$, of the unknown system all that is available are two sequences $\{ x(n) \}$ and $\{ y(n) \}$ which are noisy observations of the input and output sequence respectively.

$$x(n) = \alpha(n) + \upsilon(n)$$

$$y(n) = \beta(n) + \eta(n)$$

The sequences $\{ \alpha(n) \}$, $\{ \upsilon(n) \}$, and $\{ \eta(n) \}$ are assumed to wide sense stationary mutually *uncorrelated* random processes. The noise sequences $\{ \upsilon(n) \}$, and $\{ \eta(n) \}$ are white with variances σ_υ and σ_η respectively.

Application of the Wiener filter to this problem involves constructing an estimate $\hat{y}(n)$ of the observed output $y(n)$ by passing the observed input sequence $\{ x(n) \}$ through a system modelling filter with impulse response vector $\underline{h}$. The impulse response of the system model is chosen to minimise the MSE.

$$E[\, (\, y(n) - \hat{y}(n)\,)^2\,]$$

The solution, $\underline{h}_{opt}$, provided by (2.2.9), is an estimate of the unknown system impulse response. To calculate this estimate it is necessary to first form the autocorrelation

matrix

$$\Phi_{xx} = \Phi_{\alpha\alpha} + \sigma_v I_N \tag{2.2.11}$$

where

$$\Phi_{\alpha\alpha} = E[\, \underline{\alpha}(n)\, \underline{\alpha}^T(n)\,]$$

and then the cross correlation vector.

$$\Phi_{xy} = E[\, \underline{\alpha}(n)\, \beta(n)\,]$$

$$= \Phi_{\alpha\beta} \tag{2.2.12}$$

I_N is the $(N \times N)$ identity matrix. Substitution of (2.2.11) and (2.2.12) in (2.2.9) yields.

$$\underline{h}_{opt} = (\Phi_{\alpha\alpha} + \sigma_v I_N)^{-1} \Phi_{\alpha\beta}$$

which after application of the matrix inversion lemma [24] can be re-arranged to give an expression for the impulse response of the Wiener FIR filter in terms of the impulse response of the unknown system.

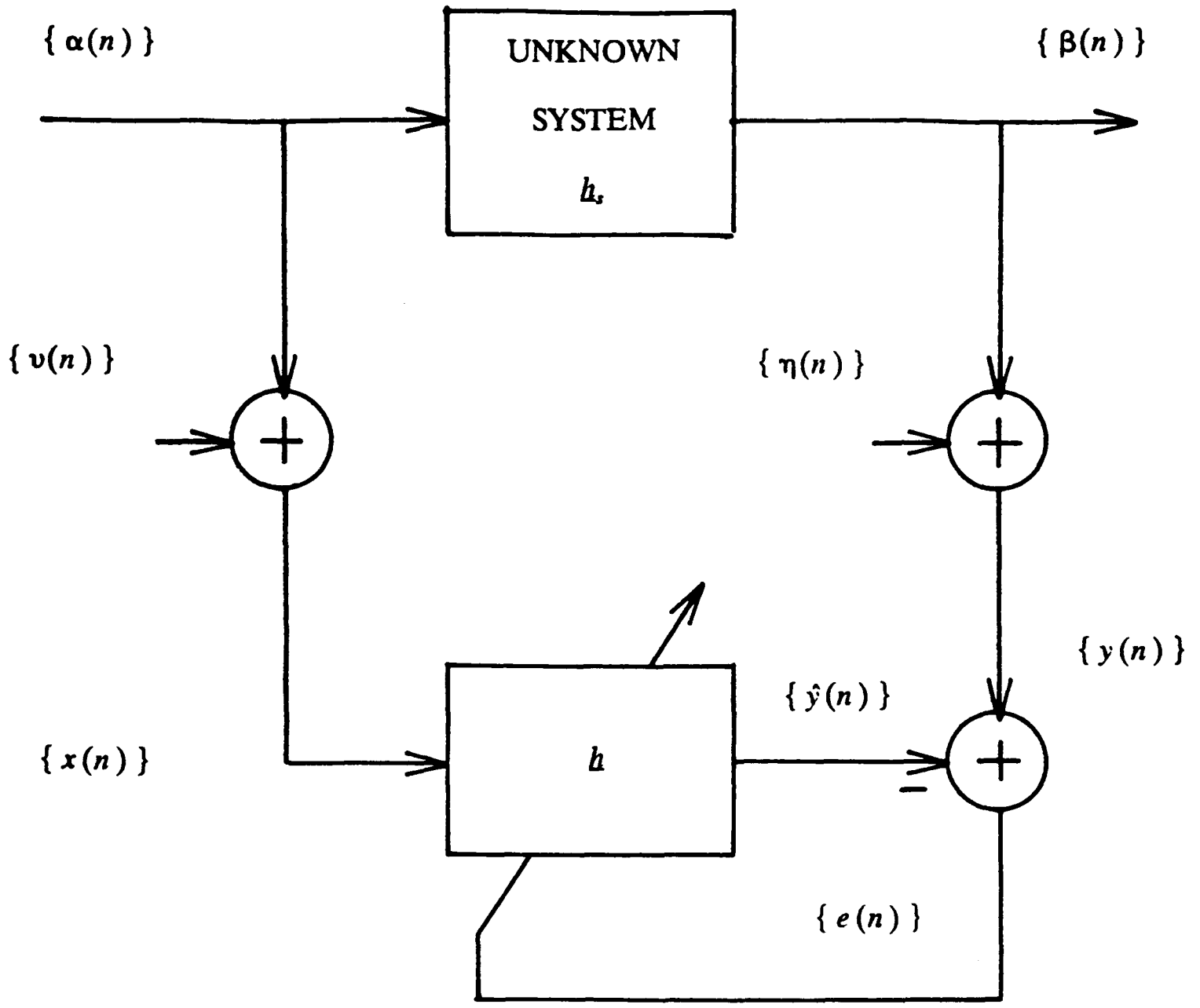
$$\underline{h}_{opt} = \underline{h}_s - \sigma_v (\Phi_{\alpha\alpha} + \sigma_v I_N)^{-1} \underline{h}_s$$

This equation provides some useful results. First the Wiener filter is a biased estimate of the unknown system impulse response. Second the bias is removed if there is no noise on the input process ie.

$$\sigma_v = 0$$

Finally the noise on the output process, $\{ \eta(n) \}$, does not affect the estimate.

Figure 2.3 FIR SYSTEM IDENTIFICATION



2.3 SAMPLED MATRIX INVERSION

As indicated in the previous section, the optimum tap vector, $\underline{h}_{opt}$, in a MSE sense for an adaptive FIR filter is given by the solution of the Wiener equation, (2.2.9). The calculation of $\underline{h}_{opt}$ from (2.2.9) requires the solution of N simultaneous linear equations in N unknowns. For a general non singular matrix the most efficient method of solution is Gaussian elimination which requires order N^3 operations ie. the number of calculations is proportional to the cube of the number of coefficients in the impulse response vector. However the matrix $\underline{\Phi}_{xx}$ has two structural properties: (i) it is symmetric i.e.

$$\underline{\Phi}_{xx}[i,j] = \underline{\Phi}_{xx}[j,i]$$

(ii) it is Toeplitz i.e.

$$\underline{\Phi}_{xx}[i,j] = \Phi_{xx}(i-j)$$

where $\underline{\Phi}_{xx}[i,j]$ is the element in row i column j of the matrix $\underline{\Phi}_{xx}$ and $\{\Phi_{xx}(m)\}$ is the autocorrelation sequence associated with the wide sense stationary sequence $\{x(n)\}$. All the elements of the autocorrelation matrix can be generated from knowledge of its first row. A more efficient method of solution, which exploits this very special structure, was originally devised by Levinson [22] . The Levinson algorithm requires order N^2 operations, a significant improvement on Gaussian elimination.

The direct application of the Wiener solution requires prior knowledge of the second order statistics of the process. This runs contrary to the concept of adaptive filtering. However, the efficiency of the Levinson algorithm and the availability of current large scale integration (LSI) techniques [30] still makes the direct Wiener solution appear attractive. An intuitive approach, known as sampled matrix inversion (SMI), has been suggested for spectral analysis [11] and automatic equalisation

[23, 31]. This techniques involves two stages.

In the first stage, parameter estimation, the autocorrelation and cross correlation coefficients are estimated from the available data. In the second stage, these estimated coefficients are used to form the Wiener equation which is then solved using the Levinson algorithm. The Wiener equation can thus be rewritten,

$$\underline{h} = \hat{\Phi}_{xx}^{-1} \hat{\Phi}_{xy} \tag{2.3.1}$$

where the symbol $\hat{}$ denotes an estimate. Since $\hat{\Phi}_{xx}$ and $\hat{\Phi}_{xy}$ are estimates of the correlation matrices Φ_{xx} and Φ_{xy} respectively, the solution $\underline{h}$ is no longer the Wiener optimum $\underline{h}_{opt}$ but an estimate of it.

For a wide sense stationary process, the autocorrelation coefficients $\Phi_{xx}(m)$, $m = 0,1,..,N-1$, from which Φ_{xx} is composed are given exactly by the time average.

$$\Phi_{xx}(m) = E[\, x(n)\, x(n+m)\,]$$

$$= \lim_{L \to \infty} \frac{1}{(2L+1)} \sum_{n=-L}^{L} x(n)\, x(n+m) \tag{2.3.2}$$

In practice, only a finite data set $\{ x(n) \}$, $n = 1,2,..,k$, is available. So the summation of (2.3.2) is truncated. Three techniques for doing this have been suggested in the literature. These are described briefly. (Note: only the autocorrelation estimate is considered here as the cross correlation estimate is analogous). An unbiased estimate of the autocorrelation coefficients, as suggested in [11] and [32] is

$$\hat{\Phi}_1(m) = \frac{1}{(k-m)} \sum_{n=1}^{k-m} x(n)\, x(n+m) \tag{2.3.3}$$

$$m = 0,\ 1,\ \cdots\ ,N-1$$

Alternatively, an estimate which has been used for autoregressive spectral estimation [11] is

$$\hat{\Phi}_2(m) = \frac{1}{k}\sum_{n=1}^{k-m} x(n)\,x(n+m) \tag{2.3.4}$$

This is a biased estimate which has a smaller variance than (2.3.3). Finally Butler and Cantoni [23] suggest using

$$\hat{\Phi}_3(m) = \frac{1}{(k-N+1)}\sum_{n=1}^{k-N+1} x(n)\,x(n+m) \tag{2.3.5}$$

The estimate $\hat{\Phi}_3$ differs from the other two estimates in that the number of product terms of the form $x(n)x(n+m)$ which are added to form the estimate is the same for all lags, m. For the estimates $\hat{\Phi}_1$ and $\hat{\Phi}_2$ the number of product terms decreases with increasing lag. In (2.3.3) the summation is divided by the number of products, $(k-m)$, while in (2.3.4) it is divided by the number of data points, k.

While all three techniques will converge to $\Phi_{xx}(m)$ as the number of data points, k, tends to infinity, what is of primary interest from the point of view of adaptive filtering is their performance in the short term eg. when k<10N. Equations (2.3.3) and (2.3.4) are examined in some detail in [33] with the conclusion that both perform poorly in the short term. In fact, it is suggested in [32] that as a rule of thumb k should be greater than 10N. In [23] an expression for the variance of $\hat{\Phi}_3$ is derived which is only valid when k>>N.

2.4 LEAST SQUARES ESTIMATION

The MSE cost function, ξ, includes the expectation operator, $E[.]$, because the processes involved in the estimation problem are random. This cost function leads to an optimum solution, (2.2.8), in terms of ensemble averages or expected values ie. the autocorrelation matrix Φ_{xx} and the cross-correlation vector Φ_{xy}. When only finite data sets, ($x(n)$, $y(n)$, $n = 0, .. k$), rather than ensemble averages are available, the expectation operator, $E[.]$, in the MSE cost function may be replaced by a summation, $\sum$, over the available data, to yield the least squares (LS) cost function.

$$\sum_{n=0}^{k} (y(n) - \hat{y}(n))^2 \tag{2.4.1}$$

Minimisation of this cost function with respect to the impulse response vector, $\underline{h}$, associated with the estimate, $\hat{y}(n)$, leads to a LS estimate. The impulse response vector, $\underline{h}(k)$, which minimises the LS cost function is now a function of the available data rather than ensemble averages. Just as the LS cost function can be obtained from the MSE cost function by replacing expectation with summation, likewise, $\underline{h}(k)$ can be obtained from (2.2.8) by replacing expectation with summation ie. there is a duality between the MSE and LS minimisation problems. Thus

$$\underline{\underline{r}}_{xx}(k)\, \underline{h}(k) = \underline{r}_{xy}(k) \tag{2.4.2}$$

where

$$\underline{\underline{r}}_{xx}(k) = \sum_{n=0}^{k} \underline{x}(n)\, \underline{x}^T(n) \tag{2.4.3}$$

and

$$\underline{r}_{xy}(k) = \sum_{n=0}^{k} \underline{x}(n)\, y(n) \tag{2.4.4}$$

A unique solution to (2.4.2) will exist if $\underline{\underline{r}}_{xx}(k)$ is nonsingular. For small data sets, ie. $k \leq N-1$, $\underline{\underline{r}}_{xx}$ is always singular. In this case, the number of data points, k+1, is less

than the number of unknowns, N, which describe the impulse response and hence no unique solution can exist. For $k \geq N$, the singularity of $\underline{\underline{r}}_{xx}(k)$ will depend on the particular data sets that are available. The singularity problem in LS squares estimation is often avoided through the use of either the Moore-Penrose pseudoinverse [34], where the solution of minimum length is chosen when (2.4.2) does not provide a unique solution, or by adding a small positive scalar, α, to the leading diagonal of $\underline{\underline{r}}_{xx}(k)$.

$$\underline{\underline{r}}_{xx}(k) = \sum_{n=0}^{k} \underline{x}(n)\, \underline{x}^T(n) + \alpha\, I_N \tag{2.4.5}$$

The latter method while ensuring that the solution to (2.4.2) is always unique is not in a strict sense LS.

Four different forms of LS estimate are possible depending upon what assumptions are made about the available data [35]. These are the covariance form, the pre-windowed form, the post-windowed form and the autocorrelation form. Of these the covariance form is the most straightforward as it involves no assumptions about the data. The other three forms are constructed from permutations of the pre-windowed assumption,

$$x(n) = 0, \quad n < 0$$

and the post-windowed assumption.

$$x(n) = 0, \quad n > k - N + 1$$

The autocorrelation form is at the other extreme from the covariance form combining both pre-windowed and post-windowed assumptions. These data windows are usually invoked with the view to reducing the number of computations necessary to solve (2.4.2). However as the available data is usually taken from continuous data sequences the pre-windowed and post windowed assumptions will in general be invalid and hence will lead to a degradation in the quality of the LS estimate.

Unlike the autocorrelation matrix, Φ_{xx}, the LS matrix, $\underline{\underline{r}}_{xx}(k)$, is not Toeplitz. However it does have a significant structural property which can be identified by rewriting the summation of (2.4.3) as the product of two matrices.

$$\underline{\underline{r}}_{xx}(k) = \sum_{n=0}^{k} \underline{x}(n)\,\underline{x}^T(n)$$

$$= \underline{\underline{x}}(k)\,\underline{\underline{x}}^T(k) \qquad (2.4.6)$$

where $\underline{\underline{x}}(k)$ is a $N \times (k+1)$ rectangular matrix.

$$\underline{\underline{x}}(k) = [\,\underline{x}(0) \;\cdots\; \underline{x}(n) \;\cdots\; \underline{x}(k)\,]$$

Since $\underline{\underline{x}}(k)$ is Toeplitz, $\underline{\underline{r}}_{xx}(k)$ is the product of two Toeplitz matrices. This near to Toeplitz structure [36] has lead to the development of efficient algorithms for the solution of (2.4.2) which lie between Gaussian elimination and the Levinson algorithm in computational complexity [37, 38]. If however the autocorrelation form is assumed, ie. the data is both pre- and post-windowed, the LS matrix $\underline{\underline{r}}_{xx}(k)$ becomes Toeplitz and the computational load is further reduced since the Levinson recursion can be used to solve (2.4.2).

An expression for the element in row i column j of the matrix $\underline{\underline{r}}_{xx}(k)$ can be written down directly from (2.4.3).

$$r_{xx}[i,j] = \sum_{n=0}^{k} x(n-i)\,x(n-j) \qquad (2.4.7)$$

The time index (k) has been dropped from $\underline{\underline{r}}_{xx}(k)$ for this discussion to simplify the notation. To prove that the autocorrelation form of $\underline{\underline{r}}_{xx}(k)$ is Toeplitz it is sufficient to show that (2.4.7) may be rewritten as a function of a single variable (i-j) as opposed to a function of two variables i and j. First a change of variable is made to highlight the effect of the pre-windowed assumption. Let

$$m = n - i$$

and replace n in (2.4.7) with m+i.

$$r_{xx}[i,j] = \sum_{m=-i}^{k-i} x(m)\, x(m+i-j) \tag{2.4.8}$$

Since $r_{xx}(k)$ is symmetric, (2.4.6), generality will not be lost if it is assumed that

$$i - j \geq 0 .$$

In which case the pre-windowed assumption implies that the range of m must be limited at its lower end to

$$m \geq 0$$

and hence (2.4.8) may be rewritten

$$r_{xx}[i,j] = \sum_{m=0}^{k-i} x(m)\, x(m+i-j) .$$

Performing a second change of variable,

$$l = m + i - j ,$$

which highlights the effect of the post-windowed assumption, yields the desired result.

$$r_{xx}[i,j] = \sum_{l=i-j}^{k-N-1} x(l)\, x(l+i-j) \tag{2.4.9}$$

Equation (2.4.9) is similar in form to the autocorrelation estimate $\hat{\Phi}_2$ of (2.3.4) since the number of products $x(l)x(l+i-j)$ that are added decreases linearly with increasing lag, $i-j$. In fact for a given data set the SM technique based on (2.3.4) will produce the same estimate of h_{opt} as the autocorrelation form of LS.

2.4.1 Recursive Least Squares

In many applications it is necessary to update the LS estimate provided by the solution to (2.4.2) as new data becomes available. The simplest approach is to reconstruct (2.4.2) and resolve it. However this is equivalent to performing a matrix inversion as each new data point becomes available and thus has the possibility of being expensive computationally. An alternative is to seek a time recursion for $\underline{h}(k)$ in terms of the previous least squares solution $\underline{h}(k-1)$ and the new data, $\underline{x}(k)$ and y(k). A recursive solution for $\underline{h}(k)$, may be obtained as follows. From the definitions of (2.4.3) and (2.4.4) and assuming that the data has not been post-windowed.

$$\underline{\underline{r}}_{xx}(k) = \underline{\underline{r}}_{xx}(k-1) + \underline{x}(k)\,\underline{x}^T(k) \tag{2.4.10}$$

$$\underline{r}_{xy}(k) = \underline{r}_{xy}(k-1) + \underline{x}(k)\,y(k) \tag{2.4.11}$$

Substitute for $\underline{r}_{xy}$ in (2.4.11) using (2.4.2).

$$\underline{\underline{r}}_{xx}(k)\,\underline{h}(k) = \underline{\underline{r}}_{xx}(k-1)\,\underline{h}(k-1) + \underline{x}(k)\,y(k)$$

Then using (2.4.10).

$$\underline{\underline{r}}_{xx}(k)\,\underline{h}(k) = \left\{ \underline{\underline{r}}_{xx}(k) - \underline{x}(k)\,\underline{x}^T(k) \right\} \underline{h}(k-1) + \underline{x}(k)\,y(k)$$

After rearrangement this yields

$$\underline{h}(k) = \underline{h}(k-1) + \underline{k}(k)\,e(k) \tag{2.4.12}$$

where

$$e(k) = y(k) - \underline{h}^T(k-1)\,\underline{x}(k) \tag{2.4.13}$$

and

$$\underline{k}(k) = \underline{\underline{r}}_{xx}^{-1}(k)\,\underline{x}(k) \tag{2.4.14}$$

A recursion for $r_{xx}^{-1}(k)$ may be obtained by application of the Sherman-Morrison identity [39] to (2.4.10).

$$r_{xx}^{-1}(k) = r_{xx}^{-1}(k-1) - \frac{r_{xx}^{-1}(k-1)\, x(k)\, x^T(k)\, r_{xx}^{-1}(k-1)}{\left(1 + x^T(k)\, r_{xx}^{-1}(k-1)\, x(k) \right)} \tag{2.4.15}$$

This recursion is often initialised using the definition of (2.4.5) ie.

$$r_{xx}(0) = \frac{1}{\alpha} I_N$$

Collectively, (2.4.12), (2.4.13), (2.4.14) and (2.4.15) are known as the recursive least squares (RLS) algorithm. Godard [40] derived an almost identical algorithm by using a Kalman filter [3] to update the coefficients of an adaptive transversal equaliser. The number of arithmetic operations per iteration that are necessary for the RLS is: $\frac{3N^2}{2} + \frac{5N}{2}$ additions/subtractions, $\frac{3N^2}{2} + \frac{9N}{2}$ multiplications, and 1 division. Thus it can be an expensive algorithm to implement even for a system of moderate order.

2.4.2 Data Windows

In a non-stationary or time varying environments, the least squares estimate provided by (2.4.2) and the corresponding recursive least squares algorithm of subsection 2.4.1 are inappropriate [24]. This is because the tap vector, $\underline{h}(k)$, associated with this growing memory LS estimate [24] is a function of all the data in a window from $n = 0$ to $n = k$. The optimum filter may itself be time varying within the data window. One possible approach to this problem is to replace the sum of squares cost function of (2.4.1) with an exponentially weighted sum of squares cost function

$$\sum_{n=0}^{k} (y(n) - \hat{y}(n))^2 \lambda^{k-n} , \tag{2.4.16}$$

where

$$0 \leq \lambda \leq 1 .$$

This essentially reduces the effect of old data samples on the current estimate, $\underline{h}(k)$, of the optimum tap vector. The parameter, λ, controls the length of the memory. When $\lambda = 1$ all the available data is weighted equally and this exponential LS algorithm reduces to the growing memory form discussed previously. However as λ is reduced the effective memory of the algorithm is also reduced and the algorithm may then be capable of tracking changes in the optimum tap vector.

Minimisation of the cost function of (2.4.16) give rise to a recursive least squares algorithm which is very similar in form to that presented in subsection 2.4.1. The only difference is that (2.4.15) is now replaced [24] with

$$\underline{r}_{xx}^{-1}(k) = \frac{1}{\lambda} \left[\underline{r}_{xx}^{-1}(k-1) - \frac{\underline{r}_{xx}^{-1}(k-1)\, \underline{x}(k)\, \underline{x}^T(k)\, \underline{r}_{xx}^{-1}(k-1)}{\left(\lambda + \underline{x}^T(k)\, \underline{r}_{xx}^{-1}(k-1)\, \underline{x}(k) \right)} \right] .$$

Thus the recursive form of the exponentially weighted LS algorithm represents only a

slight increase in computational load over the growing memory form. Other data windows are possible. These include the sliding window least squares algorithm where the cost function of (2.4.1) is altered to

$$\sum_{n=k-M+1}^{k} (y(n) - \hat{y}(n))^2$$

in order to include only the last M data points in the estimate of the optimal tap vector. However the computational burden of the sliding form is significantly greater than both the growing memory and exponentially weighted least squares algorithms [24]. There is also evidence to suggest that the sliding window algorithm may not offer any improvement in performance over the exponential form in nonstationary environments. [41]

2.4.3 Fast Algorithms

While equations (2.4.12)-(2.4.13) provide a RLS algorithm, the near to Toeplitz structure of the LS matrix $\underline{\underline{L}}_{xx}(k)$ is not exploited in their derivation. This structure is a result of the shifting property of the data vector $\underline{x}(k)$. The shifting property can be illustrated by comparing $\underline{x}(k)$ with $\underline{x}(k-1)$ and observing that they have N-1 common elements.

$$\underline{x}^T(k) = [\, x(k) \;\; x(k-1)\; x(k-2) \;\cdots\; x(k-N+2)\; x(k-N+1) \,]$$

$$\underline{x}^T(k-1) = [\, x(k-1)\; x(k-2)\; x(k-3) \;\cdots\; x(k-N+1)\; x(k-N) \;\;]$$

In fact $\underline{x}^T(k)$ can be obtained from $\underline{x}^T(k-1)$ by shifting the elements of $\underline{x}^T(k-1)$ to the the right, hence losing $x(k-N)$, and then replacing the leftmost element of $\underline{x}^T(k-1)$ with x(k). In 1978 Ljung et al [25] derived a recursion for the calculation of $\underline{k}(k)$, which uses the concept of forward and backwards least squares linear prediction to exploit the shifting property. This recursion, known as the fast Kalman algorithm, is

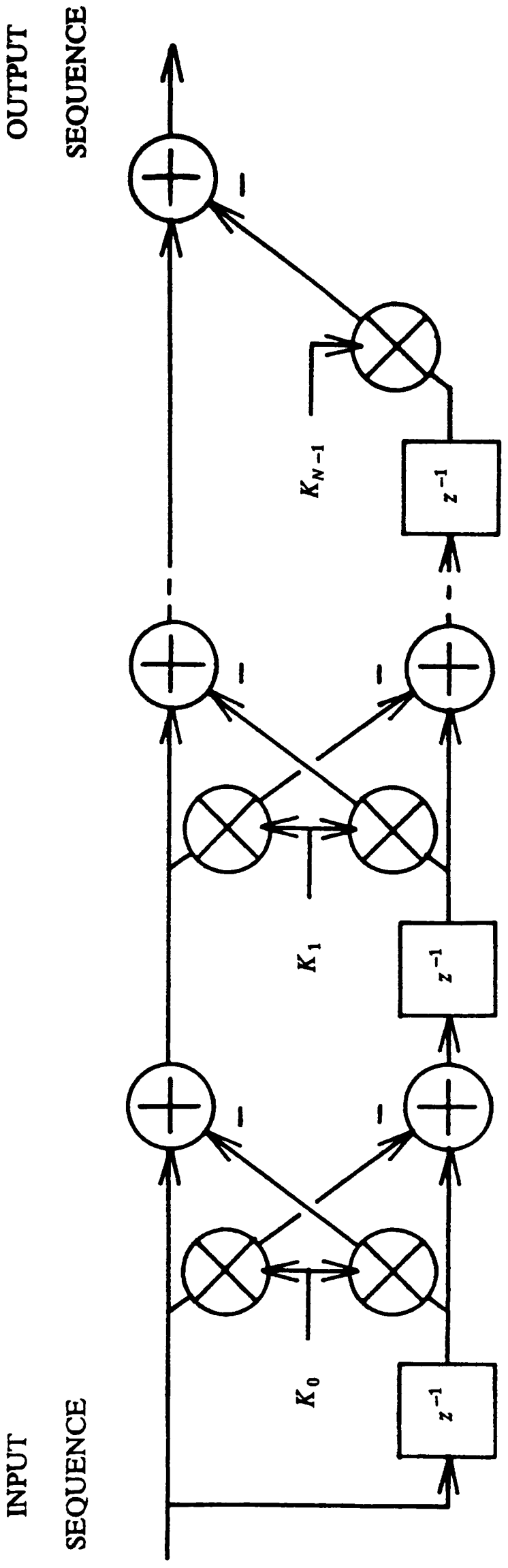
an order of magnitude more efficient computationally than equations (2.4.12)-(2.4.15). The fast Kalman algorithm for the pre-windowed form of LS is derived in Appendix A. The number of arithmetic operations per new data point (iteration) that are necessary for the fast Kalman algorithm is: 9N+1 additions/subtractions, 10N+1 multiplications, and 2 divisions. This is a significant increase in computational efficiency compared to the standard RLS algorithm, (2.4.12)-(2.4.14).

Since the publication of [25] , other fast least squares algorithm have appeared. Covariance forms of the algorithm were developed [42] . Computationally more efficient forms appeared such as: (i) the fast a posteriori error technique (FAEST) [43] , and (ii) the fast transversal filter (FTF) which is derived through the geometrical interpretation of least squares [44]. With the increased efficiency of this family of fast algorithms with respect to the standard RLS algorithm has come an increase in sensitivity to the effects of finite precision arithmetic e.g. [45]. In an attempt to alleviate these problems square root or normalised forms of the algorithm have been developed [44] , and periodic re-initialisation techniques suggested [46]. Finally, in an attempt to solve certain non-stationary problems, sliding window forms of the algorithm have been developed [47, 48].

The transversal filter structure illustrated in Figure 2.2. is not the only realisation of a FIR filter that is possible. A significant alternative is the lattice or ladder filter structure [49, 50], which exhibits several properties not found in a transversal filter realisation. These properties, which include modularity and good numerical round-off characteristics, are achieved at the expense of increased computational load for a given order of filter [51]. A lattice filter structure is illustrated in Figure 2.4. The major developments in fast RLS lattice structures have parallels with those in fast transversal filter form. Prewindowed and square root normalised algorithms first appeared in [52] and [53]. Because the square root normalised lattice algorithm has a natural interpretation as a set of rotations it can be realised efficiently using a coordinate rotation digital computer (CORDIC) [54]. Both normalised and unnormalised,

covariance and sliding window forms have also been developed [55]. The effects of fixed point implementation on the RLS lattice is examined in [56]. Recently alternative lattice structures have been developed for certain classes of nonstationary process [57].

Figure 2.4 A LATTICE FIR FILTER



2.4.4 Properties of the Least Squares Estimate

For wide sense stationary random processes ensemble averages may be replaced by time averages. Therefore the MSE cost function, ξ, may be replaced by an infinite summation.

$$\xi(n) = E[\, e^2(n)\,] = \lim_{k\to\infty} \frac{1}{k} \sum_{n=0}^{k} e^2(n) \tag{2.4.17}$$

The summation on the right hand side of (2.4.17) is recognisable as the LS cost function, (2.4.1). The scaling factor $\frac{1}{k}$ does not affect the LS solution given by (2.4.2). Thus for an infinite data set, the MSE cost function and the LS cost function are minimised by the same tap vector $\underline{h}_{opt}$. Hence it is concluded that the solution to the LS problem $\underline{h}(k)$ converges asymtotically to the Wiener solution $\underline{h}_{opt}$ [58, 59, 60]. However for finite data sequences analytic results for the properties of the LS estimate are more difficult to derive. The approximate analysis presented in [58] gives the following three useful results which are applicable to what is termed long sequences ie. $N \ll k < \infty$.

$$E[\, \underline{h}(k) - \underline{h}_{opt}\,] = 0 \tag{2.4.18}$$

$$E[\, (\, \underline{h}(k) - \underline{h}_{opt}\,)\,(\, \underline{h}(k) - \underline{h}_{opt}\,)^T\,] = \frac{\xi_{opt}}{k}\, \Phi_{xx}^{-1} \tag{2.4.19}$$

$$E[\, e^2(k)\,] = \xi_{opt} + \frac{N}{k}\, \xi_{opt} \tag{2.4.20}$$

These results suggest that: (i) for finite data sets the LS solution, $\underline{h}(k)$, is an unbiased estimate of the Wiener solution, $\underline{h}_{opt}$, (ii) the error covariance of the estimate, (2.4.19), whilst being a function of the signal autocorrelation Φ_{xx}, decreases with the size of the available data set, k, and (iii) the excess MSE, $\frac{N}{k}\, \xi_{opt}$, increases linearly with the number of coefficients, N, in the impulse response and decreases with the size

of the data set. Equation (2.4.18) to (2.4.20) concur with the results obtained in [40] for a channel equaliser application. Equation (2.4.20) is commonly used to conclude that the RLS algorithm converges within 2N iterations since for $k = 2N$,

$$10 \log_{10} \left(\frac{E[\, e^2(2N)\,]}{\xi_{opt}} \right) = 1.8\ dB. \tag{2.4.21}$$

The MSE is within 1.8 *dB* of the MMSE in 2N iterations of the RLS algorithm. This rule of thumb has to be used with caution as (2.4.21) is a relative measure, relative to the MMSE. Adequate performance with respect to the MMSE in one application might not be adequate performance with respect to the MMSE in another application.

By far the most rigorous analysis of the performance of the LS estimator appears in [24] and applies to the specific example of the system identification problem illustrated in Figure 2.3. when the input noise term is removed. If the input sequence, $\{ x(n) \}$, is considered to be a known deterministic process then the only random process is the noise sequence, $\{ \eta(n) \}$, and the LS estimate, $\underline{h}(k)\ k \geq N$, of the system impulse response, $\underline{h}_s$, is the best linear unbiased estimate (BLUE). The error covariance of the $\underline{h}(k)$ with respect to the noise sequence, $\{ \eta(n) \}$, is given by

$$E_n[\ (\ \underline{h}(k) - \underline{h}_s\ )\ (\ \underline{h}(k) - \underline{h}_s\ )^T\ ] = \underline{r}_{xx}^{-1}(k)\ \sigma_\eta.$$

Further if the random noise sequence is Gaussian then the LS estimate achieves the Cramer-Rao lower bound and is the minimum variance unbiased estimate (MVUE).

2.5 STOCHASTIC GRADIENT METHODS

While Gaussian elimination or the Levinson recursion provide the means for solving the Wiener equation, (2.2.8), in a fixed number of calculations, iterative techniques are also available where the number of calculations necessary to find the solution are not known beforehand [34]. The method of steepest decent is an iterative technique that has been used in linear programming and optimisation problems to find a variable, (eg. $\underline{h}_{opt}$), which minimises an objective or cost function (eg. ξ). To apply the method to the solution of (2.2.8) a guess at the solution, $\underline{h}_i$, is made. The subscript i is used to denote the ith step in an iterative process rather than a time recursion such as (2.4.12). This guess will have associated with it a particular value of the MSE cost function, $\xi(\underline{h}_i)$, given by (2.2.4). This guess is then improved by a two stage process. First the gradient vector , $\underline{\nabla}_i$, associated with the guess is calculated using (2.2.7). The gradient vector is in the direction of the greatest rate of change of the MSE cost function. Second a scaled version of the gradient, $\mu\underline{\nabla}(\underline{h}_i)$, is subtracted from the guess to form a new guess $\underline{h}_{i+1}$. The MSE at the new guess will be smaller than the MSE at the initial guess if the small positive scalar μ is chosen correctly. Hence if the two stages are repeated the MSE associated will be reduced until it reaches the minimum, ξ_{opt}, at which point the guess $\underline{h}_i$ will equal $\underline{h}_{opt}$. The method of steepest descent is defined by the following two equations.

$$\underline{h}_{i+1} = \underline{h}_i - \mu\, \underline{\nabla}_i \tag{2.5.1}$$

$$\underline{\nabla}_i = \frac{\partial \xi}{\partial \underline{h}} (\underline{h}_i)$$

$$= 2\, \underline{\Phi}_{xx}\, \underline{h}_i - 2\, \underline{\Phi}_{xy} \tag{2.5.2}$$

The method will converge to the optimum solution provided the step size or convergence factor, μ, lies within the range

$$0 < \mu < \frac{1}{\lambda_{max}}. \tag{2.5.3}$$

where λ_{max} is the largest eigenvalue of the autocorrelation matrix, Φ_{xx} [21]. However the importance of the method of steepest descent in adaptive filtering is not as an iterative alternative to direct methods such as Gaussian elimination or the Levinson recursion, rather it is that it is the basis of the least mean squares (LMS) [21] and the block least mean squares (BLMS) [61] stochastic gradient algorithms.

2.5.1 The Least Mean Squares Algorithm

While application of the method of steepest descent avoids the direct matrix inversion inherent in the Wiener equation, (2.2.9), explicit knowledge of the autocorrelation matrix, Φ_{xx}, and the cross-correlation vector, Φ_{xy}, is still required, (2.5.2). The requirement for knowledge of these statistics is circumvented in the LMS stochastic gradient algorithm by replacing the iterative step, (2.5.1), with a time recursion and the gradient with an estimate of the gradient.

$$\underline{h}(k+1) = \underline{h}(k) - \mu\, \underline{\hat{\nabla}}(k) \tag{2.5.4}$$

The vector $\underline{h}(k)$ is an estimate of the Wiener filter, $\underline{h}_{opt}$, at time sample k and the vector $\underline{\hat{\nabla}}(k)$ is an estimate of the gradient, $\underline{\nabla}(k)$, of the MSE cost function at the point where the impulse response is $\underline{h}(k)$. The exact gradient, $\underline{\nabla}(k)$, is defined in similar manner to $\underline{\nabla}_i$, (2.5.2).

$$\underline{\nabla}(k) = \frac{\partial \xi}{\partial \underline{h}} (\underline{h}(k))$$

$$= \frac{\partial}{\partial \underline{h}} E[(y(n) - \underline{h}^T(k)\, \underline{x}(n))^2] \tag{2.5.5}$$

After some rearrangment (2.5.5) yields

$$\underline{\nabla}(k) = -2\, E[\underline{x}(n) (y(n) - \underline{h}^T(k)\, \underline{x}(n))] . \tag{2.5.6}$$

To form an estimate, $\hat{\underline{\nabla}}(k)$, of the gradient, $\underline{\nabla}(k)$, the ensemble average of (2.5.6) could be replaced by a time average over n. However since $\underline{h}(k)$ changes at every data point, (2.5.4), the time average reduces to a single value at $n = k+1$.

$$\hat{\underline{\nabla}}(k) = -2\,\underline{x}(k+1)\,e(k+1) \tag{2.5.7}$$

$$e(k+1) = y(k+1) - \underline{h}^T(k)\,\underline{x}(k+1) \tag{2.5.8}$$

This gradient estimate is unbiased [21] . Substitution of (2.5.7) into (2.5.4) yields the LMS algorithm.

$$\underline{h}(k+1) = \underline{h}(k) + 2\,\mu\,\underline{x}(k+1)\,e(k+1) \tag{2.5.9}$$

Although the LMS algorithm is considerably simpler than the RLS algorithm of subsection 2.4.1 its convergence properties are nonetheless difficult to analysis rigorously. The simplest approach is to examine the evolution of the mean of the estimated impulse response vector. Taking expected values of both sides of (2.5.9) gives

$$E[\,\underline{h}(k+1)\,] = E[\,(\,I - 2\,\mu\,\underline{x}(k+1)\,\underline{x}^T(k+1)\,)\,\underline{h}(k)\,]$$

$$+ 2\,\mu\,E[\,\underline{x}(k+1)\,y(k+1)\,]\,.$$

Evaluation of the first expected value on the right hand side is difficult because of the dependence of the vectors $\underline{x}(k+1)$ and $\underline{h}(k)$. If however they are assumed to be independent, the algebra becomes more tractable and a simple recursion for $E[\,\underline{h}(k)\,]$ is obtained.

$$E[\,\underline{h}(k+1)\,] = (\,I - 2\,\mu\,\Phi_{xx}\,)\,E[\,\underline{h}(k)\,] + 2\,\mu\,\Phi_{xy} \tag{2.5.10}$$

Although the independence assumption is in a strict sense invalid it does yield results that agree well with experiment [26]. From (2.5.10) bounds on the step size, μ, that ensure convergence in the mean of $\underline{h}(k)$ to $\underline{h}_{opt}$ can be obtained.

Rewriting (2.5.10) in terms of an error vector

$$\tilde{\underline{h}}(k) = \underline{h}(k) - \underline{h}_{opt}$$

gives

$$E[\,\tilde{\underline{h}}(k+1)\,] = (\,I - 2\,\mu\,\Phi_{xx}\,)\,E[\,\tilde{\underline{h}}\,]\,. \tag{2.5.11}$$

If $E[\,\tilde{\underline{h}}(k)\,]$ converges to the zero vector as $k \to \infty$, then $E[\,\underline{h}(k)\,]$ converges to $\underline{h}_{opt}$. Assuming that the autocorrelation matrix, Φ_{xx}, is symmetric and positive definite it can be factorised as

$$\Phi_{xx} = V\,\Lambda\,V^T\,, \tag{2.5.12}$$

where Λ is the diagonal matrix of eigenvalues

$$\Lambda = diag\,[\,\lambda_0\,\lambda_1\,\cdots\,\lambda_{N-1}\,]$$

and V is the orthonormal matrix whose jth column is the eigenvector of Φ_{xx} associated with the jth eigenvalue. Substitution of (2.5.12) into (2.5.11) effectively transforms the error vector $\underline{h}(k)$ to a vector $\underline{H}(k)$ whose components, $H_j(k)$, evolve in time independently of each other. Thus

$$E[\,\tilde{\underline{H}}(k+1)\,] = (\,I - 2\,\mu\,\Lambda\,)\,E[\,\tilde{\underline{H}}(k)\,]$$

where

$$\tilde{\underline{H}}(k) = V^T\,\tilde{\underline{h}}(k)\,.$$

Since V^T is unitary the vectors $\tilde{\underline{H}}(k)$ and $\tilde{\underline{h}}(k)$ have the same Euclidean norm , and hence convergence of $\tilde{\underline{H}}(k)$ to the zero vector is equivalent to convergence of $\tilde{\underline{h}}(k)$ to the zero vector. Further since the matrix $(\,I - 2\,\mu\,\Lambda\,)$ is diagonal, a separate recursion for each element, $\tilde{H}_j(k)$, of $\tilde{\underline{H}}(k)$ can be obtained.

$$E[\,\tilde{H}_j(k+1)\,] = (\,1 - 2\mu\lambda_j\,)\,E[\,\tilde{H}_j(k)\,], \quad j = 0, 1, \ldots, N-1$$

(2.5.14)

Each of these modes decays exponentially to zero provided

$$| 1 - 2\mu\lambda_j | < 1 .$$

Convergence of all the elements of $\tilde{\underline{U}}(k)$ to zero is assured if

$$o < \mu < \frac{1}{\lambda_{max}} , \tag{2.5.15}$$

which is identical to the condition for convergence of the method of steepest descent, (2.5.3). The time constant, τ_j associated with equation (2.5.14) is given approximately by

$$\tau_j = \frac{1}{2\mu\lambda_j} .$$

Hence the longest time constant, τ_{max}, is associated with the smallest eigenvalue, λ_{min}.

$$\tau_{max} = \frac{1}{2\mu\lambda_{min}} \tag{2.5.16}$$

Combining (2.5.15) with (2.5.16) gives,

$$\tau_{max} > \frac{\lambda_{max}}{2\lambda_{min}} \tag{2.5.17}$$

which suggests that the larger the eigenvalue ratio (condition number), $\frac{\lambda_{max}}{\lambda_{min}}$, of Φ_{xx} the longer the LMS algorithm will take to converge.

Analyses of the MSE convergence properties of the LMS algorithm are available in [27] and [26]. Again the independence of the vectors $\underline{x}(k+1)$ and $\underline{h}(k)$ is assumed in order to make the algebra tractable. In both [27] and [26], the bounds on the step size, μ, that ensure MSE convergence of the LMS algorithm are more restrictive than those of (2.5.15). For example in [27] the bounds on the step size are

$$0 < \mu < \frac{1}{3\,N\,E[\,x^2(n)\,]} . \tag{2.5.18}$$

These analysis also indicate that the rate of convergence of the MSE is dependent on the eigenvalues of Φ_{xx} but in a more complex way than that suggested by (2.5.17).

2.5.2 The Block Least Mean Squares Algorithm

If the estimated impulse response, $\underline{h}(k)$, is held constant for a block of L data points, the recursion of (2.5.4) becomes

$$\underline{h}(k+L) = \underline{h}(k) - \mu_b \, \hat{\underline{\nabla}}(k) \, . \tag{2.5.19}$$

The gradient estimate of (2.5.7) can then be improved by replacing the ensemble average of (2.5.6) with a time average over the L values of the time index n for which the estimated impulse response is constant.

$$\hat{\underline{\nabla}}(k) = - \frac{2}{L} \sum_{n=k+1}^{k+L} \underline{x}(n) \, (\, y(n) - \underline{h}^T(k) \, \underline{x}(n) \,)$$

This can be written more succinctly as

$$\hat{\underline{\nabla}}(k) = - \frac{2}{L} \, \underline{\chi}(k+L) \, \underline{e}(k+L) \tag{2.5.20}$$

where $\underline{\chi}(k+L)$ is an $(N \times L)$ matrix constructed from L input vectors $\underline{x}(n)$, $n = k+1, k+2. \cdots k+L$

$$\underline{\chi}(k+L) = [\, \underline{x}(k+L) \, \underline{x}(k+L-1) \, \cdots \, \underline{x}(k+1) \,]$$

and the $(L \times 1)$ error vector, $\underline{e}(k+L)$, is the difference between the signal vector

$$\underline{y}(k+L) = [\, y(k+L) \, y(k+L-1) \, \cdots \, y(k+1) \,]^T$$

and the estimated signal vector

$$\underline{\hat{y}}(k+L) = \underline{\chi}^T(k+L) \, \underline{h}(k) \tag{2.5.21}$$

that is,

$$\underline{e}(k+L) = \underline{y}(k+L) - \underline{\hat{y}}(k+L). \tag{2.5.22}$$

Together (2.5.19) to (2.5.22) define the BLMS adaptive filter algorithm [14].

Equations (2.5.20) and (2.5.21) represent linear convolution operations. For example, the gradient estimate of (2.5.20) contains N scaled outputs at

$$n = k+L-N+1, k+L-N+2, \cdots k+L-1, k+L$$

of a FIR filter with impulse response vector $\underline{e}(k+L)$. Similarly the estimated signal vector of (2.5.21) contains L outputs at

$$n = k+1, k+2, \cdots K+L-1, k+L$$

of a FIR filter with impulse response vector $\underline{h}(k)$. The input sequence $\{x(n)\}$ for $n = k-N+2 \cdots k \cdots k+L$ is applied to both filters. These linear convolution operations can be implemented using a combination of a circular convolution algorithm such as the fast Fourier transform (FFT) [7], the rectangular transform (RT) [62], or the number theoretic transform (NTT) [63]. and the overlap add or overlap save data sectioning technique [7]. The relationship between circular and linear convolution and the use of the method of overlap save are described in Appendix B. This implementation results in a substantial computational saving on the direct time domain approach and is the major motivation behind the use of the BLMS algorithm. Although it is possible to use any block length, L, the most efficient adaptive filter structures are obtained when the block length is equal to the filter length, N [61]. As this condition also simplifies the algebra it is the only one that is considered here.

In order to calculate N outputs of a N-point FIR filter a 2N-point circular convolution is required [7]. Such a circular convolution can be performed using a transform domain processor defined in general by two $(M \times 2N)$ matrices A_{2N} and B_{2N} and a $(2N \times M)$ matrix C_{2N}, where $M \geq 2N$. Applying the techniques described in Appendix B to (2.5.21), for $L = N$

$$\underline{\hat{y}}(k+N) = \underline{X}(k+N)\,\underline{h}(k)$$

$$= [\, O_N \;\; T_N \,]\, C_{2N} \{ \underline{\mathcal{X}}(k+N) \times \underline{H}(k) \} \qquad (2.5.23)$$

where

$$\underline{\mathcal{X}}(k+N) = B_{2N} \begin{bmatrix} \underline{x}(k) \\ \underline{x}(k+N) \end{bmatrix}$$

and

$$\underline{H}(k) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}(k)\,.$$

The symbol $\times$ represents the point by point multiplication of the vectors $\underline{\mathcal{X}}(k+N)$ and $\underline{H}(k)$. The matrix I_N is the $(N \times N)$ identity matrix and the $(N \times N)$ matrix O_N has all zero elements. The reversal matrix T_N has 1's on the secondary diagonal and zero elsewhere. The linear convolution of (2.5.20) can be performed in a similar manner,

$$\underline{\hat{\nabla}}(k) = -\frac{2}{N}\,\underline{X}(k+N)\,\underline{e}(k+N)$$

$$= -\frac{2}{N}\,[\, O_N \;\; T_N \,]\, C_{2N} \{ \underline{\mathcal{X}}(k+N) \times \underline{E}(K+N) \} \qquad (2.5.24)$$

where

$$\underline{E}(k+N) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{e}(k+N)\,.$$

Since $\underline{\mathcal{X}}(k+N)$ appears in both (2.5.23) and (2.5.24) it is only necessary to calculate it once per block of data. The complete circular convolution implementation of the BLMS algorithm is summarised in Table 2.1.

Using a similar analysis to that presented in section 2.5.1 for the LMS algorithm, convergence in the mean of the BLMS algorithm is assured provided

$$0 < \mu_b < \frac{1}{\lambda_{max}} \qquad (2.5.25)$$

and the time constant, τ_j, associated with the jth mode is given approximately by

$$\tau_j = \frac{1}{2\,\mu\,\lambda_j} \quad blocks$$

However for the BLMS algorithm the time constant is measured in blocks of L data points and must be multiplied by L to give an equivalent time constant in samples in order to facilitate comparisons with the LMS algorithm.

$$\tau_j = \frac{L}{2\,\mu\,\lambda_j} \quad samples$$

In common with the LMS algorithm the largest time constant τ_{max}, is associated with the smallest eigenvalue, λ_{min}.

$$\tau_{max} = \frac{L}{2\,\mu\,\lambda_{min}} \quad samples \tag{2.5.26}$$

Comparing (2.5.26) and (2.5.15) for a given value of the step size, the largest time constant associated with the BLMS algorithm is L times larger then the largest time constant associated with the LMS algorithm. In order to achieve the same rate of convergence as the LMS algorithm the value of μ_b must be made L times greater than μ in which case there is greater danger of the stability condition of (2.5.25) being infringed. If (2.5.26) and (2.5.24) are combined the dependence of the rate of convergence of the BLMS algorithm on the eigenvalue ratio is illustrated.

$$\tau_{max} > \frac{L\,\lambda_{max}}{2\,\lambda_{min}} \tag{2.5.27}$$

The MSE analysis of the BLMS algorithm presented in [28] is an extension of the convergence analysis of the LMS algorithm which appears in [27] and also indicates that the rate of convergence of the BLMS algorithm decreases with increasing block length L. However the relationship is not as simple as (2.5.27). The MSE analysis also reveals that the bounds on μ_b that ensure convergence are more restrictive than (2.5.25).

$$0 < \mu < \frac{L}{(L+2)\, N\, E[\, x^2(n)\,]} \tag{2.5.28}$$

This condition simplifies to (2.5.18) when $L = 1$. In conclusion the computational advantages afforded by the BLMS algorithm when compared with the LMS algorithm are obtained at the expense of degraded convergence performance.

Table 2.1 THE BLMS ADAPTIVE FILTER

$$\underline{X}(j) = B_{2N} \begin{bmatrix} T_N\ \underline{x}(j-1) \\ T_N\ \underline{x}(j) \end{bmatrix}$$

$$\underline{H}(j-1) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}(j-1)$$

$$\underline{\hat{y}}(j) = [\ O_N\ T_N\ ]\ C_{2N} \left\{ \underline{X}(j) \times \underline{H}(j-1) \right\}$$

$$\underline{\varepsilon}(j) = \underline{y}(j) - \underline{\hat{y}}(j)$$

$$\underline{E}(j) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{\varepsilon}(j)$$

$$\underline{\varepsilon}(j) = [\ O_N\ T_N\ ]\ C_{2N} \left\{ \underline{X}(j) \times \underline{E}(j) \right\}$$

$$\underline{h}(j) = \underline{h}(j-1) + \frac{2\mu_b}{N} \underline{\varepsilon}(j)$$

2.6 TRANSFORM DOMAIN ALGORITHMS

The convergence analyses of the LMS algorithm (either in a mean or a MSE sense) strongly indicate that the rate of convergence of the algorithm is dependent on the spread of the eigenvalues of the autocorrelation matrix, Φ_{xx}. This dependency can be removed by first applying a linear transformation defined in terms of a $(N \times N)$ matrix P, to the input vector, $\underline{x}(n)$, to form a second vector, $\underline{z}(n)$,

$$\underline{z}(n) = P\ \underline{x}(n) \tag{2.6.1}$$

such that

$$\Phi_{zz} = E[\ \underline{z}(n)\ \underline{z}^T(n)\ ] = I_N\ . \tag{2.6.2}$$

If $\underline{z}(n)$ is then used as the input vector to the LMS algorithm instead of $\underline{x}(n)$, the convergence performance of the algorithm will be improved since the eigenvalues of Φ_{zz} are all unity. The matrix P is obtained from the square root factorisation of the autocorrelation matrix. If Φ_{xx} is a positive definite matrix then a matrix P exists such that

$$P\ \Phi_{xx}\ P^T = I_N \tag{2.6.3}$$

and hence

$$\Phi_{xx} = P^{-1}(\ P^{-1}\ )^T\ . \tag{2.6.4}$$

The factorisation of (2.6.4) is not unique [39] and thus for a given autocorrelation matrix there are many transforms such as P which exhibit the property summarised in (2.6.1) and (2.6.2).

The linear transformation of (2.6.1) reduces the symmetric positive definite matrix, Φ_{xx}, to the identity matrix, Φ_{zz}. This whitening of the signal can be performed in two stages. First a linear transformation is applied to the input vector, $\underline{x}(n)$, to produce an intermediate vector whose autocorrelation matrix is diagonal ie.

the input vector is orthogonalised. Then each element of the intermediate vector is scaled to produce a vector, $\underline{z}(n)$, whose autocorrelation matrix is the identity. The second stage is known as power normalisation. The two stage process is dependent on the existence of factorisations of the form,

$$\Phi_{xx} = W\ D\ W^T \tag{2.6.5}$$

where D is an $(N \times N)$ diagonal matrix. The transform of the input vector, $\underline{x}(n)$, to the vector $\underline{z}(n)$ is defined as the product of two matrices W^{-1} and $D^{-1/2}$.

$$\underline{z}(n) = D^{-1/2}\ W^{-1}\ \underline{x}(n)$$

The $(N \times N)$ diagonal matrix $D^{-1/2}$ contains the reciprocals of the positive square roots of the elements on the diagonal of D. For a symmetric positive matrix two factorisations of this form are possible: Cholesky factorisation and eigenvalue decomposition. For the latter, which is described in section 2.5.1, the transform matrix, W^{-1}, is the Karhunen-Loeve transform (KLT). For the Cholesky factorisation the $(N \times N)$ matrix W is a lower or upper diagonal with a unit leading diagonal. The transformation matrix, W^{-1}, is of the same form. In fact the lower (upper) diagonal matrix, W^{-1}, contains the coefficients of all backward (forward) prediction error filters from order 0 to order N-1 [64, 35] and hence multiplication of the input vector $\underline{x}(n)$ by W^{-1} can be implemented efficiently using a lattice filter structure [35].

The forward prediction error of order j , $e_j^f(n)$, is the output of a FIR filter with j+1 taps.

$$e_j^f(n) = x(n) + \sum_{i=1}^{j} a_{ij}\ x(n-i)$$

The forward prediction coefficients, $a_{ij}\quad i = 1, \cdots j$, are chosen to minimise the MSE cost function.

$$E[\ (\ e_j^f(n)\ )^2\ ]$$

Similarly the jth order backward prediction error, $e_j^b(n)$, is the output of a backward prediction filter with j+1 taps.

$$e_j^b(n) = x(n-N) + \sum_{i=1}^{j} b_{ij}\, x(n-i+1)$$

The coefficents, b_{ij}, are chosen to minimise a MSE cost function.

$$E[\,(\,e_j^b(n)\,)^2\,]$$

All forward and backward prediction errors from order 0 to order N-1 can be generated using the following equations which define one stage of a lattice filter.

$$e_j^f(n) = e_{j-1}^f(n) + K_j\, e_{j-1}^b(n-1)$$

$$e_j^b(n) = e_{j-1}^b(n-1) + K_j\, e_{j-1}^f(n)$$

The partial correlation (PARCOR) coefficients, K_j, can be calculated using the Levinson recursion [22].

The two factorisations summarised in (2.6.5) cannot be used to construct an adaptive filter algorithm since explicit knowledge of the autocorrelation matrix, Φ_{xx}, is required before any factorisation can be performed. However approximate factorisations of this form lead to adaptive filter algorithms whose convergence properties are less sensitive to the eigenvalue spread of the autocorrelation matrix than the LMS algorithm. The two major classes of transform domain algorithm are based on the two possible factorisation of the form given in (2.6.5). Since the Cholesky factorisation can be implemented using a lattice filter structure, the transformation matrix, W^{-1}, can be estimated from the available data by estimating the PARCOR coefficients that define the lattice filter. The simplest such algorithm is the stochastic gradient (SG) adaptive lattice filter of [65], where the PARCOR coefficients are estimated using the forward and backward prediction error and a constant μ_{lat}.

$$K_j(k+1) = K_j(k) + \mu_{lat}\,(\,e_j^f(k+1)\,e_{j-1}^b(k) + e_{j-1}^f(k+1)\,e_j^b(k+1)\,)$$

The convergence properties of the SG adaptive lattice filter are analysed in [66]. Alternatively the eigenvalue decomposition leads the signal dependent KLT which can be approximated using non signal dependent unitary transforms such as the discrete Fourier transform (DFT) [67], the discrete cosine transform (DCT) [29] and the Walsh transform (WT) [68]. To illustrate a transform domain adaptive filter the DFT based structure is examined in more detail.

2.6.1 The Sliding DFT Adaptive Filter

In the DFT based structure of [67] the input vector, $\underline{x}(n)$, is transformed to a vector, $\underline{X}(n)$ using the $(N \times N)$ complex DFT matrix F_N.

$$\underline{X}(n) = [\, X_0 \, X_1 \, \cdots \, X_{N-1} \,]^T$$

$$= F_N \, \underline{x}(n) \qquad (2.6.6)$$

The element in row l column m of the DFT matrix is a complex exponential term.

$$F_N[l,m] = exp\left(- \sqrt{(-1)} \, \frac{2\pi lm}{N} \right)$$

The elements of the transformed vector, $\underline{X}(n)$, are weighted and summed to produce an estimate, $\hat{y}(n)$, of $y(n)$.

$$\hat{y}(n) = \underline{H}^T \, \underline{X}(n)$$

$$\underline{H} = [\, H_0 \, H_1 \, \cdots \, H_{N-1} \,]^T$$

The value of the vector $\underline{H}$ which minimises the MSE cost function, ξ (), is $\underline{H}_{opt}$.

Since the matrix F_N is complex the vector $\underline{X}(n)$ will in general be complex even if $\underline{x}$ is real and hence the complex LMS algorithm [69] must be used to estimate the optimum vector $\underline{H}_{opt}$ which minimises the MSE.

$$\underline{H}(k+1) = \underline{H}(k) + 2\,\mu_c\, D^{-1}\, \underline{X}^*(k+1)\, e(k+1) \qquad (2.6.7)$$

$$e(k+1) = y(k+1) - \hat{y}(k+1)$$

The superscript * denotes complex conjugate. The real positive constant, μ_c, is the convergence factor or step size. The autocorrelation matrix, $E[\,\underline{X}(n)\,(\,\underline{X}^*(n)\,)^T\,]$, is assumed to be diagonal.

$$D = diag\,[\,D_0\, D_1\, \cdots\, D_{N-1}\,]$$

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

The elements of the diagonal matrix D are estimated from the data using a recursion of the form,

$$D_i(k+1) = \alpha\, D_i(k) + (\,1 - \alpha^2\,)\, X_i(k+1)\, X_i^*(k+1)$$

where α is a positive constant less than unity. The use of D^{-1} in (2.6.7) to control the adaptive step size, $\frac{\mu}{D_i(k+1)}$, is equivalent to power normalisation and avoids the square root operation associated with $D^{-1/2}$. An efficient structure for implementing (2.6.6) is obtained by using the following recursion [7].

$$X_i(k+1) = exp(\,-\sqrt{(-1)}\,\frac{2\pi i}{N}\,)\, X_i(k) + x(k+1) - x(k-N)\,.$$

2.7 ALGORITHM SUMMARY AND COMPLEXITY COMPARISON

Table 2.2 shows a summary of the computational complexity of a selection of time recursive adaptive FIR filter algorithms. The algorithms are divided into three types; recursive least squares, stochastic gradient and transform domain or quasi-orthogonalising algorithms. Two RLS algorithms are considered; the original fast Kalman algorithm of [25] which is derived in Appendix A, and the lattice filter algorithm of [52]. Both algorithms are prewindowed. A more comprehensive comparison of the computational complexity of the various RLS implementation options is given in [16]. Two stochastic gradient algorithms are considered; the LMS algorithm of [21] described in subsection 2.5.1 and the FFT-based BLMS of [14] described in subsection 2.5.2. Finally the sliding transform algorithm of subsection 2.6.1 is taken as an example of a transform domain or quasi-orthogonalising algorithm. The computational load of each algorithm is assessed in terms of the three major arithmetic operations ie. divisions, multiplication and additions/subtractions. The figures quoted are numbers of operations to process each new pair of data points data point $x(k+1)$ and $y(k+1)$.

The stochastic gradient algorithms are the least demanding computationally of all the adaptive FIR filter algorithms with the BLMS requiring even less computations than the LMS for $N \geq 64$. Unfortunately they exhibit the poorest convergence performance since, as indicated in section 2.5, the convergence rate is dependent on the eigenvalues of the autocorrelation matrix associated with the input signal. The RLS algorithms, on the other hand, exhibit consistent fast convergence properties, as indicated by (2.4.19), but are the most expensive computationally. Finally the transform domain or quasi-orthogonalising algorithms are less sensitive to the eigenvalue spread of the input autocorrelation matrix than the stochastic gradient algorithms. Thus they offer convergence performance that lies between the RLS and SG algorithms. However the sliding DFT algorithm is closer to the RLS algorithms

than the SG algorithms in computational load.

The convergence characteristics of stochastic gradient, recursive least squares and transform domain algorithms are examined further in chapter 3 through the medium of computer simulation.

Table 2.2 COMPLEXITY COMPARISON
(Adaptive FIR Filter Algorithms)

Algorithm	Implementation	Computational Load		
		mult	add/sub	div
Recursive Least Squares	fast Kalman	10N+1	9N+1	2
	lattice	8N	8N	6N
Stochastic Gradient	LMS	2N	2N	-
	BLMS (FFT)	10log(N)+8	15log(N)+30	-
Transform Domain	Sliding DFT	8N+16	6N+9	N

Chapter 3

PERFORMANCE COMPARISONS

3.1 INTRODUCTION

While logistics preclude a comparison of the complete set of algorithms that have been mentioned in chapter 2, it is possible to examine the performance of a subset whose elements are representative of the three classes into which adaptive filters may be divided. These three classes are: (i) stochastic gradient search algorithms such as the LMS algorithm of subsection 2.5.1 and the BLMS algorithm of subsection 2.5.2, (ii) quasi-orthogonalising or transform domain algorithms such as the sliding DFT structure of subsection 2.6.1, and (iii) least squares techniques such as the simple RLS algorithm of subsection 2.4.1. The convergence performance of one or two algorithms from each class will be studied by computer simulation.

In the literature on adaptive filters, the LMS is taken as the benchmark against which all other algorithms are compared. This is appropriate because of the relative simplicity and hence popularity of this algorithm. Conventional analysis of the LMS [21, 26] indicates that the time to converge is a function of the eigenvalues of the autocorrelation matrix Φ_{xx}. The greater the spread of the eigenvalues, the more coloured or ill-conditioned the input signal is and the longer the algorithm takes to converge. The best performance that can be achieved with an LMS algorithm occurs when all the eigenvalues of the autocorrelation matrix are equal and hence the signal is white. The analysis of the RLS algorithm is more complex than that of the LMS [24, 70, 58]. However it is apparent that the convergence rate of the RLS is insensitive to eigenvalue spread and hence the RLS will in general converge faster than the LMS algorithm. In fact it will outperform the LMS even under white input conditions [71]. There are thus three performance goals at which an adaptive FIR filter algorithm may be aimed. These are: (i) LMS performance, (ii) LMS performance under white input

conditions, and (iii) least squares performance. The goals (i), (ii), and (iii) coincide with the three classes of algorithm already enumerated.

In comparing the performance of the various algorithms two specific applications are considered. In section 3.2, system identification, the adaptive filter is used to estimate the impulse response of an unknown system. In section 3.3, channel equalisation, the adaptive filter is used to reduce the effects of intersymbol interference on a digital communications channel. Although these two examples do not cover the complete spectrum of adaptive filter applications, they represent the two major modes of operation ie. direct and inverse system modelling. Also considered are the effects of adding a white noise term first to the training input and second to the signal input of the adaptive filter.

3.2 SYSTEM IDENTIFICATION

The architecture that was employed for the simulation of a system identification problem is illustrated in Figure 3.1. A coloured input sequence $\{ x(n) \}$, formed by passing a zero mean white Gaussian sequence $\{ z(n) \}$ through a 3-tap FIR channel with impulse response $\{ c_n \}$, was applied both to the input of the unknown system and to the input of the adaptive filter. The system output, corrupted by an additive white Gaussian noise term, $\eta(n)$, was subtracted from the adaptive filter output, y(n), to form the error term, e(n). The number of coefficients in the adaptive filter was 16, corresponding exactly with the number of coefficients in the fixed FIR filter which was used to simulate the unknown system. A selection of three channel impulse responses was used in order to vary the degree of ill-conditioning on the sequence $\{ x(n) \}$. The impulse responses are given in Table 3.1 along with the associated condition number or eigenvalue ratio [29] of the (16 × 16) autocorrelation matrix Φ_{xx}. All three channels and the unknown system were chosen to have unit gain eg. $\sum_n c_n^2 = 1$. Thus with the

variance of the white sequence $\{ z(n) \}$ at unity, both the input and output sequences of the unknown system had unit variance. This gave a convenient reference for the variance of the additive noise and for calculating the step size of the stochastic gradient algorithms.

The scenario illustrated in Figure 3.1 is essentially the same as that of subsection 2.2.2 with the exception that the input additive noise term has been removed. If a unique solution to (2.2.8) exists, ie. if Φ_{xx} has a unique inverse, then the optimum solution $\underline{h}_{opt}$ is equal to the impulse response of the unknown system. Multiple solutions may exist when the power spectral density of the input sequence $\{ x(n) \}$ has nulls. For all the examples considered here the input power spectral density is broadband and hence a unique solution will always exist. In a more general system identification scenario, as discussed in section 2.2.2, where another white noise term is added to the input of the adaptive filter alone, the Wiener solution will not equal the impulse response of the unknown system and hence the estimate formed by the adaptive filter will at best be biased.

The performance measure which is most appropriate to the system identification problem considered here is the norm

$$\rho(k) = E[(\underline{h}(k) - \underline{h}_{opt})^T (\underline{h}(k) - \underline{h}_{opt})] ,$$

where $\underline{h}(k)$ is the impulse response of the adaptive filter at iteration k, and $\underline{h}_{opt}$ is the impulse response of the unknown system. This was estimated in all the simulations by averaging over an ensemble of 25 runs of the adaptive filter. In all cases the adaptive filter was initialised with an impulse response vector of all zero elements. The variance of the noise sequence $\{ \eta(n) \}$ was set at -70dB with respect to unity.

The performance of five adaptive filter algorithms in the system identification scenario of Figure 3.1 is illustrated in the graphs of Figure 3.2. The traces labelled "1" and "2" are for the LMS algorithm defined in subsection 2.5.1 and the BLMS algorithm defined in subsection 2.5.2 respectively. The trace labelled "3" is for the SM

approach of section 2.3 using a autocorrelation estimate defined by (2.3.5). Only one example of the SM algorithm is provided because of the poor performance of the algorithm [31] and because no recursive form exists. The trace labelled "4" is for the RLS algorithm defined in subsection 2.4.1. Finally the trace labelled "5" is for the sliding DFT adaptive filter of subsection 2.6.1.

In a broad comparison of this nature, the selection of the step size or convergence factor μ for the stochastic gradient algorithms is not straightforward. The results of [26] indicate that bounds on μ necessary to ensure convergence in the mean of $\underline{h}(k)$ of the LMS algorithm are not restrictive enough to ensure convergence in a MSE sense. Recently [72] and [27] suggest an even more restrictive bound on μ with similar results being presented in [28] for the BLMS algorithm. Selection of the optimum value of μ that will produce the fastest convergence is more difficult still as it depends on having an exact knowledge of the eigenvalues of the autocorrelation matrix Φ_{xx} [27]. In order to make the analysis tractable, the results presented in [21, 26, 72, 27], and [28] makes assumptions that are more valid for a narrowband adaptive array applications than for transversal filter applications. Hence these theoretical results cannot be expected to predict the behaviour of a stochastic gradient adaptive filter exactly but they do explain general trends and are aids in the selection of μ.

For the purposes of this comparison a pragmatic approach was adopted. The region of convergence [72, 27]

$$0 < \mu \leq \frac{1}{3\,N\,E[\,x^2(n)\,]}$$

was chosen because of its practical advantages and because it is the most conservative available in the literature. The value of μ for the LMS algorithm was then chosen arbitrarily to be at the midpoint of this region.

$$\mu = \frac{1}{6NE[\,x^2(n)\,]}$$

$$= \frac{1}{6N} \tag{3.2.1}$$

The step size for the BLMS algorithm, μ_b, was set to N times this value ie.

$$\mu_b = N\ \mu = \frac{1}{6}$$

in an attempt to achieve the same convergence rate of the tap vector mean as the LMS algorithm [14]. However in the light of the bounds on μ_b, [28], that ensure MSE convergence of the BLMS algorithm ie.

$$0 < \mu \leq \frac{1}{(N+2)\ E[\ x^2(n)\ ]},$$

divergence of the adaptive filter might be expected under some input conditions. For the sliding DFT adaptive filter, the use of the adaptive step size, $\frac{\mu_c}{D_i(k+1)}$, is equivalent to normalising the variance of each of the DFT bins to unity [Narayan et al 83]. Thus the variance at each input to the linear combiner is unity which implies that a consistent choice of μ_c for this algorithm is also $\frac{1}{6N}$.

Under white input conditions the system identification problem of Figure 3.1 is similar to data driven echo canceller problem discussed in [71] and the results obtained for the LMS and RLS algorithms, Figure 3.2(a), are consistent with those obtained in [71]. The performance of the LMS algorithm under these conditions is predicted well by the theoretical results of [26] and [73]. These results indicate that the LMS algorithm will converge at a constant rate in dB/iteration until the noise floor is reached, in this case -70.0 dB. The rate is determined by the value of μ which is chosen and the number of taps N. The final value which is achieved is a function of μ and the noise variance $E[\ \eta^2(n)\ ]$. The fastest rate of convergence that can be obtained under white input conditions is,

$$10 \log_{10} \left(\frac{N}{N-1} \right) \ dB/iteration \tag{3.2.2}$$

and is obtained when μ is $\frac{1}{2N\ E[\ x^2(n)\ ]}$. This gives a final value for ρ of $E[\ \eta^2(n)\ ]$. The value of μ used here, (3.2.1), is smaller than this and hence the rate of convergence would be expected to be slower than (3.2.2) and the final value expected to be smaller than $E[\ \eta^2(n)\ ]$. As noted in [71] , the RLS algorithm outperforms the LMS algorithm even under white input conditions. An important point to note about the performance of the RLS algorithm is the sudden decrease in convergence rate once the noise floor has been reached, Figure 3.2(a). Thus in choosing between the LMS and the RLS for a particular application, consideration must be given as to where the required performance level is with respect to the noise floor. If the performance goal was -50dB then clearly the RLS is the best choice. However if the performance goal was -90dB then the LMS might be expected to reach this level in a similar time to the RLS if the value of μ was set initially for fastest convergence and then decreased by a factor of 5 once the noise floor was reached [26]. The LMS is then the best choice since it is the simplest algorithm. In general, when the noise level is high and the eigenvalue ratio is small, the convergence rates of the LMS and RLS algorithms are almost the same [44]. Turning to the BLMS algorithm, it is evident that under white input conditions its performance is very similar to the LMS algorithm, Figure 3.2(a). The staircase nature of trace '2' emphasises the block nature of the algorithm ie. the tap vector is only updated every N data points.

As the ill-conditioning of the input sequence is increased, Figure 3.2(b), the performance of the LMS algorithm degrades and it takes longer to converge than under white input conditions. The performance of the RLS, on the other hand, is only slightly affected by the increase in eigenvalue ratio. The only noticeable difference being below the noise floor. The BLMS algorithm is inferior to the LMS under these coloured input conditions. Turning to the sliding DFT algorithm, the performance illustrated in Figure 3.2(b) for a mildly ill-conditioned input sequence is very close to the performance of the LMS under white input conditions, Figure 3.2(a). This result emphasises an important point, namely that the performance goal for the

transform domain or quasi-orthogonalising algorithms such as the sliding DFT is LMS performance under white input conditions and not RLS performance. An ideal orthogonalising transform is of coarse the Karhunen-Loeve transform or a lattice filter structure whose PARCOR coefficients where calculated from prior knowledge of the autocorrelation matrix Φ_{xx}. This is in sharp contrast to the least squares lattice structure [52], whose PARCOR coefficients are calculated from $r_{xx}(k)$ and are hence time varying.

Finally if the input sequence is highly ill-conditioned, Figure 3.2(c), the LMS and BLMS algorithms do not converge even as far as the noise floor within a time window of 1000 iterations. In fact for this example the BLMS algorithm appears to be on the verge of instability. Even the performance of the RLS algorithm is degraded under these severe conditions. This result is to be expected in the light of (2.4.19) which is restated here.

$$E[\ (\ \underline{h}(k) - \underline{h}_{opt}\ )\ (\ \underline{h}(k) - \underline{h}_{opt}\ )^T\ ] = \frac{\xi_{opt}}{k}\ \Phi_{xx}^{-1}$$

An expression for the norm can be obtained by taking the trace of the above error covariance matrix and applying the eigenvalue decomposition of (2.5.12).

$$\rho(k) = tr(\ \frac{\xi_{opt}}{k}\ \Phi_{xx}^{-1}\ )$$

$$= \frac{\xi_{opt}}{k}\ tr(\ V\ \Lambda^{-1}\ V^T\ )$$

$$= \frac{\xi_{opt}}{k} \sum_{i=0}^{N-1} \frac{1}{\lambda_i} \qquad (3.2.3)$$

When the input sequence, $\{\ x(n)\ \}$, is white all the eigenvalues are equal and the expression for the norm reduces to,

$$\rho(k) = \frac{\xi_{opt}}{k}\ N. \qquad (3.2.4)$$

Thus the effect of the coloured input sequence is to increase the norm from the the baseline of (3.2.4) by a factor

$$10\ log_{10}\left(\frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{\lambda_i} \right) \qquad dB$$

The sliding DFT adaptive filter has degraded noticeably from the performance of the LMS under white input conditions. Such a loss in performance is to be expected as the sliding DFT is only a quasi-orthonalising structure. Thus the algorithms in order of increasing sensitivity to eigenvalue spread are: RLS, sliding DFT, LMS, and BLMS.

Table 3.1 CHANNEL IMPULSE RESPONSES

Channel No.	Impulse Response	Eigenvalue Ratio †
1	$1.0000 + 0.0000\,z^{-1} + 0.0000\,z^{-2}$	1.0
2	$0.2602 + 0.9298\,z^{-1} + 0.2602\,z^{-2}$	11.8
3	$0.3842 + 0.8704\,z^{-1} + 0.3482\,z^{-2}$	68.6

† the eigenvalue ratio is for a 16-tap filter.

Figure 3.1 SYSTEM IDENTIFICATION

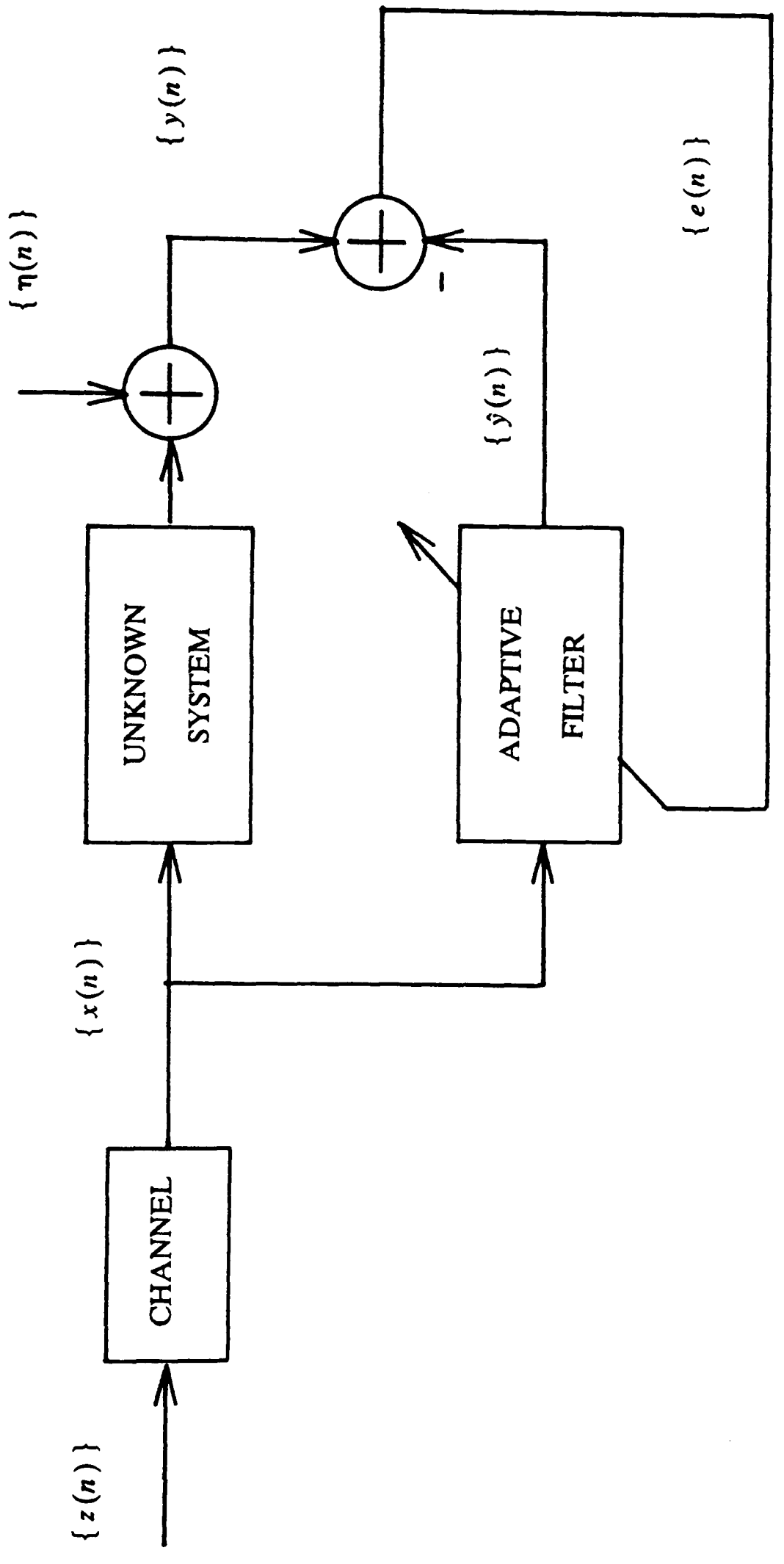
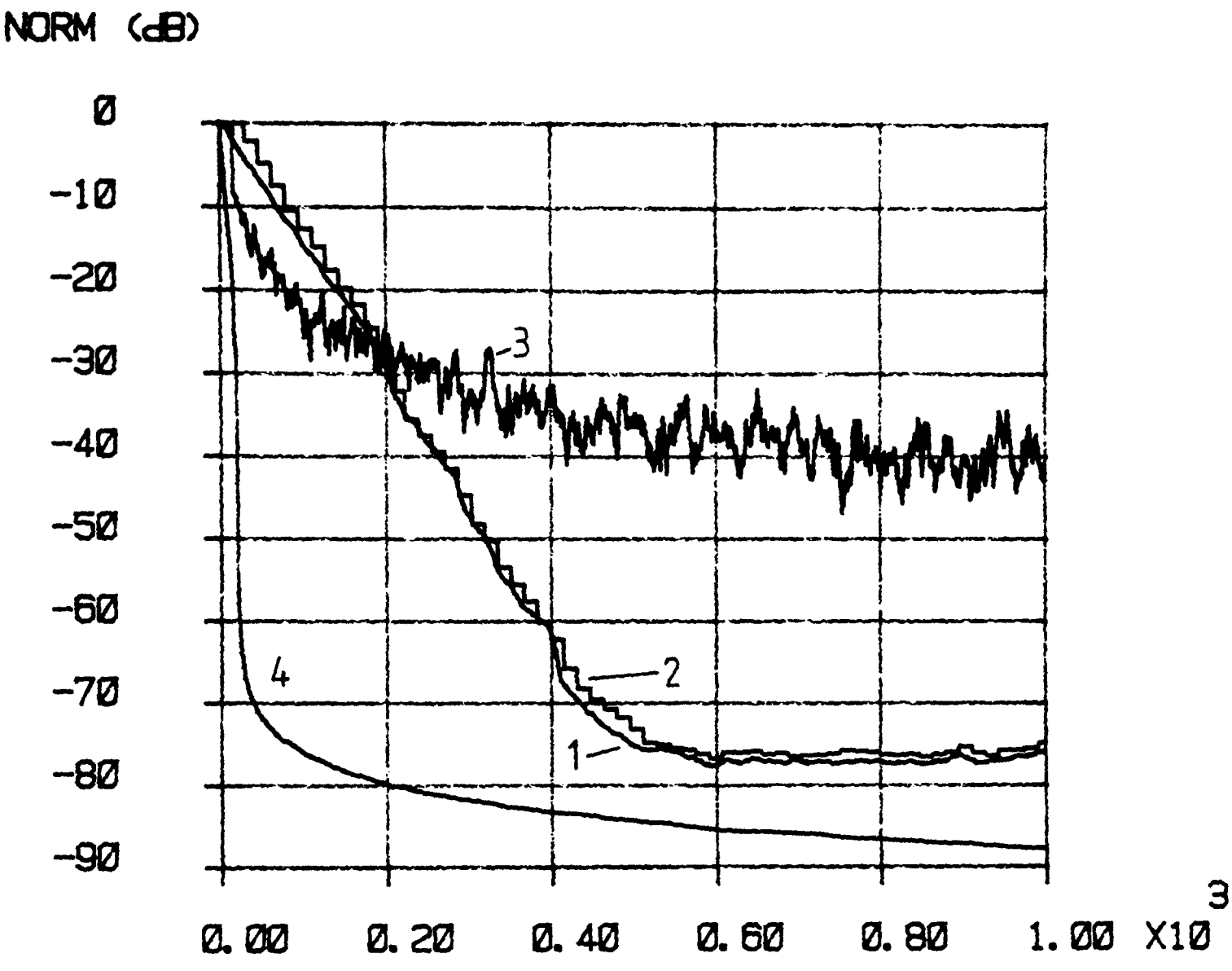


Figure 3.2(a) SIMULATION RESULTS (System Identification)

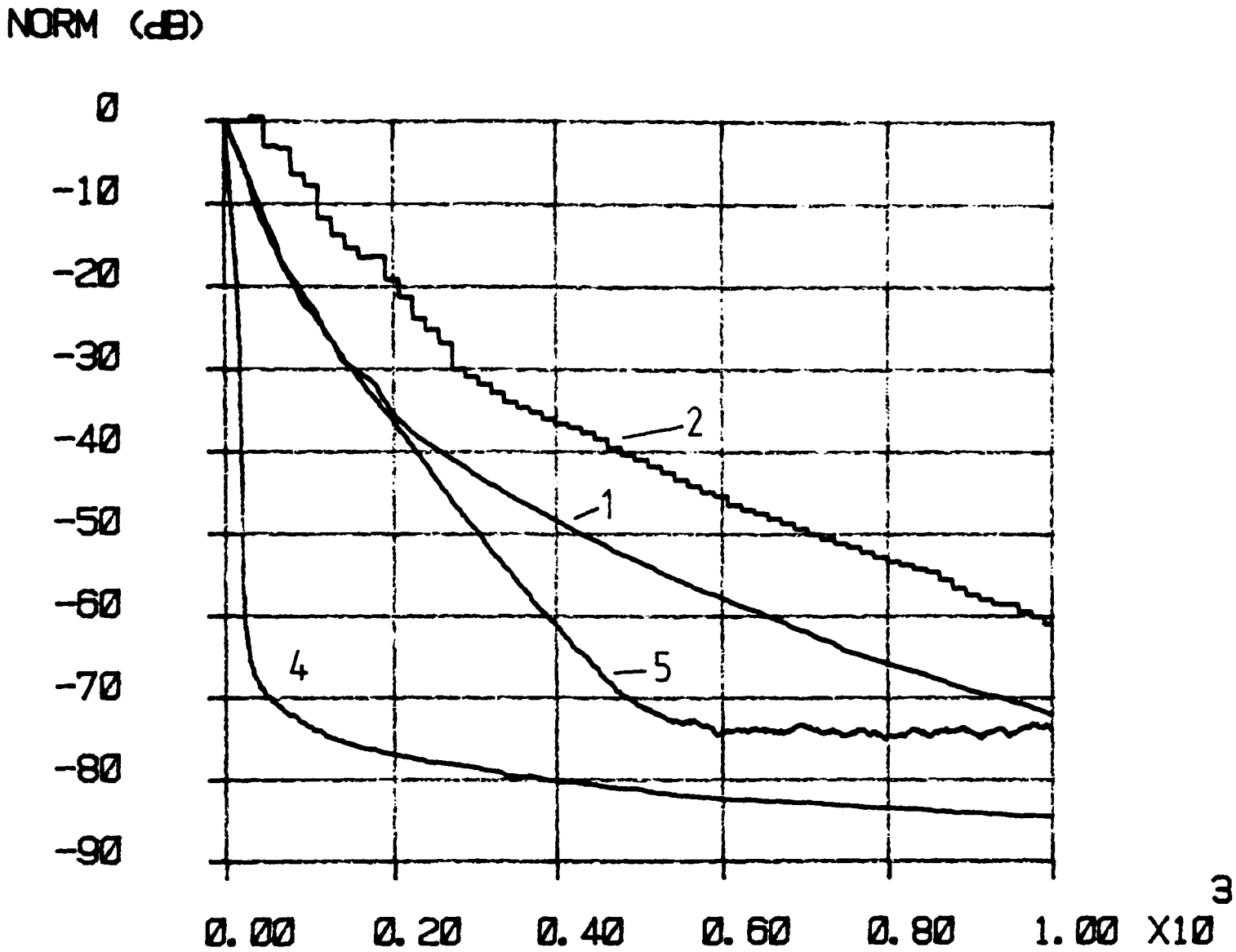
eigenvalue ratio	=	1.0	(channel no. 1)
additive noise	=	-70.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

Figure 3.2(b) SIMULATION RESULTS (System Identification)

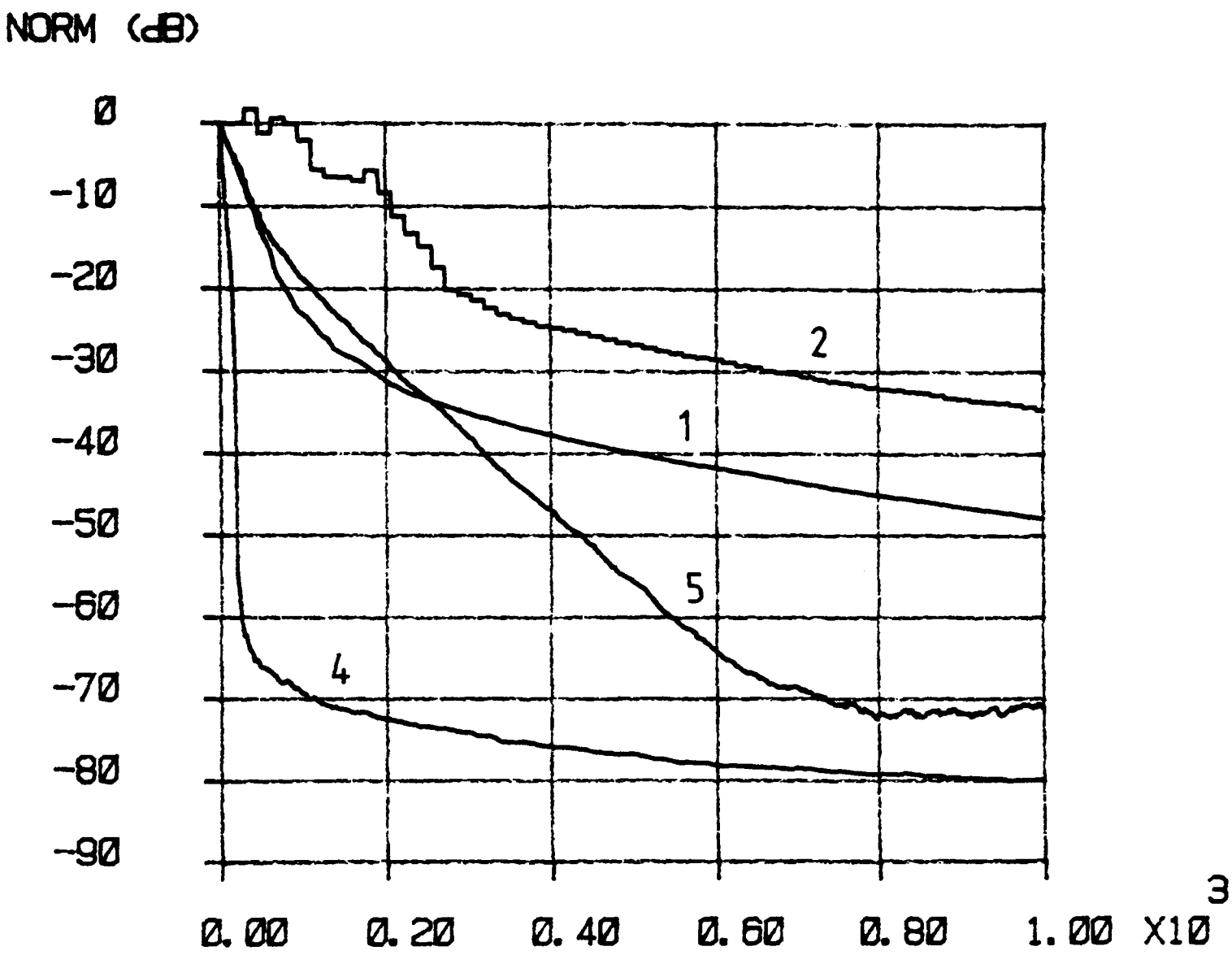
eigenvalue ratio	=	11.8	(channel no. 2)
additive noise	=	-70.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

Figure 3.2(c) SIMULATION RESULTS (System Identification)

eigenvalue ratio	=	68.6	(channel no. 3)
additive noise	=	-70.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

3.3 CHANNEL EQUALISATION

The structure that was used for the simulation of a channel equalisation problem is illustrated in Figure 3.3. For this example the block marked 'channel' is used to model the intersymbol interference on a digital communications channel [9]. The zero mean white binary sequence, $\{ y(n) \}$, represents the digital message which is first convolved with the channel impulse response sequence, $\{ c_n \}$, and then corrupted by adding a zero mean white Gaussian noise term, $\{ v(n) \}$, to form the input to the adaptive filter in the receiver, $\{ x(n) \}$. The adaptive filter then forms a linear estimate, $\hat{y}(n)$, of the each symbol, y(n). As in section 3.2, the number of coefficients in the adaptive filter was 16 and a selection of three channel impulse responses, Table 3.1, was used in order to vary the degree of ill-conditioning on the input sequence to the adaptive filter. The variance of the message sequence, $\{ y(n) \}$, was set to unity, again to provide a convenient reference point for the variance of the additive noise and for calculating the step size μ.

The purpose of the adaptive filter in the receiver is to minimise the MSE,

$$E[(y(n-d) - \hat{y}(n))^2] \qquad (3.3.1)$$

The positive integer d, where $0 \leq d < N$, is to allow for the possibility of fixed lag smoothing, which is often necessary to achieve a particular performance level. For the purposes of these simulations a value of 8 was used for d. The impulse response, $\underline{h}_{opt}$, which minimises the MSE is again provided by the solution of the Wiener equation (2.2.9). However in this scenario, because the additive noise term contributes to the input sequence $\{ x(n) \}$ and hence to the autocorrelation matrix Φ_{xx}, it will effect both the optimum solution $\underline{h}_{opt}$ and the MMSE that is achievable.

The performance measure which was used for this adaptive equalisation problem was the MSE of (3.3.1). This is perhaps not as appropriate as the probability of error for this scenario, but it is more straightforward to calculate and illustrates the transient

behaviour of the adaptive filter algorithms well. The MSE was estimated in all simulations by averaging over an ensemble of 25 runs. In all cases the adaptive filter was initialised with an impulse response vector of zero elements. The variance of the noise sequence, $\{ v(n) \}$, was set to -35 dB with respect to unit variance. The results of the simulations are illustrated in the graphs of Figure 3.4. The labelling of the traces and the choice of μ is consistent with what was used for the system identification problem.

The channel equalisation problem is not as useful as the system identification problem for comparing the performance of a range of adaptive filter algorithms. This is because in changing the channel in order to increase the ill-conditioning of the input sequence, the optimum solution, $\underline{h}_{opt}$, and hence the MMSE is also altered. However channel equalisation is an important practical problem and is a classic example of inverse system modelling or deconvolution. Also, unlike system identification, the training signal to the adaptive filter is noise free. This is reflected particularly in the performance of the RLS algorithm, where the time to converge to the minimum MSE is independent of eigenvalue spread of the input sequence $\{ x(n) \}$, Figure 3.4. Most of what was stated about the other algorithms in section 3.1 is also evident in the channel equalisation problem. The LMS degrades rapidly as the degree of ill-conditioning increases. The performance of the sliding DFT algorithm under mildly coloured input conditions, Figure 3.4(b), is close to LMS performance under white input conditions, Figure 3.4(a). Under severely coloured input conditions, Figure 3.4(c), the performance of sliding DFT algorithm degrades slightly. One notable difference between the channel equalisation examples and the system identification examples is in the performance of the BLMS algorithm. eq. in Figure 3.4(c) the convergence rates of the BLMS and LMS algorithms are almost identical whereas in Figure 3.2(c) they differ significantly. This may seem suprising at first as the autocorrelation matrices for the examples of Figure 3.2(c) and Figure 3.4(c) were almost identical. However the higher order statistics of the input sequence in the two

scenarios were different. Therefore since most of the standard analysis of the LMS and BLMS algorithms only take account of first and second order statistics [26, 14], or assume that all random variables are Gaussian [27, 28], it is to be expected that some changes in performance will occur when the distribution of the input and training signals change in such a way as to leave their first and second order moments unaltered. Thus, for the channel equalisation problem, the algorithms, in order of increasing sensitivity to eigenvalue spread, are: RLS, sliding DFT, and LMS/BLMS.

Figure 3.3 CHANNEL EQUALISATION

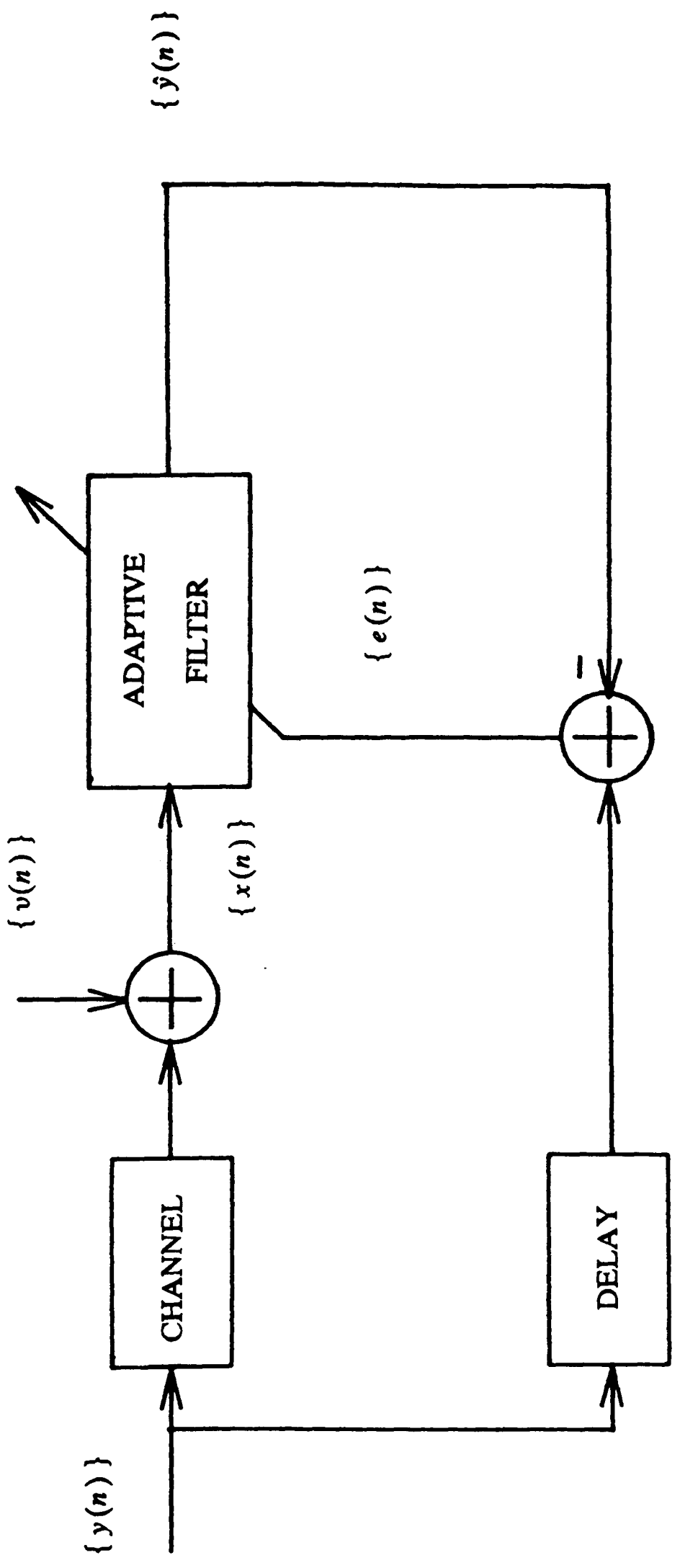
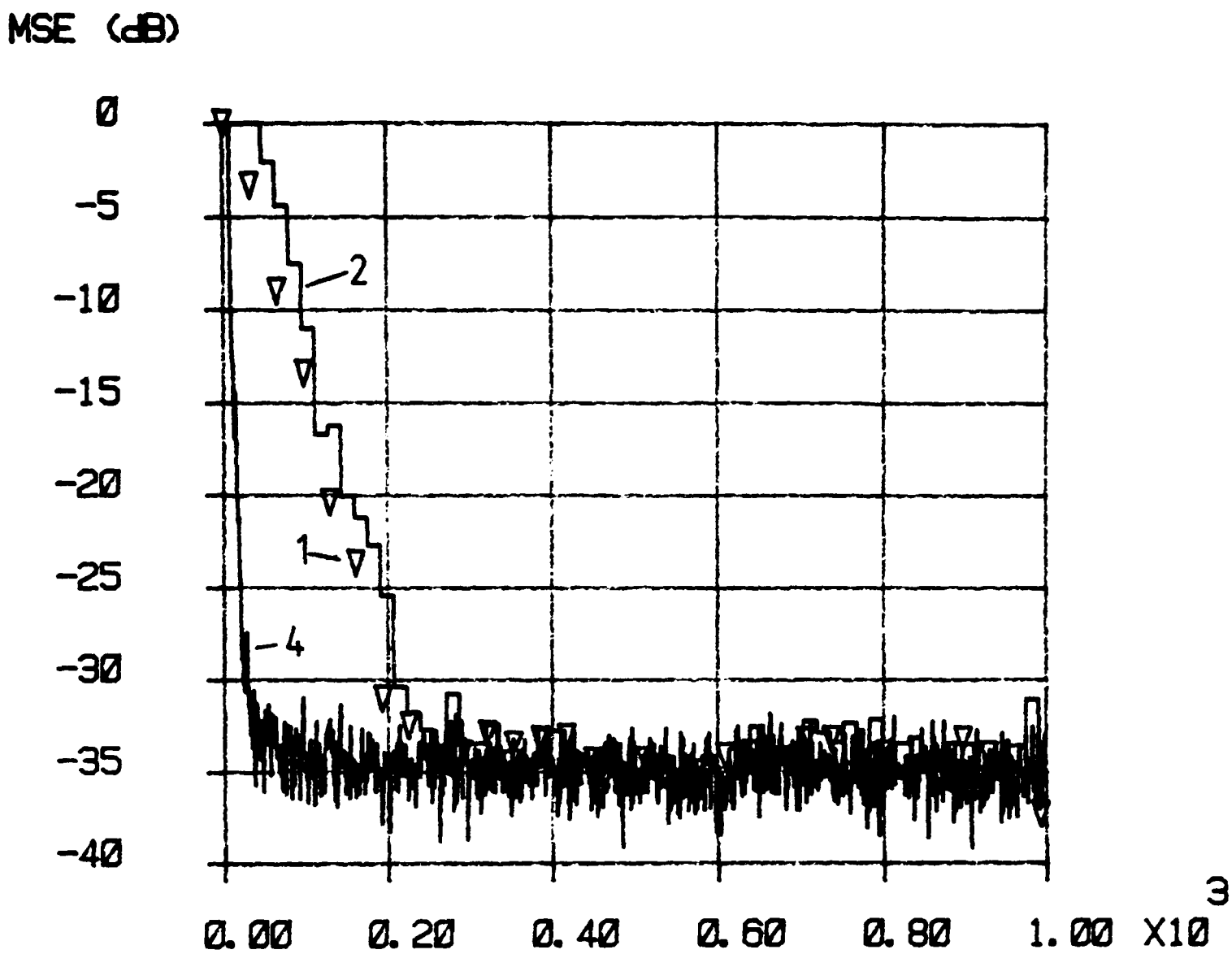


Figure 3.4(a) SIMULATION RESULTS (Channel Equalisation)

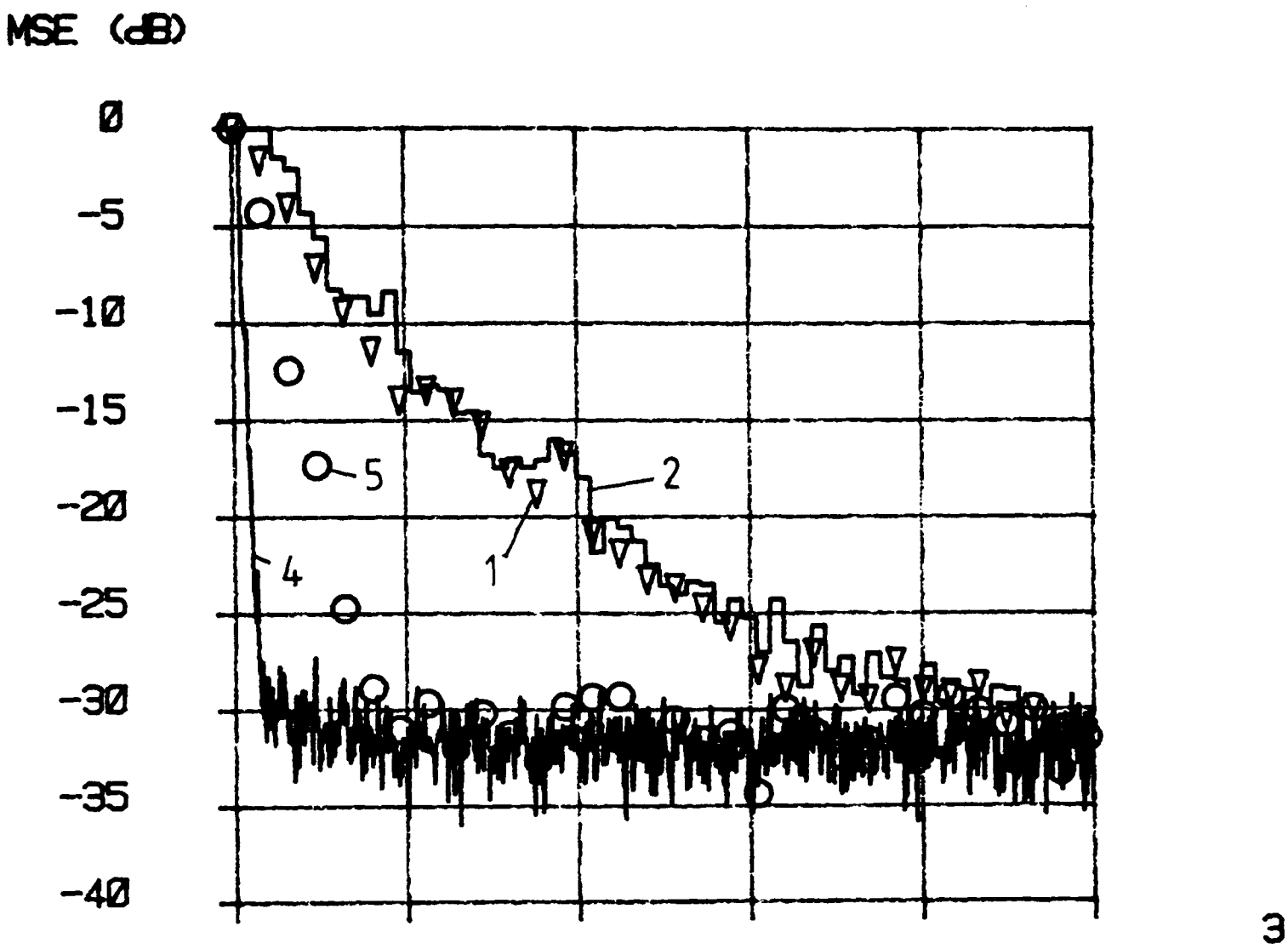
eigenvalue ratio	=	1	(channel no. 1)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

Figure 3.4(b) SIMULATION RESULTS (Channel Equalisation)

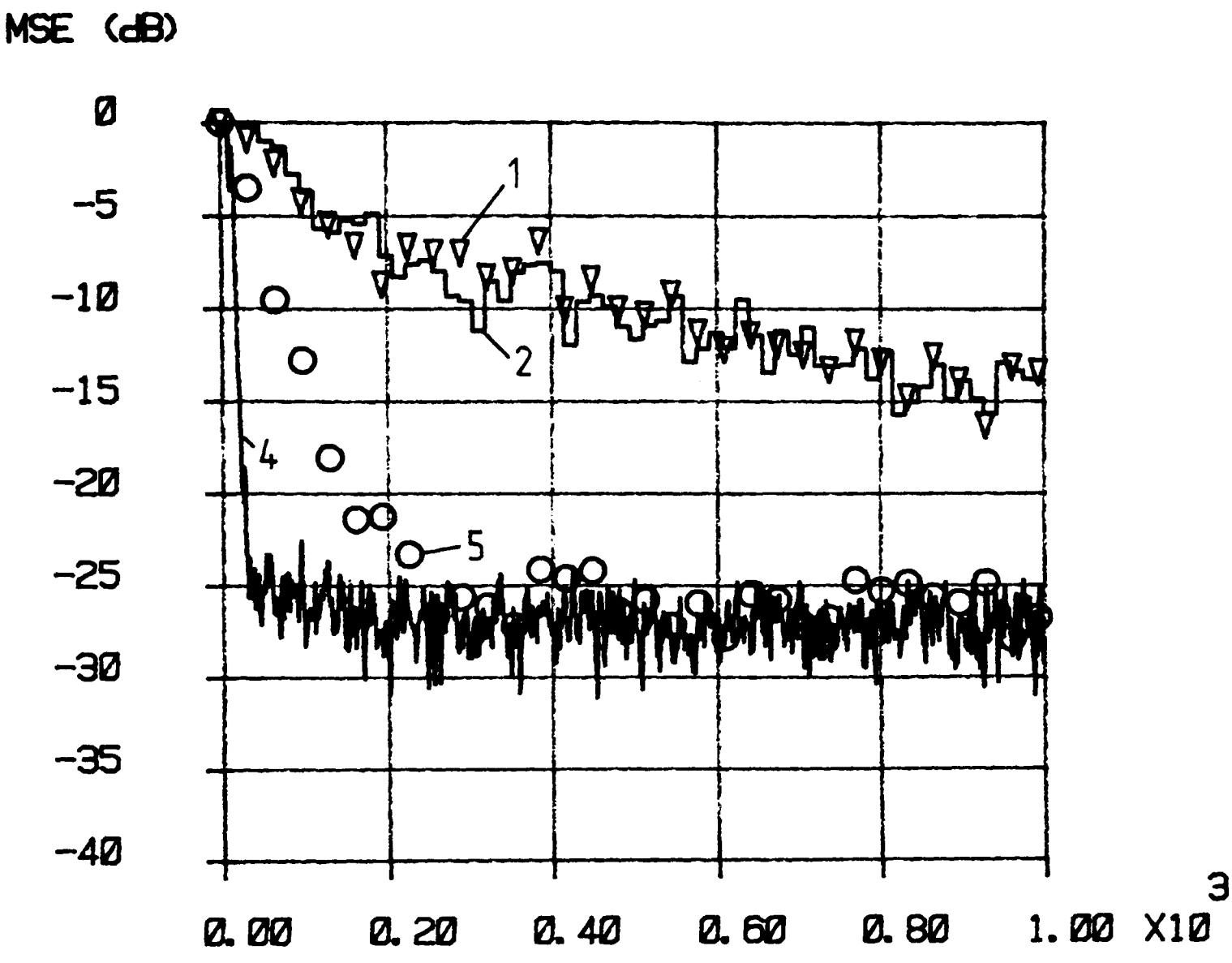
eigenvalue ratio	=	11.8	(channel no. 2)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

Figure 3.4(c) SIMULATION RESULTS (Channel Equalisation)

eigenvalue ratio	=	68.6	(channel no. 3)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	



KEY	
1	LMS
2	BLMS
3	SM
4	RLS
5	Sliding DFT

3.4 SUMMARY AND CONCLUSIONS

Because of the difficulty in obtaining rigorous analytic results for the convergence properties of a broad selection of adaptive FIR filter algorithms, an experimental comparison was made using computer simulation. The results of these experiments confirm many of the key properties suggested by approximate analysis. In particular, the performance degradation of the SG algorithms when the input sequence is highly ill-conditioned, the fast consistent convergence of the LS algorithms, and the role of the quasi-orthogonalising algorithms as a compromise in performance between the LMS and the RLS algorithms. It is not believed that a broad comparison of this nature appears anywhere else in the literature.

Chapter 4

A SELF ORTHOGONALISED BLOCK ADAPTIVE FILTER

4.1 INTRODUCTION

In the area of adaptive filtering the RLS [24] and the LMS [21] are the two major alternatives in a trade off of convergence performance against computational complexity. The conventional RLS algorithm requires a number of computations per new data point that is a function of the square of the number of coefficients (N) in the finite impulse response (FIR) filter ie. order N^2 ($O(N^2)$). By exploiting the shift invariance properties [25] this has been reduced to $O(N)$. This and subsequent developments [52, 55, 47, 48] make available the consistent rapid MSE convergence properties of the RLS algorithms at a computational cost, which is of the same order as the more commonly used LMS algorithm, whose convergence properties are generally poor [26]. However the RLS algorithm still represents a computational load which is significantly higher than the LMS algorithm. Typical figures being 10N multiplications per new data point for the RLS algorithm compared with 2N for the LMS algorithm. The original aim of work that is presented in this chapter and in [74] was to find an algorithm that lay between the RLS and the LMS in both computational complexity and performance and whose rate of convergence was independent of the input signal conditioning. In fact the algorithm that has been developed goes beyond this initial goal in that it represents a significant reduction in computational load compared to an LMS algorithm for moderate to large values of N.

The first question to pose is what algorithms already exist which might provide a combination of computational complexity and performance which is between that of the LMS and RLS algorithms? A survey of the literature rapidly yields the term self orthogonalising. The concept originated in [75, 76, 77] and was a result of the convergence analysis of the LMS algorithm and the recognition of the associated

dependence of the rate of convergence of the LMS algorithm on the eigenvalues of the input autocorrelation matrix [21]. A self orthogonalised algorithm involves constructing a linear operator (transform or preprocessor) which maps the input N-vector $\underline{x}$ to an N-vector $\underline{u}$ such that the elements of $\underline{u}$ are mutually orthogonal. Given this the matrix $E[\,\underline{u}\underline{u}^T\,]$ is a diagonal whose eigenvalue spread can be normalised to unity by dividing each element of $\underline{u}$ by the square root of the appropriate eigenvalue. The resultant N-vector $\underline{z}$ is white with unit variance ie. $E[\,\underline{z}\underline{z}\,]$ is an $(N \times N)$ identity matrix. If the vector $\underline{z}$ forms the input to an LMS algorithm it is straightforward to predict using the convergence analyses of [26] and [27] that the complete structure (linear operator + eigenvalue normalisation + LMS) will converge (in a MSE sense) under any input conditions at the same rate as an LMS algorithm would under white input conditions. This technique is equivalent to multiplying the gradient term in an LMS algorithm by the inverse of the input autocorrelation matrix $E[\,\underline{x}\underline{x}^T\,]$ [77, 29].

Only in a limited number of applications such as [78] is the input autocorrelation matrix, or equivalently an orthogonalising operator, known a priori and hence for general purpose applications two suboptimum techniques have been suggested. In the first a fixed linear operator such as a DFT or DCT is chosen that performs an approximate orthogonalisation of the input vector [29, 67]. The subsequent processing proceeds as if the orthogonalisation was exact. In the second the autocorrelation matrix is estimated directly from the data, inverted and used to multiply the gradient estimate [77]. These two techniques might be classified as explicit and implicit orthogonalisation respectively. However it should be noted that the form of estimate for the input autocorrelation matrix identified in [77] as the ideal self orthogonalising algorithm gives rise to the RLS algorithm. It is clear from [71] that even under white input conditions the RLS algorithm may outperform the LMS algorithm. Consequently it must be concluded that an RLS algorithm does more than merely orthogonalise the input signal and therefore it should not be classified as an self orthogonalising algorithm.

Explicit orthogonalisation techniques have been applied to FFT [79] based block adaptive filters [80, 81]. The FFT is used in the BLMS algorithms of [14] and [82] to provide fast convolution and fast estimation of the gradient. As a by-product of this implementation the FFT of an augmented input vector is available ie. a fixed transform that performs approximate orthogonalisation. Bartlett spectral estimation has also been considered in an attempt to improve the quality of this approximation [81]. If however other fast convolution algorithms such as the Fermat number transform (FNT) [83] or the rectangular transform (RT) [62] are to be applied to a self orthogonalised block adaptive filter as they have been applied to the BLMS algorithm [84, 85] then since they cannot be assumed to exhibit even approximate orthogonalising properties an implicit approach must be considered.

The self orthogonalising block adaptive filter (SOBAF) [74] that is described in this chapter is a unique alternative to the RLS and LMS algorithms. It provides a combination of computational load, which is significantly less than the LMS algorithm, and consistent convergence performance, which lies between that of the LMS and RLS algorithms, but unlike the LMS is virtually independent of the input statistics. Therefore it is well suited to applications where neither the LMS nor the RLS algorithm can provide the correct trade off of computational load against convergence performance. The computational efficiency is achieved by using a block filtering structure which is similar to the BLMS algorithm [14] and hence may exploit either FFT [79] or RT [62] efficient circular convolution algorithms. The convergence performance is achieved by using an implicit self orthogonalising technique, which ensures that the algorithm will converge under any input conditions at the same rate as an LMS algorithm would under white input conditions.

The chapter is subdivided in the following manner. In section 4.2 the theoretical development of the algorithm is presented and the results verified, in section 4.3, by computer simulation. Section 4.4 contains the arguments that lead to a practical SOBAF algorithm along with details of how it can be implemented efficiently. In

section 4.5 the computational load of the proposed filter is assessed. Finally, in section 4.6, results from computer simulations are presented which confirm that the practical SOBAF achieves the convergence performance that was promised in the theoretical considerations of section 4.2.

4.2 THEORY

Consider a stationary sequence of N-vectors $\{ \underline{x}(i) \}$, which is zero mean uncorrelated in time and jointly Gaussian. The sequence is completely described by the $(N \times N)$ autocorrelation matrix Φ_{xx} where,

$$\Phi_{xx} = E[\underline{x}(i) \underline{x}^T(i)] .$$

Although the matrix is positive semi-definite, in many applications it can be assumed to be positive definite in which case there exists an $(N \times N)$ matrix Q such that

$$Q^T \Phi_{xx} Q = I_N$$

where I_N is an $(N \times N)$ identity matrix. The matrix Q is not unique [39]. A second sequence of N-vectors, $\{ \underline{z}(i) \}$, which is uncorrelated in time and zero mean may be generated from $\{ \underline{x}(i) \}$ using the matrix Q^T.

$$\underline{z}(i) = Q^T \underline{x}(i) \tag{4.2.1}$$

Therefore Q^T may be considered to be a whitening filter since,

$$\Phi_{zz} = E[\underline{z}(i) \underline{z}^T(i)] = I_N$$

Because of the special structure of Φ_{zz} ie. it is diagonal with equal eigenvalues, a stochastic gradient search adaptive filter with input $\{ \underline{z}(i) \}$ will achieve rapid consistent convergence rates. Further, since $\{ \underline{z}(i) \}$ is also uncorrelated in time zero mean and jointly Gaussian [24], the theoretical results of [27,28] may be applied directly to give: (i) bounds on the step size μ_b that ensure convergence in a MSE

sense, and (ii) a single optimum value μ_{opt} for fastest convergence.

To the sequence of N-vectors $\{ \underline{z}(i) \}$ apply a BLMS algorithm [14] of block length L to form an estimate $\{ \hat{y}(i) \}$ of the stationary scalar sequence $\{ y(i) \}$. The aim is to minimise the MSE

$$E[(y(i) - \hat{y}(i))^2]$$

The estimate $\hat{y}(i)$ is linear and is formed using the weight N-vector $\underline{w}$ ie.

$$\hat{y}(i) = \underline{z}^T(i)\, \underline{w} \tag{4.2.2}$$

The optimum solution which minimises the MSE is given by,

$$\underline{w}_{opt} = \Phi_{zz}^{-1} E[\underline{z}(i)\, y(i)]$$

$$= E[\underline{z}(i)\, y(i)] .$$

The BLMS algorithm is defined by the following three equations [14].

$$\underline{\hat{y}}(j) = \underline{\zeta}(j)\, \underline{w}(j-1) \tag{4.2.3}$$

$$\underline{e}(j) = \underline{y}(j) - \underline{\hat{y}}(j) \tag{4.2.4}$$

$$\underline{w}(j) = \underline{w}(j-1) + \frac{2\mu_b}{L} \underline{\zeta}^T(j)\, \underline{e}(j) \tag{4.2.5}$$

where

$$\underline{y}(j) = [y(jL)\, y(jL-1) \ \cdots \ y(jL-L+1)]^T$$

$$\underline{\hat{y}}(j) = [\hat{y}(jL)\, \hat{y}(jL-1) \ \cdots \ \hat{y}(jL-L+1)]^T$$

and

$$\underline{\zeta}^T(j) = [\underline{z}(jL)\, \underline{z}(jL-1) \ \cdots \ \underline{z}(jL-L+1)]$$

The index j is known as the block number, where a block contains L data vectors.

The weight vector $\underline{w}$ is only updated once per block.

Using the theoretical results of [28], a recursive relationship for the block MSE σ_e^2 is obtained.

$$\sigma_e^2(j) = \left(1 - 4\mu_b + \frac{4\mu_b^2}{L}(L+N+1) \right) \sigma_e^2(j-1) + \left(4\mu_b - \frac{4\mu_b^2}{L}(L+1) \right) \sigma_{opt}^2 \tag{4.2.6}$$

where

$$\sigma_e^2(j) = \frac{1}{L} E[\, \underline{e}^T(j)\, \underline{e}(j)\,]$$

and σ_{opt}^2 is the minimum MSE that is obtained when the weight vector $\underline{w}_{opt}$ is used.

$$\sigma_{opt}^2 = E[\, (\, y(i) - \underline{w}_{opt}^T\, \underline{z}(i)\,)^2\,]$$

Equation (4.2.6) yields bounds that ensure MSE convergence,

$$0 \leq \mu_b < \frac{L}{(L+N+1)} \tag{4.2.7}$$

and a value μ_{opt} which gives fastest convergence [28].

$$\mu_{opt} = \frac{L}{2(L+N+1)}$$

Equations (4.2.1), (4.2.3), (4.2.4), and (4.2.5) thus define a self orthogonalising block adaptive filter whose MSE convergence is ensured provided μ_b is chosen within the limits of (4.2.7) and whose rate of convergence is independent of the eigenvalues of the autocorrelation matrix Φ_{xx}, (4.2.6).

This self orthogonalising block adaptive filter may be reformulated in terms of an overall weight vector $\underline{h}$ where

$$\hat{y}(i) = \underline{x}^T(i)\, \underline{h} \tag{4.2.9}$$

such that explicit knowledge of the matrix Q is unnecessary. Combining (4.2.9) and (4.2.1),

$$\hat{y}(i) = \underline{x}^T(i)\, Q\, \underline{w}$$

from which a relationship between $\underline{h}$ and $\underline{w}$ is obtained.

$$\underline{h} = Q\, \underline{w} \tag{4.2.10}$$

Application of (4.2.1) and (4.2.10) to (4.2.3) and (4.2.5) yields

$$\underline{\hat{y}}(j) = \underline{\chi}(j)\, \underline{h}(j-1) \tag{4.2.11}$$

and

$$\underline{h}(j) = \underline{h}(j-1) + \frac{2\mu_b}{L}\, \Phi_{xx}^{-1}\, \underline{\chi}^T(j)\, \underline{e}(j) \tag{4.2.12}$$

where

$$\underline{\chi}^T(j) = [\, x(jL)\; x(jL-1)\; \cdots\; x(jL-L+1)\,] .$$

4.2.1 Comparison of Theory with Simulation

The theoretical results presented above rely on the assumption that the sequence $\{\underline{x}(i)\}$ is uncorrelated in time and jointly Gaussian. For an adaptive transversal filter application the sequence $\{\underline{x}(i)\}$ is never uncorrelated in time since

$$\underline{x}(i) = [\, x(i)\; x(i-1)\; \cdots\; x(i-N+1)\,]^T$$

and

$$\underline{x}(i-1) = [\, x(i-1)\; x(i-2)\; \cdots\; x(i-N)\,]^T$$

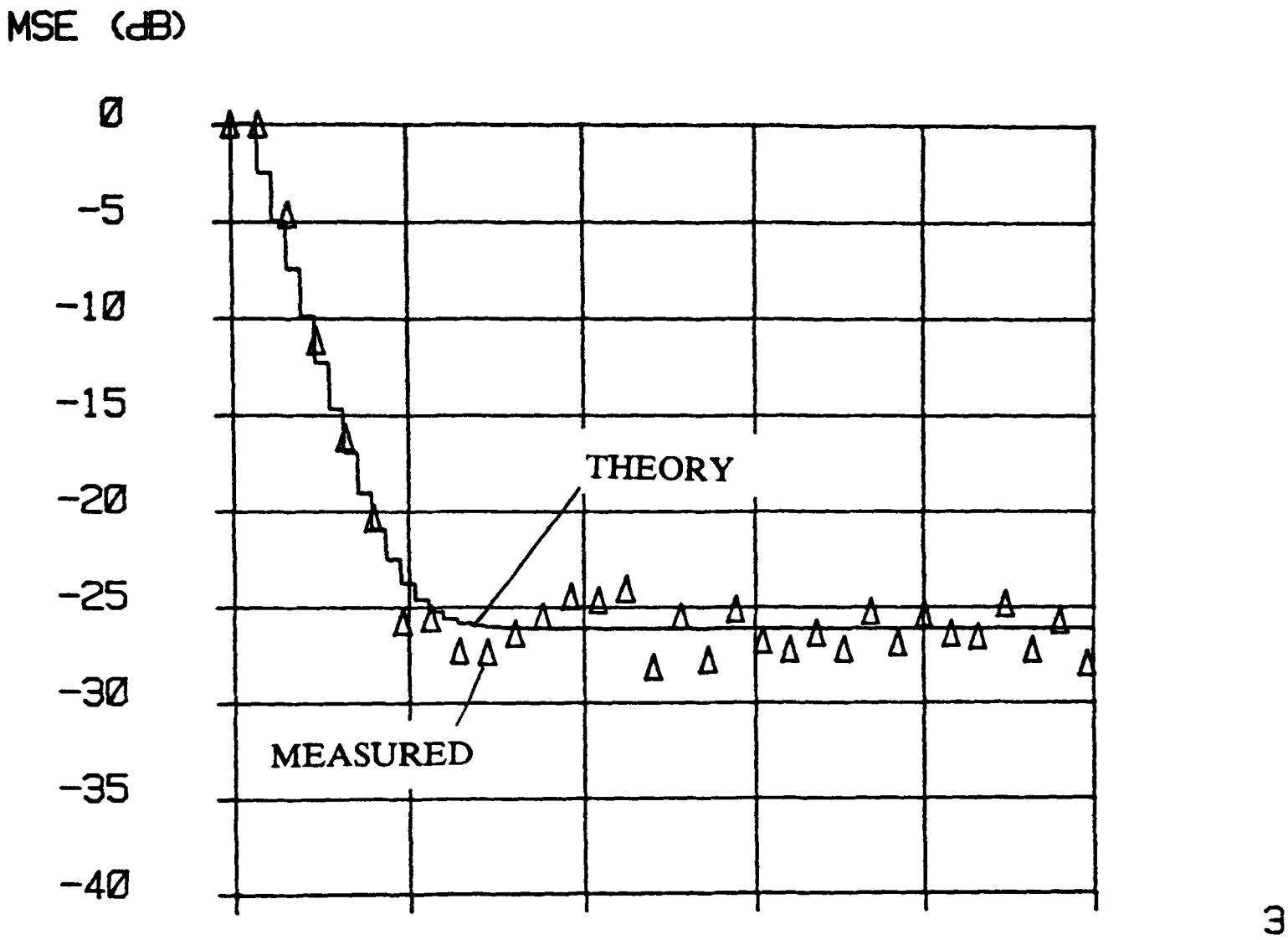
and is rarely exactly Gaussian. The aim of this section is to test the validity of the theoretical convergence results, summarised in (4.2.6), for the particular case of an

adaptive transversal communications channel equaliser where neither assumption is true.

A typical equaliser scenario is illustrated in Figure 3.3. The digital message is a zero mean binary distributed white random sequence $\{ y(n) \}$. The channel is modelled by a FIR filter whose output is corrupted by a zero mean white Gaussian sequence $\{ v(n) \}$. The role of the adaptive filter is to form a fixed lag estimate of the channel input. The training signal for the adaptive filter is thus $\{ y(n-d) \}$, where d is a positive integer. For the purposes of the simulations presented here, a 3-tap channel was used (channel no.3, Table 3.1). The signal-noise ratio, defined as $E[y^2]/E[v^2]$, was set at 35dB. The self orthogonalising adaptive algorithm, defined by (4.2.4), (4.2.7), (4.2.11), and (4.2.12), was used to update a 16-tap transversal equaliser. The block length, L, was set at 16. Under these conditions the autocorrelation matrix, Φ_{xx}, has a maximum/minimum eigenvalue ratio of 68.8. To be consistent with chapter 3 the step size, μ_b, was set at $\frac{1}{6}$. The convergence performance of the algorithm is illustrated in Figure 4.1 on which is shown both a measured MSE calculated from an ensemble of 25 runs and a theoretical MSE calculated from (4.2.6). This clearly emphasises the close agreement between theory and practice.

Figure 4.1 COMPARISON OF THEORY WITH SIMULATION

eigenvalue ratio	=	68.6	(channel no. 3)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	



4.3 A PRACTICAL ALGORITHM

In this section a new block adaptive filter algorithm is described. This algorithm is a unique combination of three concepts. The first involves the performance goal, which is chosen to be the MSE convergence of a self orthogonalised stochastic gradient search algorithm, summarised in (4.2.6). In words, the adaptive filter should converge, in a MSE sense, under any input conditions, at the same rate as it would if the input sequence was white. Thus the self orthogonalised algorithm will not, by definition, exhibit a sensitivity to the eigenvalue spread of the autocorrelation matrix Φ_{xx} that is a characteristic of both LMS [26, 27] and BLMS [28] algorithms. However it should also be noted that this performance goal is not equivalent to the performance of a RLS algorithm, since even under white input conditions the RLS algorithm will outperform a stochastic gradient search algorithm [71].

The second concept involves the choice of an estimator for the autocorrelation matrix. In a general adaptive filter application, the autocorrelation matrix is unknown and hence the tap weight update, (4.2.12), must be replaced by,

$$\underline{h}(j) = \underline{h}(j-1) + \frac{2\mu_b}{L} \hat{\Phi}_{xx}^{-1}(j)\, \underline{\chi}^T(j)\, \underline{e}(j) \tag{4.3.1}$$

where $\hat{\Phi}_{xx}(j)$ is an estimate of Φ_{xx} at block j. Several possible estimates of Φ_{xx} exist in the literature. The most notable is.

$$\hat{\Phi}_{xx}(j) = \frac{1}{jL} \sum_{i=1}^{jL} \underline{x}(i)\, \underline{x}^T(i) \tag{4.3.2}$$

However the use of this estimate would produce a RLS block adaptive filter structure [85]. Hence it is considered inappropriate here as its convergence performance would not be that of a self orthogonalised stochastic gradient filter. Further, computationally efficient RLS block adaptive filter algorithms already exist [38]. In the estimation technique that is considered here the matrix $\hat{\Phi}_{xx}(j)$ is assumed to be symmetric Toeplitz. Thus each element may be generated from knowledge of the first column

$\underline{\hat{p}}(j)$ where

$$\underline{\hat{p}}(j) = [\, \hat{p}_0(jL)\, \hat{p}_1(jL) \;\cdots\; \hat{p}_{N-1}(jL)\,]^T$$

and

$$\hat{p}_k(jL) = \frac{1}{jL} \sum_{i=1}^{jL} x(i)\, x(i-k) \qquad 0 \le k < N \;.$$

The vector $\underline{\hat{p}}(j)$ may thus be updated on a block by block basis.

$$\underline{\hat{p}}'(j) = \underline{\hat{p}}'(j-1) + \frac{1}{L}\, \delta\underline{p}(j) \tag{4.3.3}$$

$$\underline{\hat{p}}(j) = \frac{1}{j}\, \underline{\hat{p}}'(j) \tag{4.3.4}$$

where

$$\delta p(j) = \underline{\chi}^T(j)\, \underline{x}_L(j) \tag{4.3.5}$$

and

$$\underline{x}_L = [\, x(jL)\, x(jL-1) \;\cdots\; x(jL-L+1)\,]^T \;.$$

The Toeplitz assumption also allows the application of computationally efficient techniques such as the Levinson recursion [22] and more recently [86] for the solution of

$$\hat{\underline{\Phi}}_{xx}(j)\, \delta\underline{h}(j) = \underline{c}(j) \tag{4.3.6}$$

where

$$\underline{c}(j) = \underline{\chi}^T(j)\, \underline{e}(j) \;. \tag{4.3.7}$$

It is well known in equaliser [31], and spectral estimation [11] applications that the Toeplitz assumption produces poorer performance than the estimate of (4.2.2). Hence the choice of a Toeplitz assumption here may be interpreted as reflecting a desire to degrade the performance of the algorithm from that of an RLS structure to that of a

self orthogonalised structure.

The third concept involves the application of computationally efficient circular convolution algorithms, of which the fast Fourier transform (FFT) [79] and the rectangular transform (RT) [62] are but two examples, to produce an adaptive filter algorithm which is itself computationally efficient. This technique has been the motivation behind the development of the BLMS algorithm, which is computationally superior to the LMS algorithm. To utilise this technique, the linear convolution operations are first identified. In this case they are (4.2.11), (4.3.5), and (4.3.7). Of these three, (4.2.11) and (4.3.7) are common to both the BLMS and the self orthogonalised structures. The existence of (4.3.5) is a direct result of the Toeplitz assumption on $\hat{\underline{\Phi}}_{xx}$, and is a significant factor in making that assumption. Each of these linear convolution operations is then performed using a combination of either overlap-add or overlap-save data sectioning [7] and a circular convolution algorithm. To simplify the notation, only overlap-save and a block length $L = N$ will be considered here as these are known to produce the most efficient adaptive filter structures [61] This does not however detract from the generality of the results.

The linear convolution of (4.2.11), (4.3.5), and (4.3.7) are obtained in the following way. Taking (4.2.11) as an example,

$$\underline{\hat{y}}(j) = \underline{x}(j)\, \underline{h}(j-1)$$

$$= [\, O_N \;\; T_N \,] \left\{ \begin{bmatrix} T_N \; \underline{x}(j-1) \\ T_N \; \underline{x}(j) \end{bmatrix} * \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}(j-1) \right\}$$

$$= [\, O_N \;\; T_N \,]\, C_{2N} \left\{ X(j) \; x \; H(j-1) \right\}$$

where

$$\underline{X}(j) = B_{2N} \begin{bmatrix} T_N \, \underline{x}(j-1) \\ T_N \, \underline{x}(j) \end{bmatrix}$$

and

$$\underline{H}(j-1) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}(j-1) \, .$$

The two $(M \times 2N)$ matrices A_{2N} and B_{2N} and the $(2N \times M)$ matrix, C_{2N}, define a circular convolution machine which operates on 2N-vectors, $M \geq 2N$. The matrix I_N is an $(N \times N)$ identity matrix, the matrix O_N is $(N \times N)$ with all zero elements, and the $(N \times N)$ time reversal matrix, T_N, has 1's on the secondary diagonal and zeros elsewhere. The complete algorithm is summarised in Table 4.1.

Table 4.1 A SELF ORTHOGONALISED BLOCK ADAPTIVE FILTER

$$\underline{X}(j) = B_{2N} \begin{bmatrix} T_N\ \underline{x}(j-1) \\ T_N\ \underline{x}(j) \end{bmatrix}$$

$$\underline{H}(j-1) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}(j-1)$$

$$\underline{\hat{y}}(j) = [\ O_N\ T_N\ ]\ C_{2N} \left\{ \underline{X}(j) \times \underline{H}(j-1) \right\}$$

$$\underline{e}(j) = \underline{y}(j) - \underline{\hat{y}}(j)$$

$$\underline{E}(j) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{e}(j)$$

$$\underline{c}(j) = [\ O_N\ T_N\ ]\ C_{2N} \left\{ \underline{X}(j) \times \underline{E}(j) \right\}$$

$$\underline{X}'(j) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{x}(j)$$

$$\delta\underline{p}(j) = [\ O_N\ T_N\ ]\ C_{2N} \left\{ \underline{X}(j) \times \underline{X}'(j) \right\}$$

$$\underline{\hat{p}}'(j) = \underline{\hat{p}}'(j-1) + \frac{1}{N} \delta\underline{p}(j)$$

$$\underline{\hat{p}}(j) = \frac{1}{j} \underline{\hat{p}}'(j)$$

$$\delta\underline{h}(j) = \underline{\hat{\Phi}}_{xx}^{-1}(j)\, \underline{c}(j)$$

$$\underline{h}(j) = \underline{h}(j-1) + \frac{2\mu_b}{N} \delta\underline{h}(j)$$

4.4 COMPUTATIONAL COMPLEXITY

Computationally efficient implementations of the SOBAF are obtained when the A_{2N}, B_{2N} and C_{2N} matrices are replaced with the appropriate matrices which define the RT, FFT, or NTT. An FFT based implementation is considered here for the evaluation of the computational load of the algorithm. The particular FFT algorithm is based on the radix-2 formulation [1] and efficiently exploits the fact that real data rather than complex data processing is required [87]. It is also assumed that a complex multiplication is implemented through four real multiplications and two real additions. The Toeplitz system of equations, (4.3.6), is solved using the Levinson recursion [22]. Given these assumptions, the average computational load involved in processing each new data point is as given below.

$$A = 2N + 21 \log_2 N + 41 + \frac{14}{N} \tag{4.4.1}$$

$$M = 2N + 14 \log_2 N + 11 + \frac{28}{N} \tag{4.4.2}$$

$$D = 1 - \frac{1}{N} \tag{4.4.3}$$

The symbols A, M and D denote numbers of additions and/or subtractions, multiplications and divisions respectively [74].

In computing the number of operations in the filter structure, the well known Levinson recursion [22] has been used to solve the Toeplitz system of equations, (4.3.6). Significant further computational savings can be achieved by exploiting recently reported fast algorithms for solving the Toeplitz system of equations [88, 89, 86]. The technique dealt with in [86] is of particular interest since the FFT algorithm is employed to perform block convolution. The entire algorithm of [86] requires

$$2.5N \log_2 N \log_2 N + 11.5N \log_2 N + 6N \quad \textit{multiplications}$$

and the same number of additions to solve a Toeplitz system of N equations. Applying this technique to the FFT based SOBAF and computing the number of operations, the average number of operations per single output sample of the filter is found to be

$$A = \log_2 N \left[2.5 \log_2 N + 25.5 \right] + 18 + \frac{31}{N} \tag{4.4.4}$$

$$M = \log_2 N \left[2.5 \log_2 N + 32.5 \right] + 47 + \frac{14}{N} \tag{4.4.5}$$

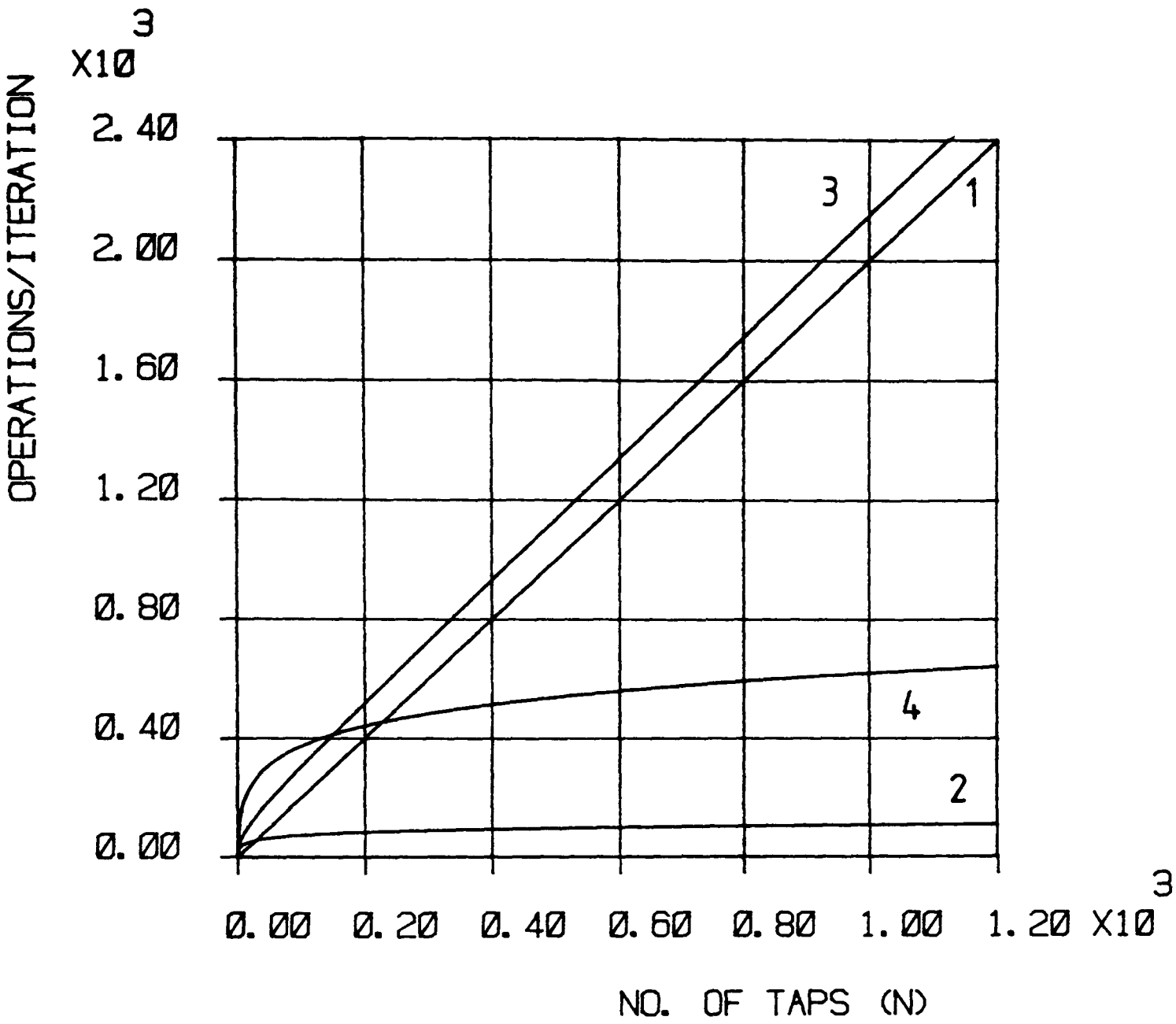
$$D = 1 - \frac{1}{N} \tag{4.4.6}$$

The computational load of an LMS algorithm is $O(N)$, ie. the number of operations increases linearly with the number of taps in the transversal filter. An FFT based BLMS algorithm on the other hand is $O(\textit{logN})$. Thus an FFT based BLMS algorithm has a significant advantage in computational efficiency over an LMS algorithm for moderate to large N. As mentioned already, the SOBAF requires the solution of a Toeplitz set of equations. Using the Levinson recursion this requires $O(N^2)$ operations, which reduces to $O(N)$ since the equations are only solved once per block. Although the remaining operations in (4.4.1) and (4.4.2) are at most $O(\textit{logN})$, the linear term will dominate and hence the overall computational load is $O(N)$, which is the same as an LMS algorithm. If however the fast inversion technique of [86] is applied to the solution of the Toeplitz equations, then the computational load of the SOBAF is $O(\textit{logN})$, (4.4.4) and (4.4.5).

In Figure 4.2 and Figure 4.3 the computational requirements of two FFT based SOBAF implementations are compared with an LMS algorithm and an FFT based BLMS algorithm. Both an implementation using the Levinson recursion and one using the fast algorithm of [86] are considered. These graphs clearly illustrate the dramatic reduction in computational load that can be achieved when the fast inversion technique

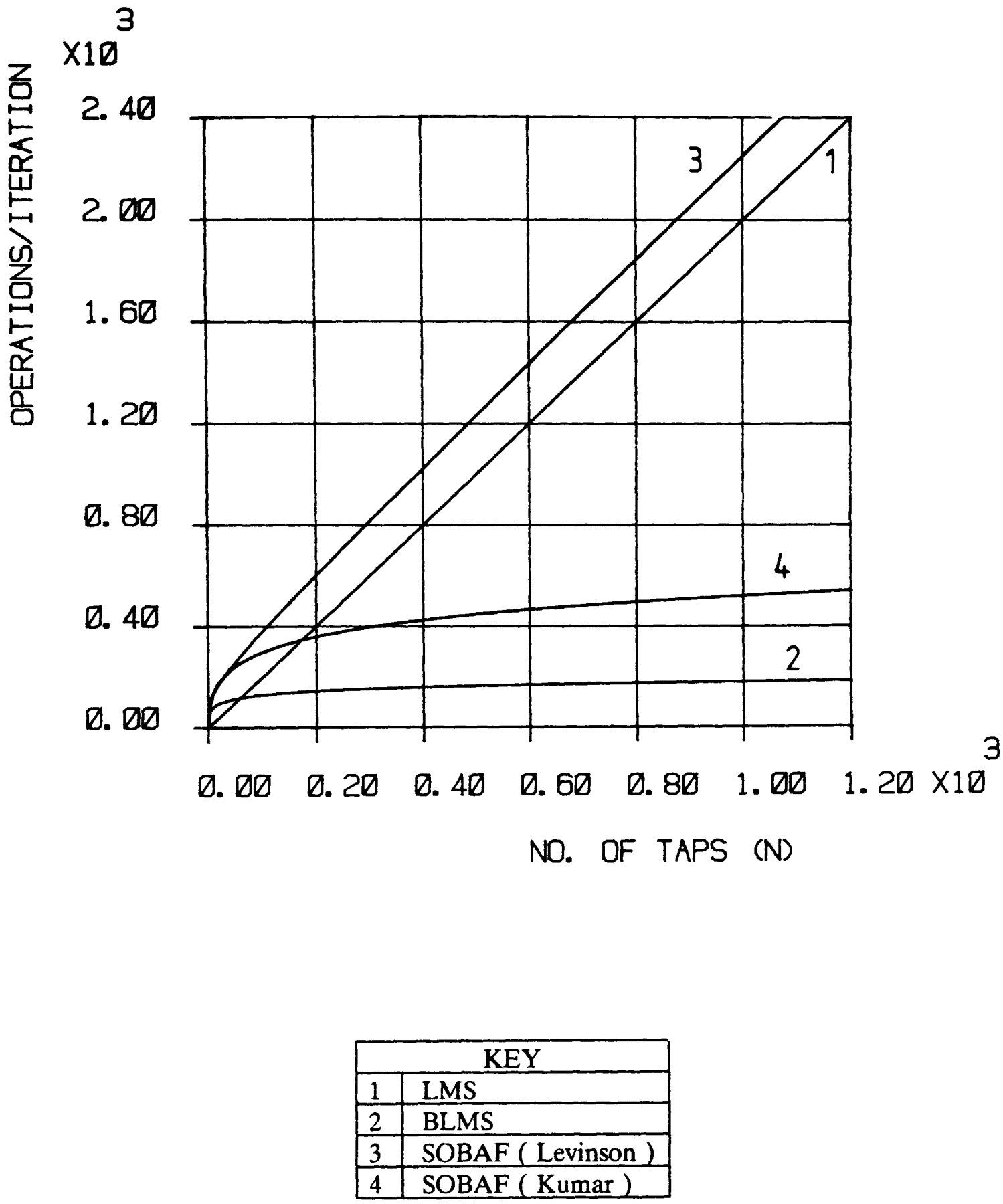
is employed in the SOBAF. Thus, in common with the BLMS algorithm, the SOBAF can exhibit a significant decrease in computational load with respect to the LMS algorithm for moderate to large N.

Figure 4.2 COMPARISON OF FFT-BASED BLOCK ADAPTIVE FILTERS
(Multiplications)



KEY	
1	LMS
2	BLMS
3	SOBAF (Levinson)
4	SOBAF (Kumar)

Figure 4.3 COMPARISON OF FFT-BASED BLOCK ADAPTIVE FILTERS
(Additions)



KEY	
1	LMS
2	BLMS
3	SOBAF (Levinson)
4	SOBAF (Kumar)

4.5 SIMULATION RESULTS

The performance of the SOBAF, summarised in Table 4.1, was examined using the equaliser scenario described in section 3.3 and illustrated in Figure 3.3. As in chapter 3 the FIR channels of Table 3.1 where used in order to vary the maximum/minimum eigenvalue ratio of the autocorrelation matrix, Φ_{xx}. For these experiments a 16-tap transversal equaliser was used, the signal/noise ratio was set at 35dB, the step size μ_b was $\frac{1}{6}$ and the MSE was calculated from an ensemble of 25 runs. The BLMS algorithm of subsection 2.5.2 was used as a reference against which to compare the performance of the SOBAF. The results are illustrated in Figure 4.4 and Figure 4.5.

As the eigenvalue ratio of the input autocorrelation matrix is increased from 11.8 to 68.6 the performance of the BLMS algorithm gets poorer. The performance of the SOBAF, on the other hand, changes very little as the eigenvalue ratio is increased. Comparison of Figures 4.5 and 4.1 indicates that the MSE convergence performance of the SOBAF of Table 4.1 is well predicted by the theoretical result summarised in (4.2.6) and is very similar to the exact algorithm of section 4.2 which uses a-priori knowledge of the autocorrelation matrix. It is clear from Figures 4.4 and 4.5 that the SOBAF is insensitive to the eigenvalue spread of the input autocorrelation matrix and has a MSE convergence performance equivalent to a BLMS algorithm under white input conditions.

Figure 4.4 SIMULATION RESULTS (Channel Equalisation)

eigenvalue ratio	=	11.8	(channel no. 2)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	

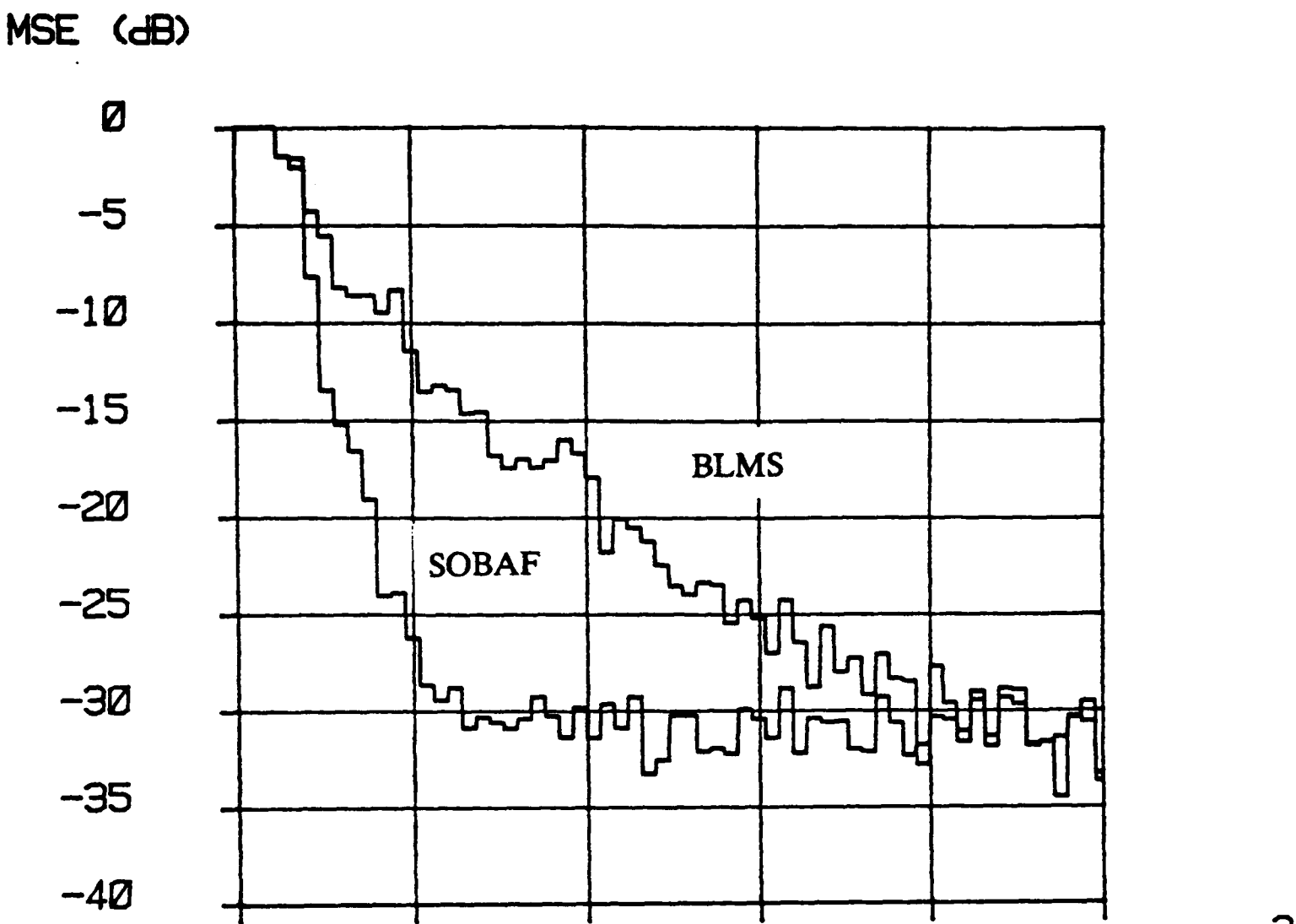
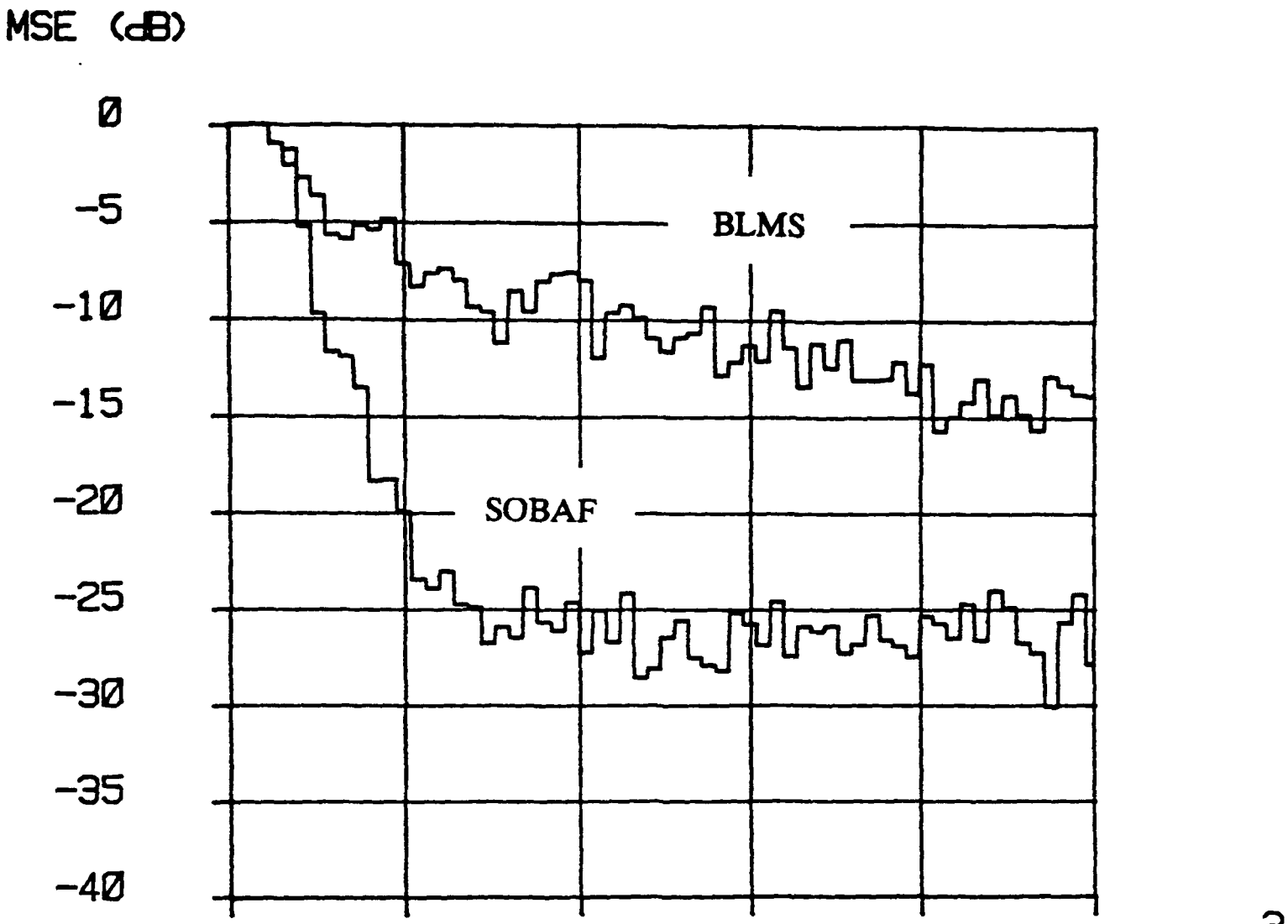


Figure 4.5 SIMULATION RESULTS (Channel Equalisation)

eigenvalue ratio	=	68.6 ·	(channel no. 3)
additive noise	=	-35.0	dB
no. of taps	=	16	
ensemble	=	25	



4.6 CONCLUSIONS

The SOBAF of Table 4.1 is a unique adaptive filter algorithm. In computational load, an FFT based SOBAF is superior to an LMS algorithm for moderate to large N, being of the same order as a BLMS algorithm ie. $O(\ logN\ )$. The SOBAF is thus a very efficient algorithm, computationally. The block nature of the SOBAF also permits the use of other efficient circular convolution algorithms such as the RT and the NTT. In performance, the SOBAF achieves the MSE convergence of a self orthogonalised structure, ie. the adaptive filter converges under any input conditions at the same rate as it would if the input was white. Further, the selection of the step size μ_b is more straightforward than for LMS and BLMS algorithms. This is because both the range of μ_b that ensures MSE convergence and the value of μ_b for fastest convergence are independent of the input autocorrelation matrix. In fact, for a given application, the approximate performance of the algorithm is easily predicted a-priori from (4.2.6).

Chapter 5

THE INFINITE IMPULSE RESPONSE LINEAR EQUALISER

5.1 INTRODUCTION

In many adaptive filtering problems, solutions that use purely FIR filters can provide acceptable performance [73, 8, 90]. Indeed FIR filters are generally to be preferred as they are unconditionally stable and because of the wide selection of well understood adaptive FIR filter algorithms that are available, cf. chapter 2. However these FIR realisations suffer from problems of indeterminate order when it is necessary to model transfer function poles. In particular, when the poles of transfer function are close to the unit circle in the z-plane, a FIR filter of high order may be required to meet a particular performance goal [91]. The obvious alternative has been the adoption of adaptive IIR filters.

Adaptive IIR filtering is a less mature, less well-understood subject than adaptive FIR filtering, witness recent textbooks such as [15, 16, 92] that are devoted almost exclusively to adaptive FIR filtering in comparison with a review article such as [93] that chronicles the current state of knowledge and major open issues to be resolved in adaptive IIR filtering. In the light of this situation only one specific application of adaptive IIR filtering is considered in this chapter rather than a broad comparison such as that which was presented in chapters 2 and 3 for adaptive FIR filters. The particular application is an adaptive IIR linear equaliser whose function is to mitigate the effects of intersymbol interference on a digital communications channel.

The IIR equaliser has received little attention in recent years due to the development of the decision feedback equaliser (DFE) [94]. However, although the DFE has superior MSE performance compared to the IIR equaliser, the latter has an inherent advantage in that it does not utilise previous decisions in forming an estimate

of the transmitted symbol and hence, unlike the DFE [95], will not propagate decisions errors.

This chapter is organised in the following manner. In section 5.2, the linear equaliser is defined, a closed form solution to the optimum IIR equaliser is derived and the structure of the IIR equaliser is investigated. In section 5.3, the MSE performance of IIR and FIR linear equalisers is compared. Finally in section 5.4, the adaptive IIR equaliser is investigated and reformulated as a system identification problem. Several candidate adaptive IIR solutions are described.

5.2 THE LINEAR EQUALISER

A digital communications channel with intersymbol interference may be modelled by an equivalent discrete time transversal filter with additive white noise [9]. Thus the channel output x(k) may be written in terms of the channel input s(k) and the noise n(k) as,

$$x(k) = \underline{h}^T \underline{s}(k) + n(k) \tag{5.2.1}$$

where $\underline{h}$ is the M point impulse response vector

$$\underline{h}^T = [\, h_0 \, h_1 \, \cdots \, h_{M-1} \,]$$

and the vector $\underline{s}(k)$ contains the last M inputs to the channel.

$$\underline{s}^T(k) = [\, s(k) \, s(k-1) \, \cdots \, s(k-M+1) \,]$$

One possible equaliser structure is illustrated in Figure 5.1. This equaliser consists of a linear filter section followed by a non-linear slicer or decision circuit. The linear filter is designed to minimise the error between the filter output and the input to the channel. A MSE cost function, L, is usually used.

$$L = E[\, (\, u(k) - \hat{u}(k) \,)^2 \,]$$

where

$$u(k) = s(k-d)$$

and

$$\hat{u}(k) = \hat{s}(k-d) \ .$$

Since the filter is linear, $\hat{s}(k-d)$ is a linear estimate of s(k-d). The delay term d, $d \geq 0$, allows for the possibility of fixed lag smoothing. The non-linear slicer makes decisions on a symbol by symbol basis. Thus for binary pulse code modulation where s(k) may be either +1 or -1 the slicer output, m(k), is defined by:

$$m(k) = +1 \quad if\ \hat{s}(k) \geq 0$$

$$m(k) = -1 \quad if\ \hat{s}(k) < 0$$

From the above it is clear that the major design effort for this form of equaliser is concentrated on the linear filter section, where linear estimation theory is applied with a view to minimising the mean-square error L at the input to the decision circuit. Hence this structure is described as a linear equaliser.

Usually the linear filter takes the form of a finite impulse response (FIR) transversal filter of order N-1 [9].

$$\hat{u}(k) = \underline{c}^T \underline{x}(k)$$

where

$$\underline{c}^T = [\, c_0\ c_1\ \cdots\ c_{N-1} \,]$$

and

$$\underline{x}^T(k) = [\, x(k)\ x(k-1)\ \cdots\ x(k-N+1) \,]$$

$$d \leq N - 1$$

Under these conditions and provided the processes s(k) and x(k) are jointly stationary then the tap vector $\underline{c}_{opt}$ which minimises the mean-square error L is given by the solution of the Wiener equation.

$$\underline{\Phi}_{xx}\ \underline{c}_{opt} = \underline{\Phi}_{ux} \tag{5.2.2}$$

where

$$\underline{\Phi}_{xx} = E[\ \underline{x}(k)\ \underline{x}^T(k)\ ]$$

and

$$\underline{\Phi}_{ux} = E[\ \underline{x}(k)\ u(k)\ ]$$

The MMSE, L_F, that can be achieved using a FIR filter of order N-1 is

$$L_F = E[\ s^2(k)\ ] - \underline{c}_{opt}^T\ \underline{r}_{ux}\ . \tag{5.2.3}$$

The FIR filter is extensively used in practical linear equalisers because it is unconditionally stable and because of the existence of many adaptive filter algorithms for the calculation of $\underline{c}_{opt}$ when the channel impulse response vector $\underline{h}$ is unknown.

In deriving the optimum transversal equaliser the MSE, L, is minimised subject to the constraint that the impulse response is finite, causal and stable. If this condition is replaced with a less stringent one, i.e. the filter should be causal and stable, the solution to the minimisation problem is provided by the infinite impulse response (IIR) Wiener filter. The optimum IIR filter is defined in terms of the z transform of its impulse response sequence $\{\ g(k)\ \}$ [96].

$$G(z) = Z\{\ g(k)\ \}$$

$$= \sum_{k=-\infty}^{+\infty} g(k)\ z^{-k}$$

$$= \frac{1}{R_{xx}^{+}(z)} \left[\frac{R_{ux}(z)}{R_{xx}^{-}(z)} \right]_{+} \tag{5.2.4}$$

where

$$R_{xx}(z) = Z\{ \Phi_{xx}(l) \}$$

$$\Phi_{xx}(l) = E[x(k+l)\, x(k)]$$

$$R_{ux}(z) = Z\{ \Phi_{ux}(l) \}$$

$$\Phi_{ux}(l) = E[u(k+l)\, x(k)]$$

The power spectrum $R_{xx}(z)$ is factorised as follows:

$$R_{xx}(z) = R_{xx}^{+}(z)\, R_{xx}^{-}(z)$$

$R_{xx}^{+}(z)$ has all poles and zeros inside the unit circle in the z-plane. $R_{xx}^{-}(z)$ has all poles and zeros outside the unit circle. The notation $[.]_{+}$ represents the causal part of [.] i.e.

$$\left[\frac{R_{ux}(z)}{R_{xx}^{-}(z)} \right] = \left[\frac{R_{ux}(z)}{R_{xx}^{-}(z)} \right]_{+} + \left[\frac{R_{ux}(z)}{R_{xx}^{-}(z)} \right]_{-}$$

All the poles of $[.]_{+}$ are inside the unit circle and all the poles of $[.]_{-}$ are outside the unit circle. The optimum IIR Wiener equalising filter, assuming a white input signal s(k), can be written down from (5.2.4).

$$G(z) = \frac{1}{\{ H(z)\, H(z^{-1})\, \sigma_s^2 + \sigma_n^2 \}^{+}} \left[\frac{z^{-d}\, H(z^{-1})\, \sigma_s^2}{\{ H(z)\, H(z^{-1})\, \sigma_s^2 + \sigma_n^2 \}^{-}} \right]_{+} \qquad (5.2.5)$$

where

$$H(z) = Z\{ h_k \} = \sum_{k=0}^{M-1} h_k\, z^{-k} \qquad (5.2.6)$$

and

$$E[\, n(k)\, n(k+l)\,] = \sigma_n^2\, \delta(l)$$

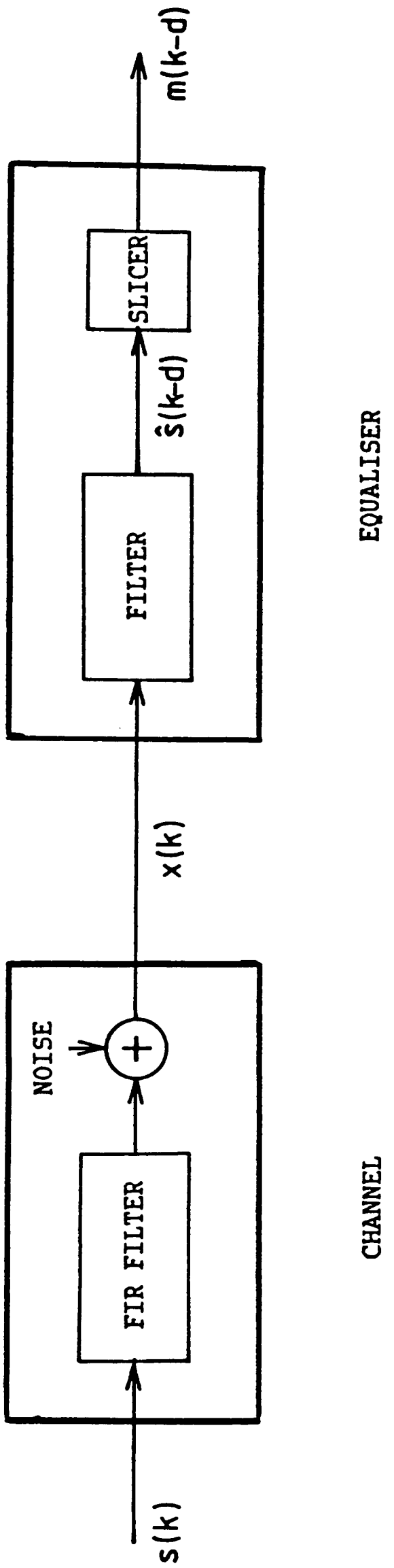
$$E[\, s(k)\, s(k+l)\,] = \sigma_s^2\, \delta(l)$$

$\delta(l)$ is the Dirac delta function. The MMSE L_I that can be achieved by use of the IIR Wiener filter is obtained by use of the orthogonality principle [96].

$$\begin{aligned} L_I &= E[\, (\, u(k) - \hat{u}(k)\,)^2\,] \\ &= E[\, (\, u(k) - \hat{u}(k)\,)\, u(k)\,] \\ &= E[\, u^2(k)\,] - E[\, \hat{u}(k)\, u(k)\,] \\ &= \sigma_s^2 \left[\, 1 - \sum_{i=0}^{+\infty} g(i)\, h(d-i) \,\right] \end{aligned} \tag{5.2.7}$$

The summation is the convolution of the channel impulse response and the equaliser impulse response evaluated at sample d.

Figure 5.1 THE LINEAR EQUALISER



5.2.1 Structure of an IIR Equaliser

The power spectral density, $R_{xx}(z)$, of the channel output sequence, $\{ x(n) \}$, is

$$R_{xx}(z) = H(z)\, H(z^{-1})\, \sigma_s^2 + \sigma_n^2$$

which is a polynomial of degree $2M-1$. Since the autocorrelation function, $\Phi_{xx}(l)$, is symmetric about the origin ie. $\Phi_{xx}(l) = \Phi_{xx}(-l)$, the zeros of $R_{xx}(z)$ have symmetry. In particular if $R_{xx}(z)$ is zero at $z = z_i$ then it will also be zero at $z = z_i^{-1}$. Thus $R_{xx}(z)$ may be factorised in terms of a polynomial P(z) which is of degree M-1 and has all its zeros inside the unit circle ie.

$$R_{xx}(z) = P(z)\, P(z^{-1})$$

where

$$P(z) = \sum_{k=0}^{M-1} p_k\, z^{-k} \ .$$

$P(z^{-1})$ is a polynomial of degree M-1 which has all its zeros outside the unit circle. The Wiener IIR equaliser of (5.2.5) may be rewritten

$$G(z) = W(z)\, [\, B(z)\,]_+$$

where

$$W(z) = \frac{1}{P(z)}$$

and

$$B(z) = \frac{z^{-d}\, H(z^{-1})}{P(z^{-1})} \ .$$

Since the transfer function P(z) is a minimum phase FIR filter, of order M-1, W(z) is a stable autoregressive filter of order M-1 . The effect of the filter, W(z), is to whiten the output sequence, $\{ x(n) \}$ [96].

The nature of $[\, B(z)\,]_+$ is now examined. By definition :

$$B(z) = \sum_{k=-\infty}^{+\infty} b(k)\, z^{-k}$$

$$= \sum_{k=0}^{+\infty} b(k)\, z^{-k} + \sum_{k=-\infty}^{-1} b(k)\, z^{-k}$$

$$= [\, B(z)\,]_+ + [\, B(z)\,]_-$$

B(z) has d poles at the origin and M-1 poles outside the unit circle. In order to ensure the convergence of both the causal part , $[\, B(z)\,]_+$, and the anticausal part , $[\, B(z)\,]_-$, of the series the region of convergence must be [7].

$$0 \leq |\, z\, | < 1$$

Therefore the causal part of b(k), $k \geq 0$, is given by

$$b(k) = \Sigma[\ \textit{residues of}\ B(z)z^{k-1}\ \textit{at the poles at the origin}\ ]$$

This is abbreviated to

$$b(k) = \Sigma[\ \textit{res.}\ B(z)z^{k-1}\ \textit{at}\ O\ ]$$

where O is the origin in the z-plane. The z-transform of the causal part can thus be written,

$$[\, B(z)\,]_+ = \sum_{k=0}^{+\infty} \Sigma[\ \textit{res.}\ B(z)z^{k-1}\ \textit{at}\ O\ ]z^{-k}$$

$B(z)z^{k-1}$ has poles at the origin provided $k \leq d$. Therefore

$$[\, B(z)\,]_+ = \sum_{k=0}^{d} \Sigma[\ \textit{res.}\ B(z)z^{k-1}\ \textit{at}\ O\ ]z^{-k}$$

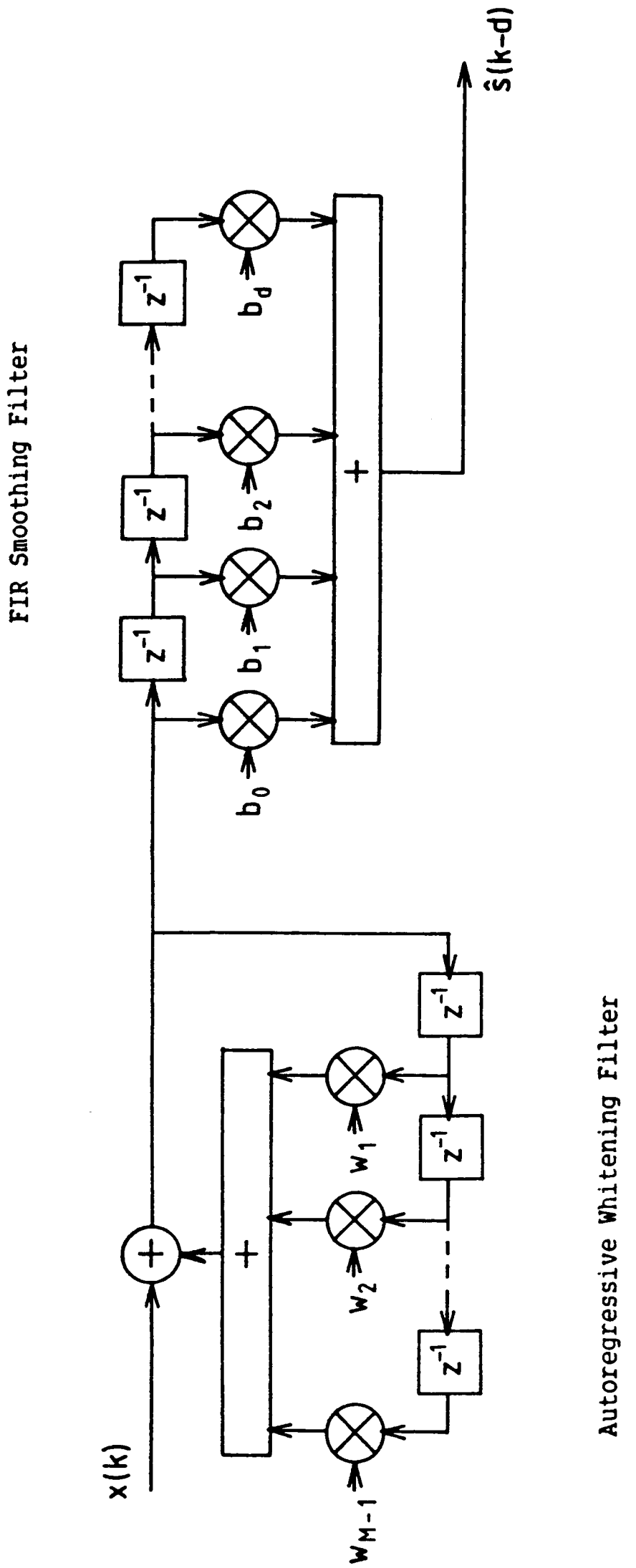
The transfer function $[\, B(z)\,]_+$ represents a FIR filter of order d. The Wiener IIR equaliser is thus the cascade of an autoregressive whitening filter of order M-1, W(z), and a FIR filter of order d. This arrangement is illustrated in Figure 5.2.

In order to highlight the effect of the channel phase characteristics on the structure of the Wiener IIR equaliser, consider an example where the signal/noise ratio is high ie. $\sigma_s^2 >> \sigma_n^2$, and the signal power, σ_s^2, is normalised to unity. When all the zeros of $H(z)$ are inside the unit circle the channel is minimum phase and the Wiener IIR equaliser is given approximately by,

$$G(z) = \frac{1}{H(z)} z^{-d}$$

In which case the estimate with lag d, $\hat{s}(k-d)$, is merely the estimate with lag zero, $\hat{s}(k)$, delayed by d samples and there is no advantage in using a fixed lag estimate of the channel input when an estimate of the current input can be obtained without loss of quality. If however the channel is non-minimum phase the Wiener filter in the absence of noise is still the cascade of an autoregressive whitening filter of order M-1 and a FIR filter of order d and hence the MSE performance will improve with increasing lag d [97].

Figure 5.2 THE WIENER IIR EQUALISER



5.3 COMPARISON OF FIR AND IIR EQUALISER PERFORMANCE

The IIR Wiener filter is the best linear unbiased estimator of the channel input s(k-d) in a MSE sense under steady state conditions [5]. The FIR Wiener filter must then be regarded as an approximation to the IIR filter and be expected to perform less well than an IIR filter of the same order. It should be noted however that while it is possible to construct a FIR filter of any order, the minimum order of the IIR filter is constrained by the order of the autoregressive whitening filter, W(z), and hence by the order of the channel to M-1. If the channel impulse response vector $\underline{h}$ is known, the MMSE, L_F, that can be obtained by a FIR filter is calculated by constructing Φ_{xx} and Φ_{ux} and then using (5.2.2) and (5.2.3). If the order of the FIR filter is fixed at (N-1), the MMSE L_F varies with lag l, $0 \leq l \leq N-1$. The value of the lag l which produces the minimum value of L_F for a given order of FIR filter is known as the optimum lag [98]. The IIR Wiener filter for a known channel may be constructed using (5.2.5) and the MMSE, L_I, calculated from (5.2.7).

A comparison of the MSE performance of the FIR and IIR filters as a function of the filter order is illustrated in Figures 5.3 to 5.5. For the sake of this comparison the FIR filter with optimum lag for a given order was chosen. Thus the MSE for the FIR filter is a lower bound on the performance that would be expected in a practical situation. The three channels that were used are summarised in Table 5.1. For channel 3, which is minimum phase, the MSE (Figure 5.3) does not vary with the order of the IIR filter and is only limited by the noise floor. This is consistent with the remarks made in subsection 5.2.1 on the performance of an IIR filter in the absence of noise when the channel is minimum phase i.e. the quality of the estimate is not a function of the lag. The order of the FIR filter increases linearly with decrease in MSE. The lowest MSE that is achievable is the same as that for the IIR filter. For channel number 1, which is non-minimum phase, the MSE (Figure 5.4) decreases linearly with the order of the IIR filter until it reaches a level just above the noise

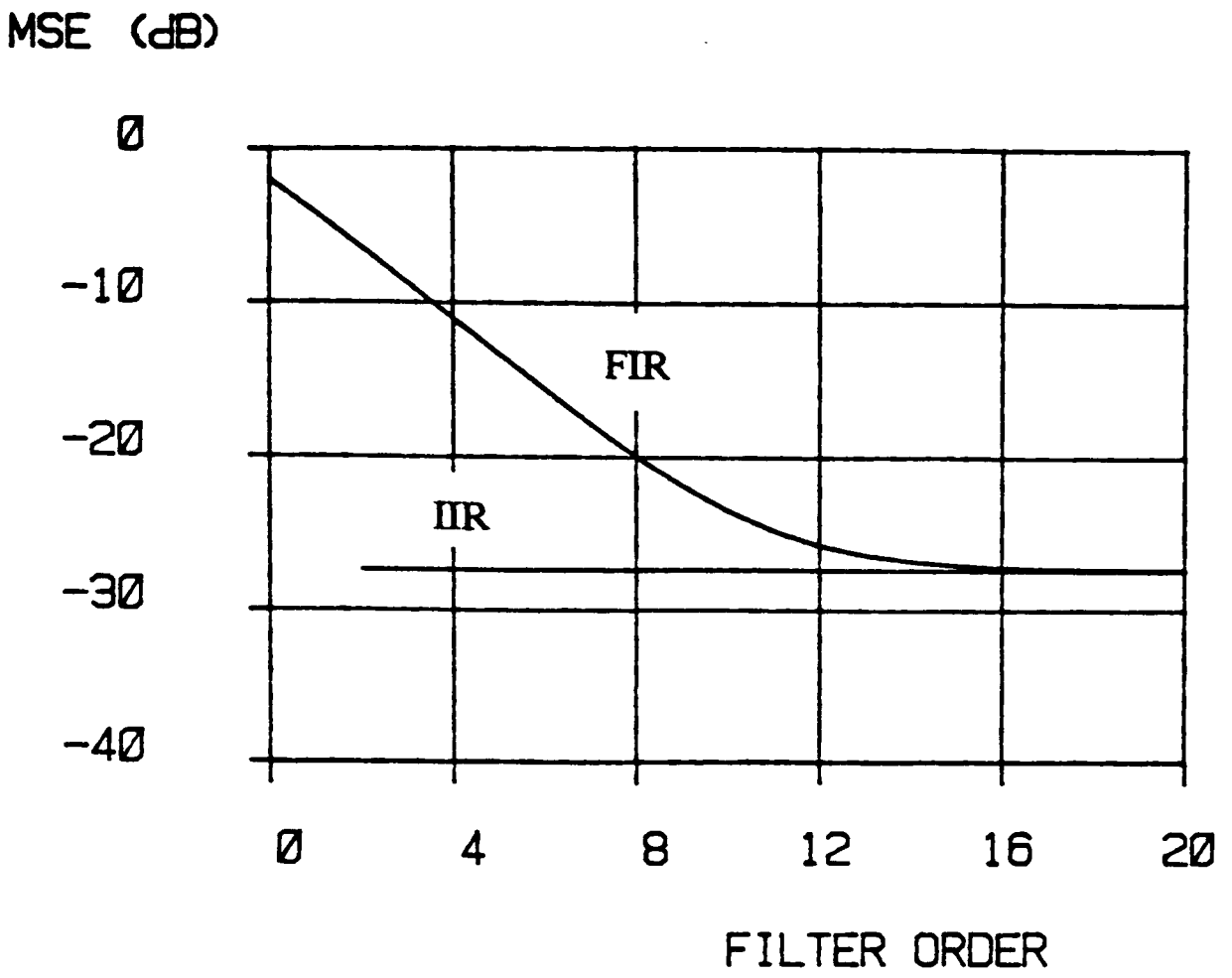
floor. Again this is consistent with the remarks made in subsection 5.2.1 on the performance of an IIR filter in the absence of noise when the channel is non-minimum phase i.e. the MSE is a function of lag. The FIR filter exhibits the same general trends for this channel as the IIR filter in that the MSE decreases linearly with an increase in filter order until a level just above the noise floor is reached. However it should be noted that for a particular MSE value the order of the IIR filter is always less than the order of the FIR filter. The results for channel number 2, which is also non-minimum phase, are similar to those obtained for channel number 1 (Figure 5.5).

In conclusion, these results indicate that an IIR filter outperforms a FIR filter of the same order in a MSE sense. Two particular cases can be identified. When the channel is minimum phase, an IIR filter of the same order as the channel produces the lowest MSE that is achievable with a linear filter. A FIR filter of order greater than the channel order is required to obtain the same result. When the channel is non-minimum phase, the performance of an IIR filter continues to improve when the order is increased beyond the order of the channel until a level just above the noise floor is obtained. The FIR filter exhibits similar characteristics but the initial rate of improvement in performance with increasing order is significantly smaller than that for the IIR filter.

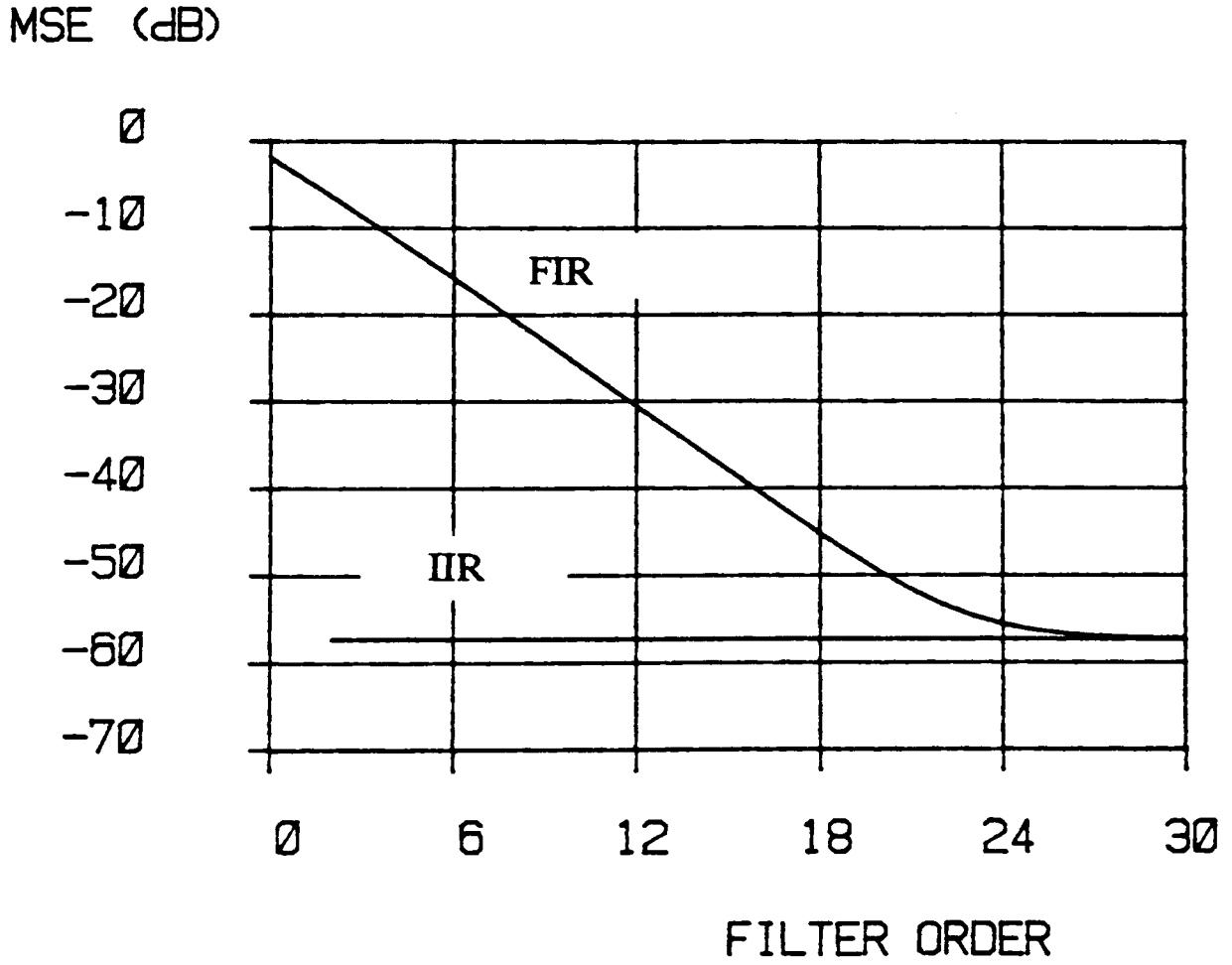
Table 5.1 CHANNEL IMPULSE RESPONSES

Channel No.	Impulse Response	Classification
1	$0.2602 + 0.9298z^{-1} + 0.2602z^{-2}$	non-minimum phase
2	$0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$	non-minimum phase
3	$0.6082 + 0.7603z^{-1} + 0.2280z^{-2}$	minimum phase

Figure 5.3 FIR/IIR EQUALISER PERFORMANCE COMPARISON
minimum phase (channel no. 3)

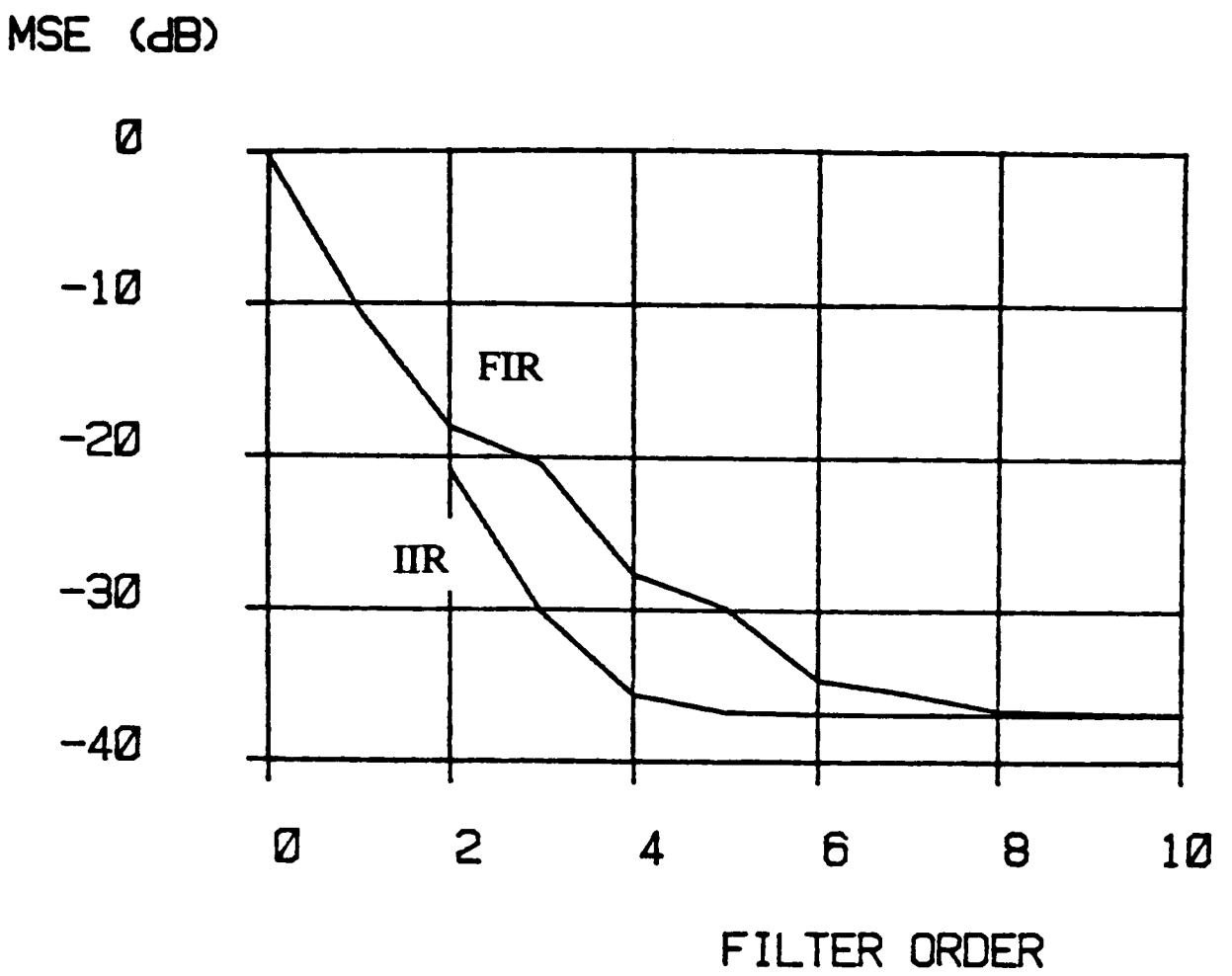


(a) additive noise = -40.0 dB

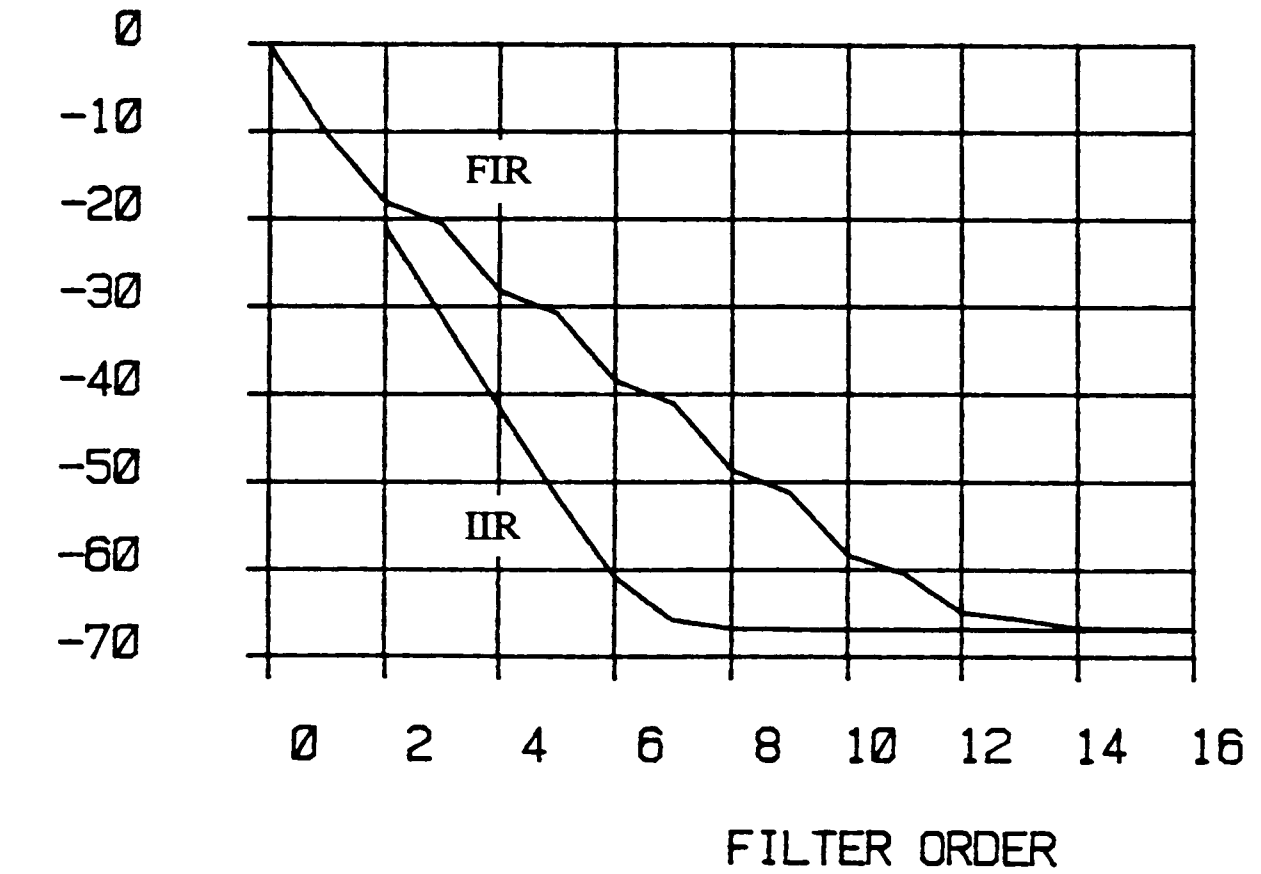


(b) additive noise = -70.0 dB

Figure 5.4 FIR/IIR EQUALISER PERFORMANCE COMPARISON
non - minimum phase (channel no. 1)

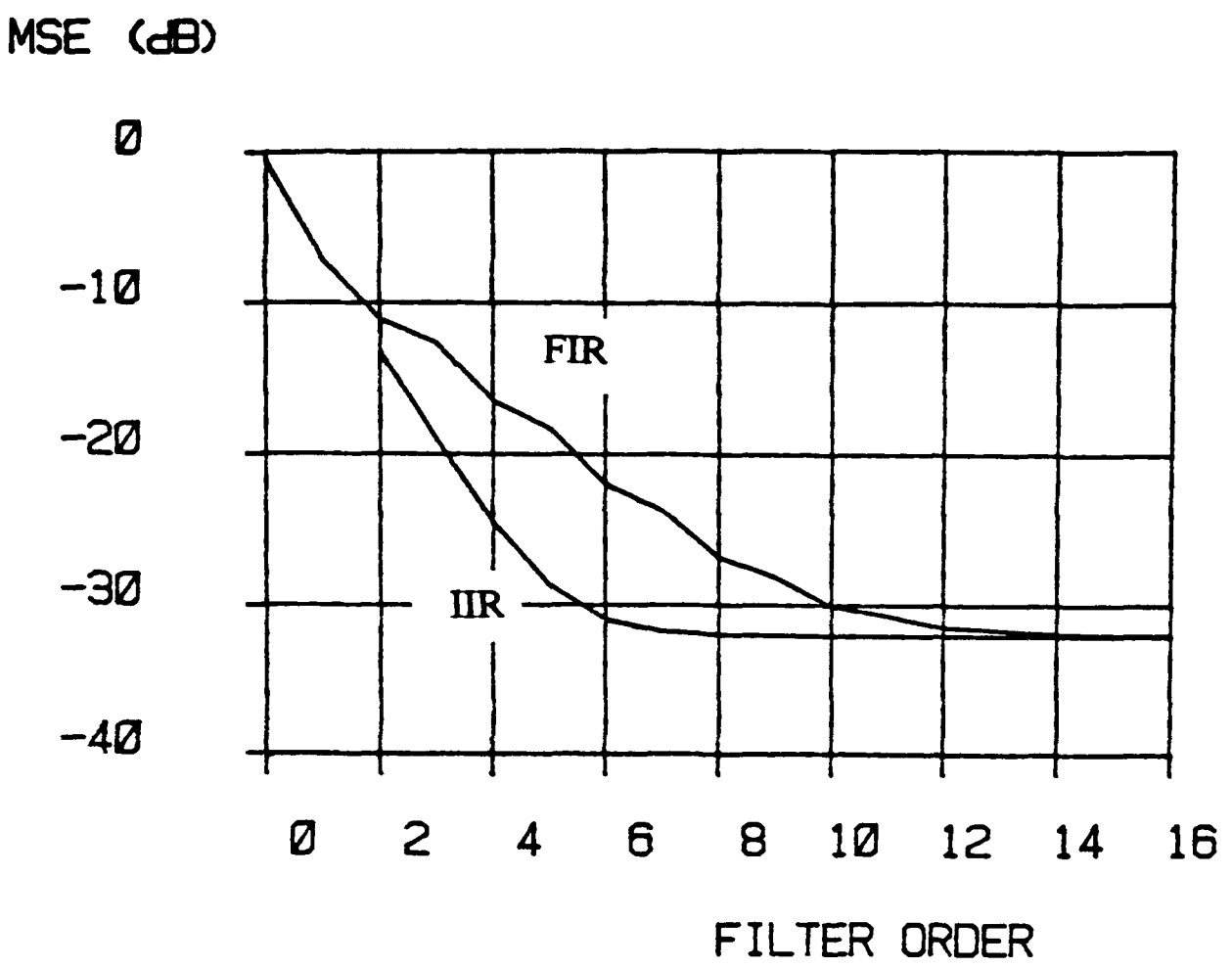


(a) additive noise = -40.0 dB

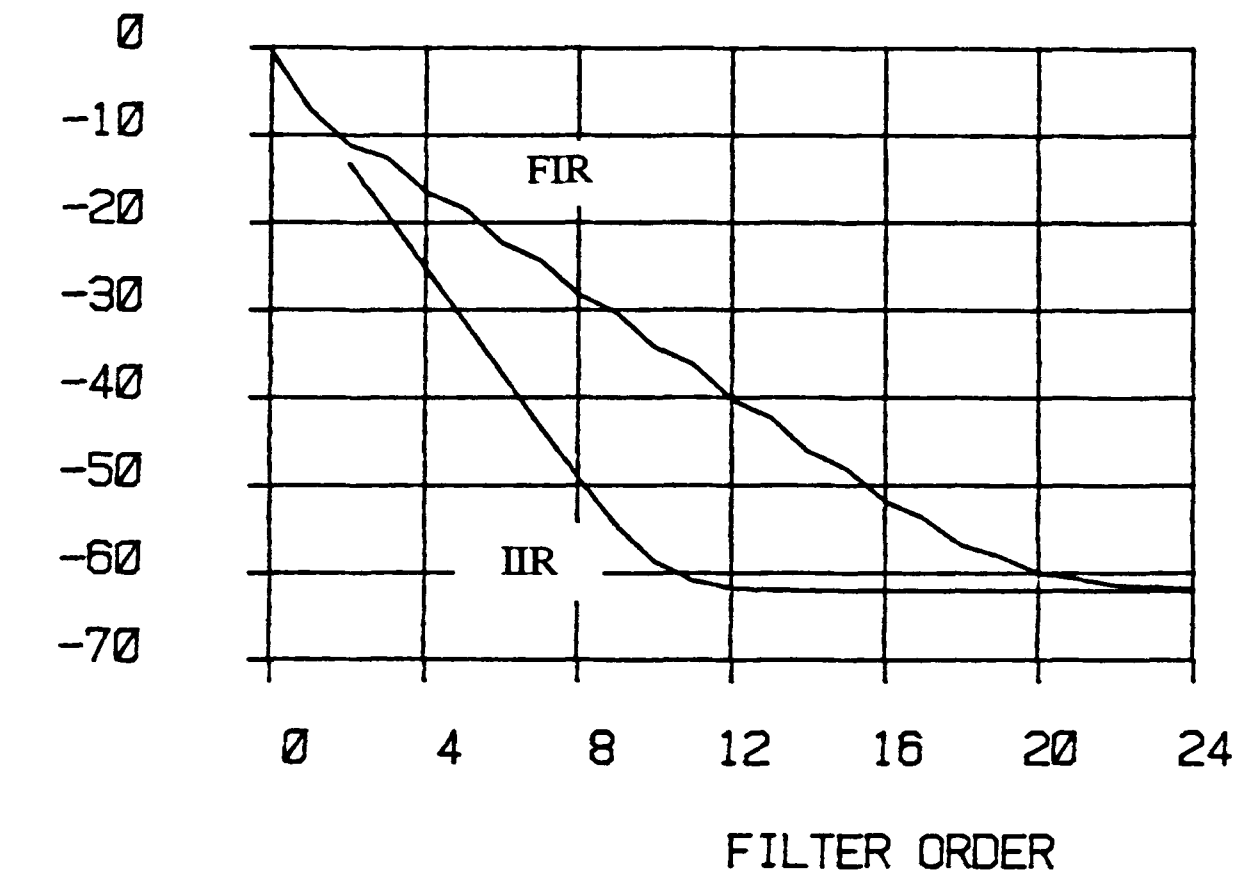


(b) additive noise = -70.0 dB

Figure 5.5 FIR/IIR EQUALISER PERFORMANCE COMPARISON
non - minimum phase (channel no. 2)



(a) additive noise = -40.0 dB



(b) additive noise = -70.0 dB

5.4 SYSTEM IDENTIFICATION

Given the considerations of subsection 5.2.1, the optimum IIR equaliser defined in (5.2.5) is an ARMA filter whose output $s_{opt}(k-d)$ may be expressed in terms of M-1 previous outputs and d inputs, ie.

$$s_{opt}(k-d) = \sum_{i=1}^{M-1} \alpha_i \; s_{opt}(k-d-i) + \sum_{j=0}^{d} \beta_j \; x(k-j) \tag{5.4.1}$$

where the coefficients α_i and β_j are chosen to minimise the MSE,

$$E[\,(\,s(k-d) - s_{opt}(k-d)\,)^2\,]\,,$$

and are defined by (5.2.5). Thus the minimum variance error sequence $\{\, e_{opt}(k)\,\}$ is given by

$$e_{opt}(k) = s(k-d) - s_{opt}(k-d)$$

which when combined with (5.4.1) provides an expression for the observable sequence $\{\, s(k-d)\,\}$ in terms of the observable sequence $\{\, x(k)\,\}$ and an unobservable noise sequence $\{\, v(k)\,\}$.

$$s(k-d) = \sum_{i=1}^{M-1} \alpha_i \; s(k-d-i) + \sum_{j=0}^{d} \beta_j \; x(k-j) + v(k) \tag{5.4.2}$$

The additive noise term v(k) is a filtered version of $e_{opt}(k)$.

$$v(k) = \sum_{i=1}^{M-1} \alpha_i \; e_{opt}(k-i) + e_{opt}(k) \tag{5.4.3}$$

Together (5.4.2) and (5.4.3) define a autoregressive moving average with exogeneous input (ARMAX) model. Thus the adaptive IIR equaliser problem may be re-interpreted as an ARMAX system identification problem, where a recursive estimate of the coefficients α_i and β_j is required from an observable input sequence $\{\, x(i)\,\}$ and an observable output sequence $\{\, s(i-d)\,\}$.

5.4.1 Adaptive IIR Solutions

Adaptive solution to this system identification problem are obtained by rewriting (5.4.3) as a vector inner product of the aggregate vectors $\underline{x}(k)$ and $\underline{\theta}_{opt}$

$$s(k-d) = \underline{\theta}_{opt}^T \underline{x}(k) + v(k) , \tag{5.4.4}$$

where

$$\underline{x}(k) = [\, s(k-d-i) \;\cdots\; s(k-d-M+1)\; x(k) \;\cdots\; x(k-d) \,]^T$$

and

$$\underline{\theta}_{opt} = [\, \alpha_1 \;\cdots\; \alpha_{M-1}\; \beta_0 \;\cdots\; \beta_d \,]^T .$$

Since (5.4.4) is similar in form to the FIR system of (3.3), it is tempting initially to apply an adaptive FIR filter algorithm such as those described in chapter 2 in order to estimate $\underline{\theta}_{opt}$. The first choice adaptive FIR filter algorithm is the RLS of subsection 2.4, since, from chapter 3, it is clear that it exhibits the best convergence properties. The resultant ARMA system identification algorithm is summarised in the following 4 equations which are identical in form to (2.4.12) - (2.4.15).

$$\underline{\theta}(k) = \underline{\theta}(k-1) + \underline{k}(k)\, e(k) \tag{5.4.5}$$

$$e(k) = s(k-d) - \underline{\theta}^T(k-1)\, \underline{x}(k) \tag{5.4.6}$$

$$\underline{k}(k) = \underline{r}_{xx}^{-1}(k)\, \underline{x}(k) \tag{5.4.7}$$

$$\underline{r}_{xx}^{-1}(k) = \underline{r}_{xx}^{-1}(k-1) - \frac{\underline{r}_{xx}^{-1}(k-1)\, \underline{x}(k)\, \underline{x}^T(k)\, \underline{r}_{xx}^{-1}(k-1)}{\left\{ 1 + \underline{x}^T(k)\, \underline{r}_{xx}^{-1}(k-1)\, \underline{x}(k) \right\}} \tag{5.4.8}$$

The vector $\underline{\theta}(k)$ is the current estimate of $\underline{\theta}_{opt}$. However unlike the system identification problem of subsection 3.2, the additive noise term in (5.4.4) is not white even if it could be assumed that $e_{opt}(k)$ was white. Under these non-white noise

conditions the RLS algorithm will produce an asymmtotically biased estimate of the coefficients of the optimum IIR equaliser, $\underline{\theta}_{opt}$ [24]. For the equaliser scenario this biased estimate is recognisable as the decision feedback equaliser (DFE) [9] with output

$$s_{dfe}(k-d) = \sum_{i=1}^{M-1} \gamma_i \, s(k-d-i) + \sum_{j=0}^{d} \delta_j \, x(k-j)$$

where the coefficients γ_i and δ_j are chosen to minimise the MSE

$$E[\,(s(k-d) - s_{dfe}(k-d))^2\,] \,.$$

In order to find the linear IIR equaliser the additive noise term $v(k)$ must be whitened. This can be done approximately by replacing $e_{opt}(k)$ in the summation of (5.4.3) with the instantaneous error e(k) [99] where

$$e(k) = s(k-d) - \hat{s}_{opt}(k-d)$$

$\hat{s}_{opt}(k-d)$ is the current estimate of $s_{opt}(k-d)$. Equation (5.4.2) then becomes

$$s(k-d) = \sum_{i=1}^{M-1} \alpha_i \, \hat{s}_{opt}(k-d-i) + \sum_{j=0}^{d} \beta_j \, x(k-j) + e_{opt}(k)$$

$$= \underline{\theta}_{opt}^T \, \underline{\hat{x}}(k) + e_{opt}(k) \,, \qquad (5.4.9)$$

where

$$\underline{\hat{x}}(k) = [\,\hat{s}_{opt}(k-d-1) \;\cdots\; \hat{s}_{opt}(k-d-M+1)\; x(k) \;\cdots\; x(k-d)\,]^T \,.$$

If the RLS algorithm is applied to (5.4.9) to form an estimate of $\underline{\theta}_{opt}$, the result is an extended least squares (ELS) [99] adaptive equaliser algorithm which is summarised by the following 4 equations.

$$\underline{\theta}(k) = \underline{\theta}(k-1) + \underline{k}(k)\, e(k) \qquad (5.4.10)$$

$$e(k) = s(k-d) - \underline{\theta}^T(k-1)\, \underline{\hat{x}}(k) \qquad (5.4.11)$$

$$\underline{k}(k) = \underline{r}_{xx}^{-1}(k)\,\hat{\underline{x}}(k) \tag{5.4.12}$$

$$\underline{r}_{xx}^{-1}(k) = \underline{r}_{xx}^{-1}(k-1) - \frac{\underline{r}_{xx}^{-1}(k-1)\,\hat{\underline{x}}(k)\,\hat{\underline{x}}^{T}(k)\,\underline{r}_{xx}^{-1}(k-1)}{\left(\,1 + \hat{\underline{x}}^{T}(k)\,\underline{r}_{xx}^{-1}(k-1)\,\hat{\underline{x}}(k)\,\right)} \tag{5.4.13}$$

Convergence of the ELS algorithm is not guaranteed [99].

An alternative approach that can remove the asymtotic bias of the RLS algorithm under coloured noise conditions is the recursive intrumental variable (RIV) algorithm, which again can be summarised by 4 equations.

$$\underline{\theta}(k) = \underline{\theta}(k-1) + \underline{k}(k)\,e(k) \tag{5.4.14}$$

$$e(k) = s(k-d) - \underline{\theta}^{T}(k-1)\,\underline{x}(k) \tag{5.4.15}$$

$$\underline{k}(k) = \underline{r}_{xz}^{-1}(k)\,\underline{z}(k) \tag{5.4.16}$$

$$\underline{r}_{xz}^{-1}(k) = \underline{r}_{xz}^{-1}(k-1) - \frac{\underline{r}_{xz}^{-1}(k-1)\,\underline{z}(k)\,\underline{x}^{T}(k)\,\underline{r}_{xz}^{-1}(k-1)}{\left(\,1 + \underline{x}^{T}(k)\,\underline{r}_{xz}^{-1}(k-1)\,\underline{z}(k)\,\right)} \tag{5.4.17}$$

The vector,

$$\underline{z}(k) = [\; z(k)\; z(k-1)\; \cdots\; z(k-M+1)\; x(k)\; \cdots\; x(k-d)\;]^{T}$$

contains the instrumental variables $\{\, z(n)\,\}$. If the instrumental variables are chosen to meet conditions which are described in [100], the RIV algorithm will produce consistent estimates of the optimum IIR equaliser. Common choices for the instrumental variables are:

$$z(k) = s(k-d-t)$$

where $t > M$ [101] or

$$z(k) = \underline{\theta}^{T}(k)\,\underline{x}(k)$$

5.5 CONCLUSIONS

A closed form expression for the optimum IIR equalising filter was derived using Wiener filtering theory. The optimum IIR filter was shown to be the cascade of an AR whitening filter of order N-1, the order of the FIR channel, and a FIR filter of order d, where d is the estimation lag. A comparison of the MSE performance of FIR and IIR equalisers illustrated the inherent order advantage in using a IIR structure, particularly for minimum phase channels, and higlighted the effect non-minimum phase distortion has on the performance of linear equaliser structures. The adaptive IIR equaliser problem was shown to be equivalent to the identification of an ARMA plant which is embedded in an ARMAX process. Although the bias associated with RLS estimates can be avoided through resort to algorithms such as RIV, the MSE convergence of these algorithms is poorly understood and hence they cannot be considered to be robust solutions.

Chapter 6

AN ADAPTIVE IIR EQUALISER

6.1 INTRODUCTION

While the IIR Wiener filter exhibits a distinct performance advantage over a FIR filter of the same order when used to equalise a known channel, significant problems are encountered with the former when the channel is unknown or time-varying and an adaptive filter structure is required. An initial approach to the problem might be to postulate an adaptive algorithm such as those suggested in [91] that would recursively estimate the coefficients of the IIR Wiener filter in the same manner as the LMS algorithm [21] is used to estimate the coefficients of the FIR Wiener filter. However, in the process of adaptation, there is a finite probability that the poles of the filter will move outside the unit circle in the z-plane. This can lead to instability if the poles remain outside the unit circle for an extended period [91]. As discussed in chapter 5, adaptive IIR filter algorithms do exist whose convergence in a mean sense is assured but few theoretical results are available with which to predict the MSE convergence properties ot these algorithms.

Central to this chapter is the recognition that the optimum Wiener IIR equaliser may be realised using state space concepts ie. the Kalman equaliser of [18]. This formulation simultaneously circumvents the minimum phase spectral factorisation which is integral to the Wiener solution and reduces the number of coefficients which define the equaliser and hence which must be estimated in any adaptive scheme from $M+d+1$ to M. To make the Kalman equaliser adaptive a system identification algorithm is used in parallel with it to estimate the M coefficients of the channel impulse response. This contrasts with the adaptive Kalman equaliser suggested in [18] where the state of the Kalman filter is augmented to include the coefficients of the unknown channel, which results in a non-linear estimation problem to which the

extended Kalman filter (EKF) is applied. However the convergence of the EKF is not guarenteed [102]. Further advantage acrues because the input to the system identification algorithm is the channel input which is usually a white process. Thus the LMS algorithm will provide consistent predictable convergence performance under these conditions. Contrast this with the adaptive FIR equaliser of section 3.3 where the channel output, an unknown non-white process, forms the input to the adaptive FIR filter algorithm.

The discrete time Kalman filter of [3] is defined in section 6.2 and subsequently applied to the channel equalisation problem in section 6.3 to derive the non adaptive smoothing filter of [103], which for a fixed channel and under steady-state conditions is equivalent to the optimum IIR equaliser of section 5.2. In Section 6.4 an adaptive Kalman equaliser structure is presented. This adaptive structure is fundamentally different from those considered in [18, 104] and [105]. In essence it is the combination of the adaptive Kalman equaliser of [106, 107] with the non-adaptive smoothing filter of [103], but unlike [106, 107], the full form of the Kalman gain equations [3] are used in order to exploit the capacity of the Kalman filter to handle nonstationary environments. This combination produces an adaptive structure which is capable of equalising both minimum and non-minimum phase channels. A new technique is presented for both the on-line estimation of the channel noise variance and the compensation of the Kalman filter for the modelling uncertainty inherent in the imperfect knowledge of the channel impulse response. The complete adaptive Kalman equaliser is compared with a RLS FIR equaliser in both performance and computational complexity. Finally in section 6.5 the LMS system identification algorithm is replaced with an RLS algorithm to improve the convergence performance of the adaptive Kalman equaliser.

6.2 THE KALMAN FILTER

With the publication of [3] and [4], Kalman and Bucy defined a powerful recursive estimation technique which has come to be known as the Kalman filter. Application of the Kalman filter assumes that the studied system may be described by a pair of state space equations. These are the state transition equation,

$$\underline{s}(k+1) = \underline{A}(k+1)\,\underline{s}(k) + \underline{w}(k+1)$$

and the observation equation.

$$\underline{z}(k) = \underline{H}(k)\,\underline{s}(k) + \underline{u}(k)$$

The M-vector $\underline{s}(k)$ contains the values of the M parameters which define the state of the system at time k and is thus the state vector. The $(M \times M)$ matrix $\underline{A}(k)$ is the state transistion matrix and the $(L \times M)$ matrix $\underline{H}(k)$ is the observation matrix. The M-vector $\underline{w}$ and the L-vector $\underline{u}$ are uncorrelated zero mean and white with covariances $\underline{W}$ and $\underline{U}$ respectively. Thus

$$E[\,\underline{w}(k)\,\underline{w}^T(k)\,] = \underline{W}(k)$$

$$E[\,\underline{u}(k)\,\underline{u}^T(k)\,] = \underline{U}(k)\,.$$

The system matrices $\underline{A}$, $\underline{W}$, $\underline{H}$ and $\underline{U}$ may be time varying but are assumed to be known a priori. Optimal estimates of the state vector $\underline{s}(k)$ are generated recursively from the sequence of noisy observations $\{\,\underline{z}(k)\,\}$ by use of the following equations. The notation $\hat{\underline{s}}(k/j)$ reads "the estimate of $\underline{s}(k)$ given data from sample 0 to sample j". The estimation equation.

$$\hat{\underline{s}}(k/k) = \hat{\underline{s}}(k/k-1) + \underline{K}(k)\,[\,\underline{z}(k) - \underline{H}(k)\,\hat{\underline{s}}(k/k-1)\,] \qquad (6.2.1)$$

The prediction equation.

$$\hat{\underline{s}}(k/k-1) = \underline{A}(k)\,\hat{\underline{s}}(k-1/k-1) \qquad (6.2.2)$$

The Kalman gain equation.

$$\underline{K}(k) = \underline{V}(k/k-1)\,\underline{H}^T(k)\,[\,\underline{H}(k)\,\underline{V}(k/k-1)\,\underline{H}^T(k) + \underline{U}(k)\,]^{-1} \qquad (6.2.3)$$

The error covariance equations

$$\underline{V}(k/k-1) = \underline{A}(k)\,\underline{V}(k-1/k-1)\,\underline{A}^T(k) + \underline{W}(k) \qquad (6.2.4)$$

and

$$\underline{V}(k-1/k-1) = \underline{V}(k-1/k-2) - \underline{K}(k-1)\,\underline{H}(k-1)\,\underline{V}(k-1/k-2) \qquad (6.2.5)$$

where

$$\underline{V}(k/k) = E[\,(\,\underline{x}(k) - \hat{\underline{x}}(k/k)\,)\,(\,\underline{x}(k) - \hat{\underline{x}}(k/k)\,)^T\,]$$

and

$$\underline{V}(k/k-1) = E[\,(\,\underline{x}(k) - \hat{\underline{x}}(k/k-1)\,)\,(\,\underline{x}(k) - \hat{\underline{x}}(k/k-1)\,)^T\,]\,.$$

In [3] Kalman showed that this filter was the solution to two different linear systems problems. First, the filter is the optimal estimator against a wide class of cost functions given that the noise processes are Gaussian and second, the filter is also the linear minimum variance estimator without the assumption of Gaussian noise.

6.3 THE KALMAN FILTER AS AN IIR EQUALISER

The major difficulty with the IIR Wiener equalising filter is the minimum phase spectral factorisation of the power spectrum $R_{xx}(z)$. One possible solution to the problem involves the use of a Kalman filter. If all processes are stationary and the observation noise is white, the steady-state Kalman filter and the IIR Wiener filter are identical [5]. Thus by use of a Kalman filter the spectral factorisation problem is solved indirectly. However this does not imply that the IIR Wiener filter formulation of section 5.2 is of no value. On the contrary it can provide insight into the relationship between the channel characteristics and the structure and performance of the Kalman equalising filter.

Although a FIR channel model lends itself readily to state-space representation and hence to a Kalman filter formulation of the equalising filter, care must be taken in the choice of states to form the state vector. A FIR filter with M tap coefficients has order M-1 and could be completely described by M-1 states. A state vector with M-1 states leads to a formulation where the plant noise and the observation noise are correlated. While it is still possible to derive a Kalman filter to handle this situation, such a filter is only conditionally stable. Its stability is dependent upon the impulse response of the channel [5]. In [18] and [108] the channel is represented by M states, which leads to a formulation where the plant noise and the observation noise are uncorrelated, and the Kalman filter is unconditionally stable.

Following [18], the state of the FIR channel model is represented by M-vector $\underline{s}(k)$, where

$$\underline{s}(k) = [\, s(k)\; s(k-1)\; \cdots\; s(k-M+1)\,]^T \;.$$

The state transition equation is therefore,

$$\underline{s}(k) = \underline{a}\; \underline{s}(k-1) + \underline{b}\; s(k) \qquad (6.3.1)$$

where $\underline{a}$ is an $(M \times M)$ shift matrix whose elements $\underline{a}(i,j)$ are equal to unity if i-j=1 and are zero otherwise, $\underline{b}$ is a column vector with M elements.

$$\underline{b}^T = [\,1\,0\,0\,\cdots\,0\,]$$

The observation equation is then obtained directly from (5.2.1).

$$x(k) = \underline{h}^T \underline{s}(k) + n(k)$$

As in section 5.2, $s(k)$ and $n(k)$ are assumed to be zero mean uncorrelated white noise sequences with variances $\sigma_s^2(k)$ and $\sigma_n^2(k)$ respectively.

The above state-space description is adequate if all that is required is an estimate of the current state of the channel $\underline{s}(k)$. However, to handle non-minimum phase channels, a fixed lag smoothing form of the Kalman filter is used [5]. A fixed lag smoother with lag d is usually derived by augmenting the state vector to

$$[\,\underline{s}^T(k)\,\underline{s}^T(k-1)\,\cdots\,\underline{s}^T(k-d)\,]^T\,.$$

However because the state transition matrix $\underline{a}$ is a shift matrix, this degree of complexity is not necessary. The state vector is merely augmented to contain d+1 elements [103].

$$\underline{s}^T(k) = [\,s(k)\,s(k-1)\,\cdots\,s(k-M+1)\,\cdots\,s(k-d)\,]$$

The resultant state transition equation and observation equation are identical in form to (6.3.1) and (5.2.1) respectively with $\underline{a}$ and $\underline{b}$ redefined as follows: $\underline{a}$ is a $(d+1 \times d+1)$ shift matrix, $\underline{b}$ is a vector with d+1 elements

$$\underline{b}^T = [\,1\,0\,0\,\cdots\,0\,],$$

and the $(1 \times d+1)$ observation matrix, $\underline{H}$, constructed by augmenting the channel impulse response with zeros.

$$\underline{H} = [\,h_0\,h_1\,\cdots\,h_{M-1}\,0\,0\,\cdots\,0\,]\,.$$

The Kalman IIR equaliser equations can be written down directly from the above definitions.

$$\hat{\underline{s}}(k/k) = \hat{\underline{s}}(k/k-1) + \underline{K}(k)\,[\,x(k) - \underline{H}\,\hat{\underline{s}}(k/k-1)\,] \tag{6.3.2}$$

$$\hat{\underline{s}}(k/k-1) = \underline{a}\,\hat{\underline{s}}(k-1/k-1) \tag{6.3.3}$$

$$\underline{K}(k) = \underline{V}(k/k-1)\,\underline{H}^T\,[\,\underline{H}\,\underline{V}(k/k-1)\,\underline{H}^T + \sigma_n^2\,]^{-1} \tag{6.3.4}$$

$$\underline{V}(k/k-1) = \underline{a}\,\underline{V}(k-1/k-1)\,\underline{a}^T + \underline{b}\,\underline{b}^T\,\sigma_s^2 \tag{6.3.5}$$

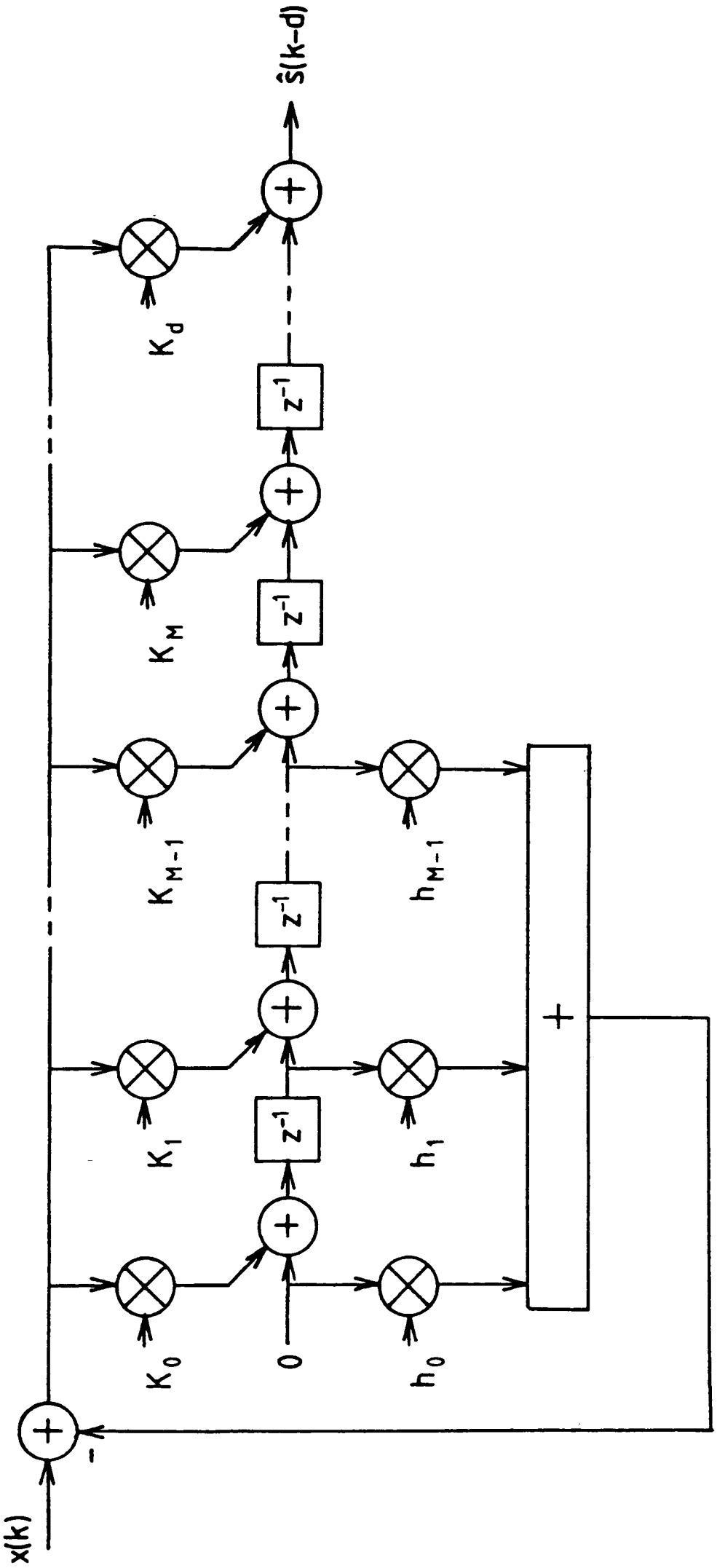
$$\underline{V}(k/k) = [\,I - \underline{K}(k)\,\underline{H}\,]\,\underline{V}(k/k-1) \tag{6.3.6}$$

The structure of the Kalman equalising filter is illustrated in Figure 6.1. In the steady state the elements of the Kalman gain vector

$$\underline{K}^T = [\,K_0\,K_1\,\cdots\,K_d\,]$$

will be constant. In which case the Kalman filter structure illustrated in Figure 6.1 is a canonical form of the Wiener filter structure illustrated in Figure 5.2. Unlike the optimum FIR equaliser and the Wiener IIR equaliser the Kalman equaliser provides estimates of the channel input at a range of lags ie. from lag 0 to lag d.

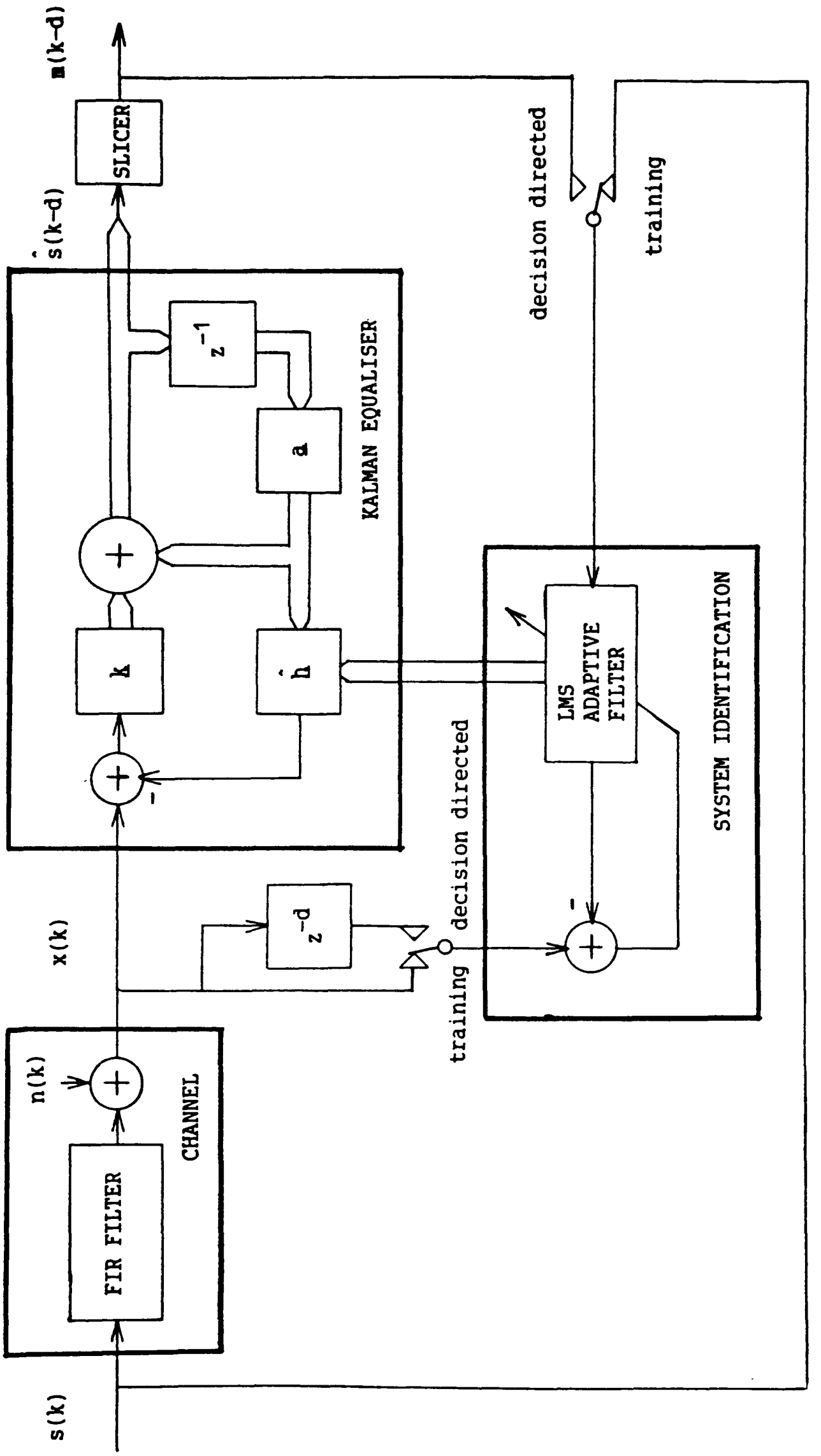
Figure 6.1 KALMAN EQUALISER STRUCTURE



6.4 AN ADAPTIVE KALMAN EQUALISER

In order to make the Kalman equaliser of section 6.3 adaptive, a system identification algorithm is used in parallel with it to estimate the impulse response of the unknown channel. This adaptive equaliser, (Figure 6.2), operates in the following manner. During an initial training period a predetermined sequence is transmitted and the system identification algorithm forms an estimate $\hat{\underline{h}}$ of the channel impulse response vector $\underline{h}$. The estimate is passed to the Kalman filter which is then initialised and transmission of data begins. Because a training sequence is employed, initialisation of the Kalman filter is exact. The state vector at time zero, $\hat{\underline{s}}(0/0)$, consists of the last d samples of the training sequence. The error covariance matrix $\underline{V}(0/0)$, is the zero matrix since $\hat{\underline{s}}(0/0)$ is known with probability one. In addition to estimating the signal at all lags up to lag d, the Kalman filter provides the error variance of these estimates on the leading diagonal of the error covariance matrix, $\underline{V}(k/k)$. Thus once the Kalman filter has reached a steady state, the element of the state vector $\hat{\underline{s}}(k/k)$ which achieves a predetermined performance bound is used as the input to the decision circuit or slicer. During data transmission it is possible to operate the adaptive equaliser in decision directed mode. In this mode, the output of the slicer, m, is used as an input to the system identification algorithm. Thus, provided the slicer makes correct decisions, the system identification algorithm and hence the adaptive equaliser will be capable of tracking time varying channels. when the channel is non-minimum phase adequate performance may only be achieved by using a fixed lag d. Under these conditions the output of the slicer will represent the channel input d samples ago. In order that the system identification algorithm produce correct results a delay d must be introduced between the channel output and the training input to the system identification algorithm.

Figure 6.2 AN ADAPTIVE IIR EQUALISER



6.4.1 System Identification

Since the channel input sequence $\{ s(k) \}$ is white, the autocorrelation matrix $E[\, \underline{s}(k)\, \underline{s}^T(k)\,]$ is diagonal with equal eigenvalues. Under these conditions a stochastic gradient LMS algorithm achieves its best performance [21] and hence it is well suited to perform the system identification task. Of course, a RLS adaptive transversal filter converges faster than the LMS algorithm even under white input conditions [71]. However this performance gain is achieved at the expense of increased complexity and since a Kalman algorithm has already been postulated to perform the equalisation task, it is natural to choose the simpler LMS algorithm for initial study.

The stochastic gradient LMS algorithm is summarised by the following two equations:

$$\underline{\hat{h}}(k+1) = \underline{\hat{h}}(k) + 2\mu\, \underline{s}(k+1)\, e(k+1) \tag{6.4.1}$$

$$e(k+1) = x(k+1) - \underline{\hat{h}}^T(k)\, \underline{s}(k+1) \tag{6.4.2}$$

The constant μ is the convergence factor and the M-vector $\underline{\hat{h}}$ is an estimate of the channel impulse response vector $\underline{h}$. For system identification, the performance measure which is of most use is the norm ρ of the estimated tap weight error vector $\underline{\tilde{h}}$ where

$$\rho(k) = E[\, \underline{\tilde{h}}^T(k)\, \underline{\tilde{h}}(k)\,]$$

$$\underline{\tilde{h}}(k) = \underline{\hat{h}}(k) - \underline{h} \tag{6.4.3}$$

In geometrical terms, the norm is the average length of the error vector $\underline{\tilde{h}}$ and hence is a measure of how close the estimate $\underline{\hat{h}}$ is to the vector $\underline{h}$.

Standard theoretical analysis of the LMS algorithm [73, 26], yields the following equations which summarise the convergence properties when the input signal is white.

$$E[\, e^2(k+1)\,] = \left(\, 1 - 4\mu\sigma_s^2 + 4\mu^2\sigma_s^4 M \,\right) E[\, e^2(k)\,] + 4\mu\sigma_s^2\sigma_n^2$$

(6.4.4)

$$E[\, e^2(k+1)\,] = \sigma_s^2\, \rho(k) + \sigma_n^2 \tag{6.4.5}$$

$$0 \leq \mu \leq \frac{1}{M\sigma_s^2}$$

Although the analysis relies on assumptions that are often invalid, it produces figures which agree well with experimental results and hence is useful in the understanding and operation of the adaptive Kalman equaliser. Combining (6.4.4) and (6.4.5) a recursive equation for the norm $\rho(k)$ is obtained.

$$\rho(k) = \left(\, 1 - 4\mu\sigma_s^2 + 4\mu^2\sigma_s^4 M \,\right) \rho(k-1) + 4\mu^2\sigma_s^2\sigma_n^2 M \tag{6.4.6}$$

In order to achieve fast convergence, let

$$\mu = \frac{1}{2M\sigma_s^2}\,.$$

If the signal power is normalised to unity ie.

$$\sigma_s = 1$$

equation (6.4.6) becomes

$$\rho(k) = \left(\, 1 - \frac{1}{M} \,\right) \rho(k-1) + \frac{\sigma_n^2}{M}\,. \tag{6.4.7}$$

Equation (6.4.7) provides two important results: (i) the fastest convergence rate that can be obtained using the LMS algorithm is

$$10 \log_{10} \left(\frac{M}{M-1} \right) \quad \textit{dB/iteration}$$

and (ii) the final value of the norm under these conditions is determined by the variance of the noise i.e.

$$\lim_{k \to \infty} \rho(k) = \sigma_n^2 \ .$$

6.4.2 Model Uncertainty

The two parameters in the channel model, (5.2.1), of which there is uncertainty or imperfect knowledge are the channel impulse response vector $\underline{h}$ and the additive noise variance σ_n^2. In [109] it is shown that if the noise variance used in the Kalman filter is greater than the actual noise variance, then the diagonal elements of the error covariance matrix $\underline{V}(k/k)$ are an upper bound on the MSE performance of the filter. The effect is illustrated in Figure 6.3, which also indicates that the MSE is not significantly degraded by using a noise variance parameter in the Kalman filter that is 10 dB greater than the actual noise variance. The theoretical result presented in [109] and experimental results such as Figure 6.3 combine to suggest a simple solution to the problem of uncertainty in the observation noise i.e. the noise variance in the Kalman filter should be set to a maximum or worst case value that is expected in a particular application. There are however two disadvantages with this strategy: (i) the performance of the Kalman filter will be degraded if only slightly, and (ii) the diagonal elements of the error covariance matrix will be greater than the actual MSE and hence will not be useful as indicators of the equaliser performance. An alternative solution is to estimate the noise variance directly from the channel [108]. Before considering this solution, the effect of uncertainty in the channel tap vector $\underline{h}$ on the performance of the Kalman filter will be considered briefly.

In subsection 6.4.1 it was shown that the lower bound on the norm ρ is σ_n^2, when the convergence factor μ is set for fastest convergence. Experiment indicates that this is the point at which the uncertainty in the tap vector $\underline{h}$ starts to degrade the

performance of the Kalman filter (Figure 6.4). For this experiment the estimated tap vector $\hat{\underline{h}}$ used in the Kalman filter was formed by adding a suitably scaled random noise term to each coefficient of the channel tap vector $\underline{h}$. Recent developments [110] provide the means for incorporating some of the effects of model uncertainty into the Kalman filter formulation. However these rely on spectral factorisation. The technique that is now presented provides a method for both estimating the noise variance and compensating the Kalman filter for the uncertainty in the channel impulse response vector.

Combining (5.2.1) with (6.4.2) and (6.4.3) yields.

$$x(k+1) = \underline{h}^T \underline{s}(k+1) + n(k+1)$$

$$= \left(\hat{\underline{h}}(k) - \tilde{\underline{h}}(k) \right)^T \underline{s}(k+1) + n(k+1)$$

$$= \hat{\underline{h}}^T(k)\, \underline{s}(k+1) + e(k+1) \qquad (6.4.8)$$

If, to a first order approximation, $\{ e(k+1) \}$ is considered to be a white noise process, then (6.4.8) provides an alternative interpretation of how the observation sequence $\{ x(k+1) \}$ was formed. If it is assummed that $\hat{\underline{h}}(k)$ and $\underline{s}(k+1)$ are uncorrelated then a simple expression for the variance σ_e^2 of the error e can be obtained from (6.4.5).

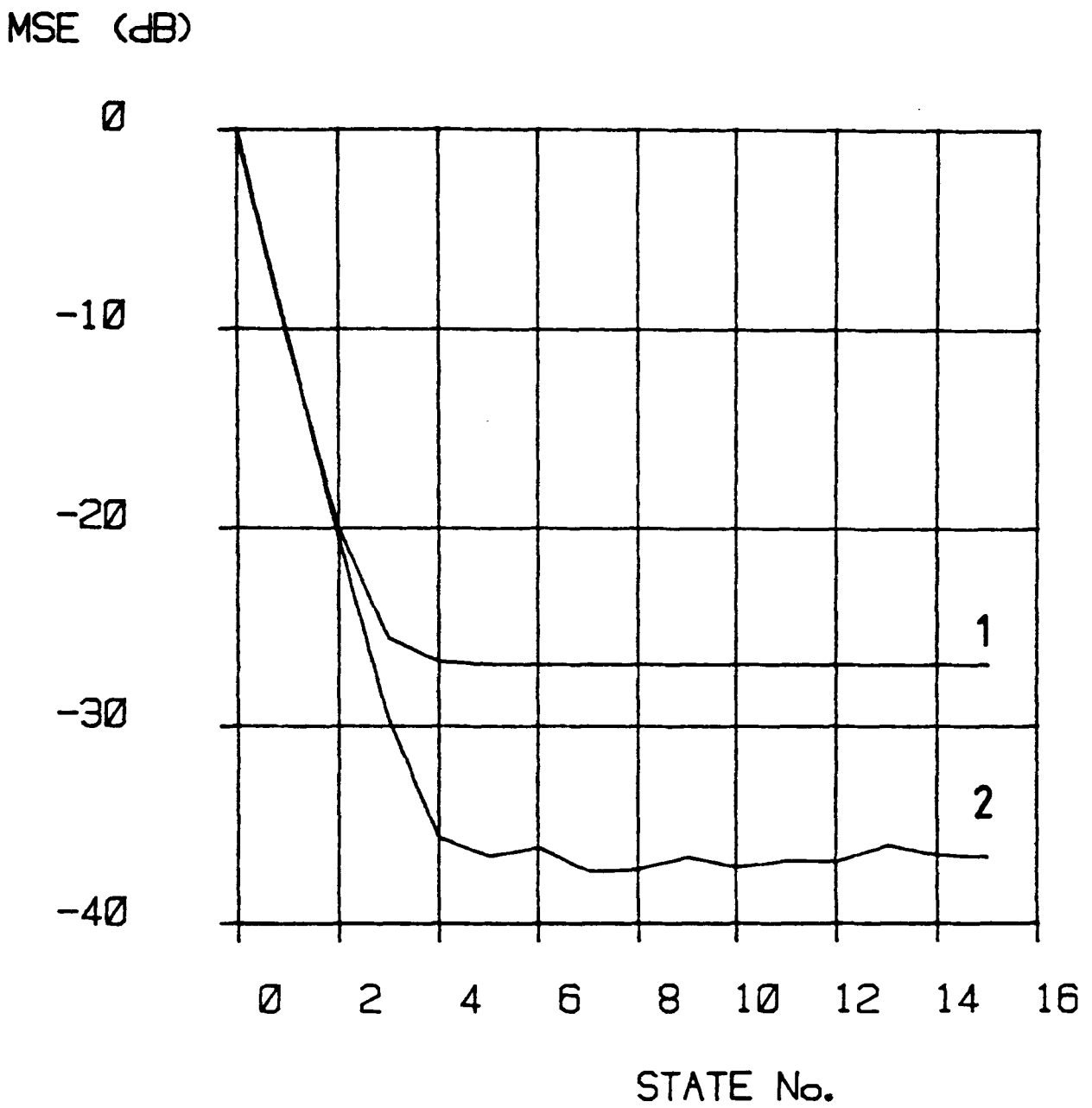
$$\sigma_e^2(k) = \sigma_s^2\, \rho(k) + \sigma_n^2 \qquad (6.4.9)$$

In words, x(k+1) is formed by adding the output of a time varying FIR filter with tap vector $\hat{\underline{h}}(k)$ to a time varying white noise sequence with variance $\sigma_e^2(k+1)$. This variance combines both the effects of model uncertainty, represented by the norm $\rho(k)$, and the variance of the additive noise σ_n^2. Using the alternative model of (6.4.8) the observation noise variance, $\underline{U}(k)$ used in the Kalman equaliser is replaced by $\sigma_e^2(k)$. Although $\sigma_e^2(k)$ will not be known a priori in a particular application the

parameter e(k+1) is directly available as it is an output of the LMS system identification algorithm (6.4.2) and thus $\sigma_e^2(k+1)$ may be readily estimated on-line. Since σ_e^2 is the output of a process with a time constant determined by $\left(1 - \frac{1}{M}\right)$, a suitable recursive estimate $\hat{\sigma}_e^2$ is.

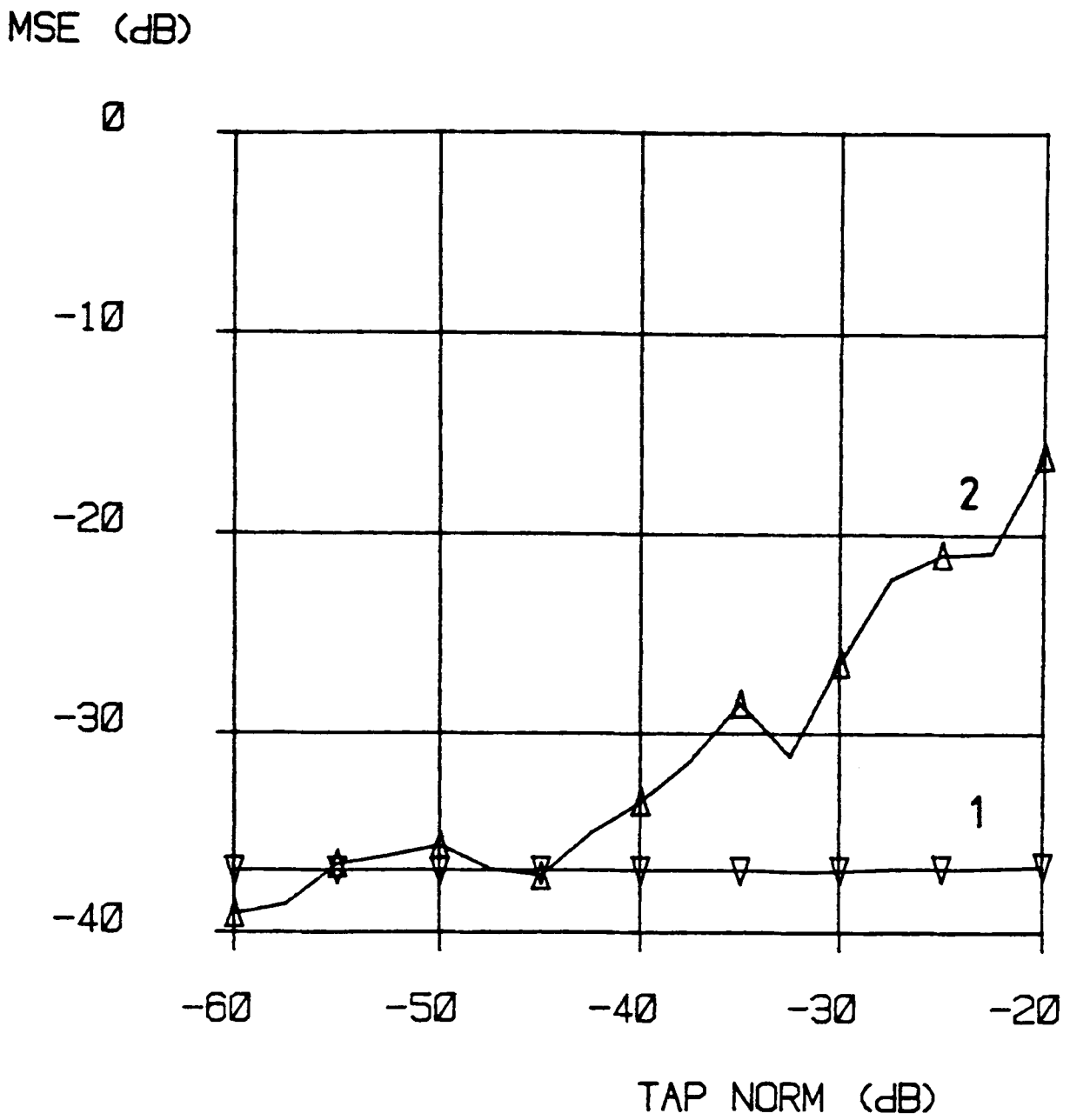
$$\hat{\sigma}_e^2(k+1) = \left(1 - \frac{1}{M}\right)\hat{\sigma}_e^2(k) + \frac{e^2(k+1)}{M}. \qquad (6.4.10)$$

Figure 6.3 MODELLING ERRORS (Observation Noise)
non - minimum phase (channel no. 1)
actual additive noise = -40.0 dB
noise variance in Kalman filter = -30.0 dB
ensemble = 100



KEY	
1	MSE from error covariance equations
2	measured MSE

Figure 6.4 MODELLING ERRORS (Channel Impulse Response)
non - minimum phase (channel no. 1)
additive noise = -40.0 dB
ensemble = 100



KEY	
1	MSE from error covariance equations
2	measured MSE

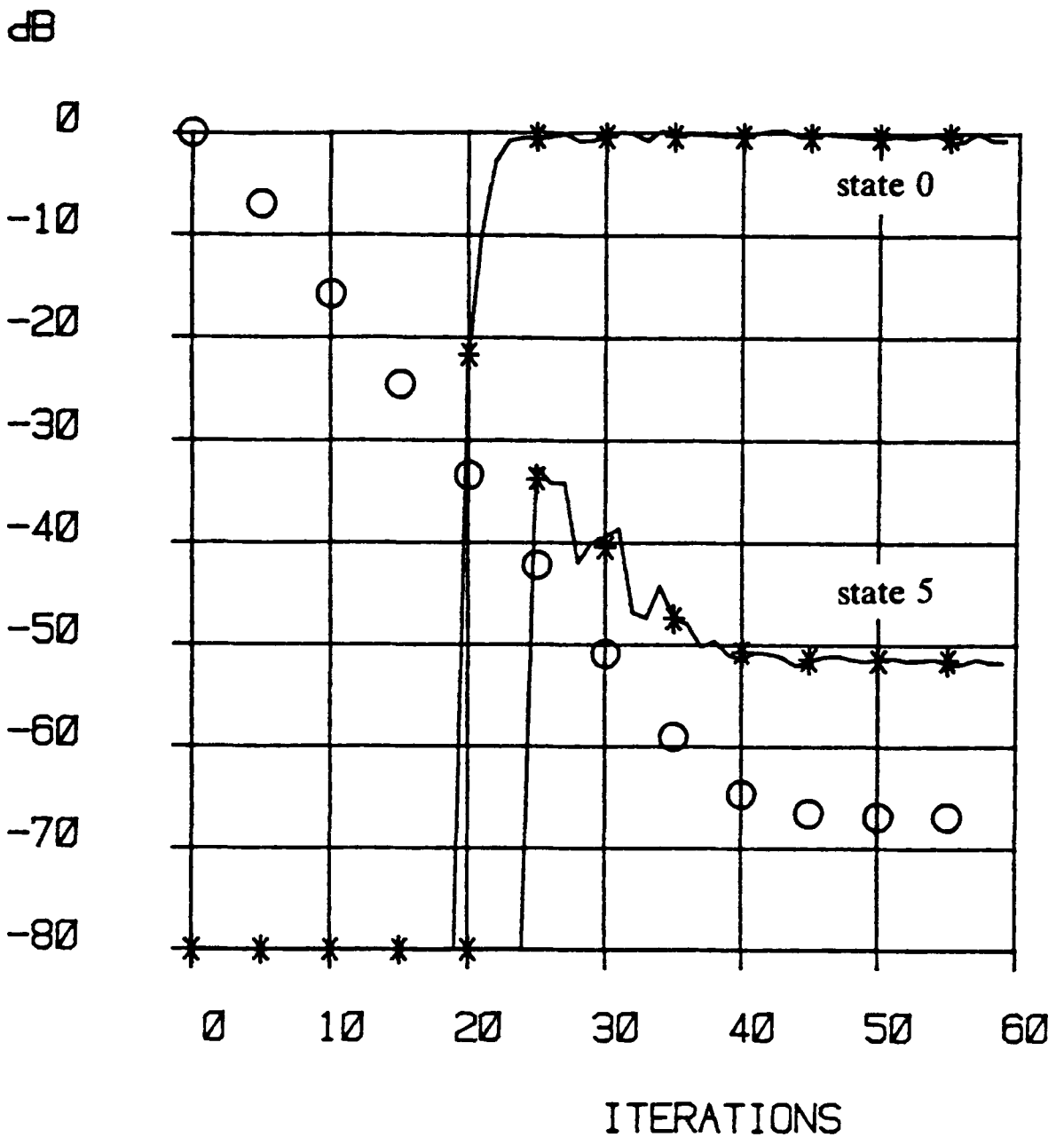
6.4.3 Verification of Compensation Technique

While the compensation technique postulated in subsection 6.4.1 has an intuitive appeal, it is not an exact result and hence it was necessary to examine its validity by experiment. For this and subsequent simulations, the channel was modelled by (5.2.1), a zero mean binary white sequence of unit variance was used as the channel input, and a zero mean white Gaussian sequence was used for the additive noise. The results are summarised in Figure 6.5. For this particular experiment the observation noise variance in the Kalman filter, $\underline{U}$(k), was calculated from (6.4.4) using exact knowledge of the channel noise variance, σ_n^2 ($= -70dB$). The estimated impulse response, $\hat{\underline{h}}(k)$, from the LMS algorithm was used to form the $(1 \times d+1)$ observation matrix, $\underline{H}$(k) ie.

$$\underline{H}(k) = [\, \hat{h}_0(k)\ \hat{h}_1(k)\ \cdots\ \hat{h}_{M-1}\ 0\ 0\ \cdots\ 0\,]$$

A 3-tap FIR filter was used in both the channel simulation and the LMS algorithm. The Kalman filter was initialised after 20 iterations of the LMS algorithm with an error covariance matrix, $\underline{V}(0/0)$, of all zero elements. Hence the initial divergence of the filter state 5 to a MSE value that is above the level of the observation noise variance, σ_e^2, is to be expected. It then reconverges due to the subsequent reduction in the observation noise, as the LMS algorithm converges, until a final MSE value of -53dB is achieved. Because the channel is non-minimum phase (channel no. 1, Table 5.1), the MSE of the states decreases with increasing state number and increasing estimation lag. Thus the significant difference in performance between states 0 and 5 is to be expected. For clarity the performance of the intermediate states has not been included. The close agreement between the measured MSE of states of the Kalman equaliser and the theoretical performance predicted by the error covariance equations indicates that the compenation technique is a valid one.

Figure 6.5 COMPENSATION FOR MODEL UNCERTAINTY
non - minimum phase (channel no. 1)
additive noise = -70.0 dB
ensemble = 100



KEY	
O	theoretical MSE of LMS algorithm (6.4.4)
* *	error covariance
~~	measured MSE of Kalman equaliser

6.4.4 Comparison with an RLS FIR Equaliser.

In this section, the results of a simulation study of the adaptive Kalman equaliser are presented. In contrast to subsection 6.4.3 the observation noise variance, $\sigma_e^2(k)$, was estimated directly from the LMS error sequence $\{ e(n) \}$, using (6.4.10). Thus the equaliser is fully adaptive in that a priori knowledge of the channel noise variance, σ_n^2 or the tap vector norm, $\rho(k)$, is not required. The measured values of the tap norm, $\rho(k)$, are included as a check on the theoretical result summarised in (6.4.7). A RLS adaptive transversal equaliser [40] was chosen as a benchmark against which to compare the adaptive Kalman equaliser.

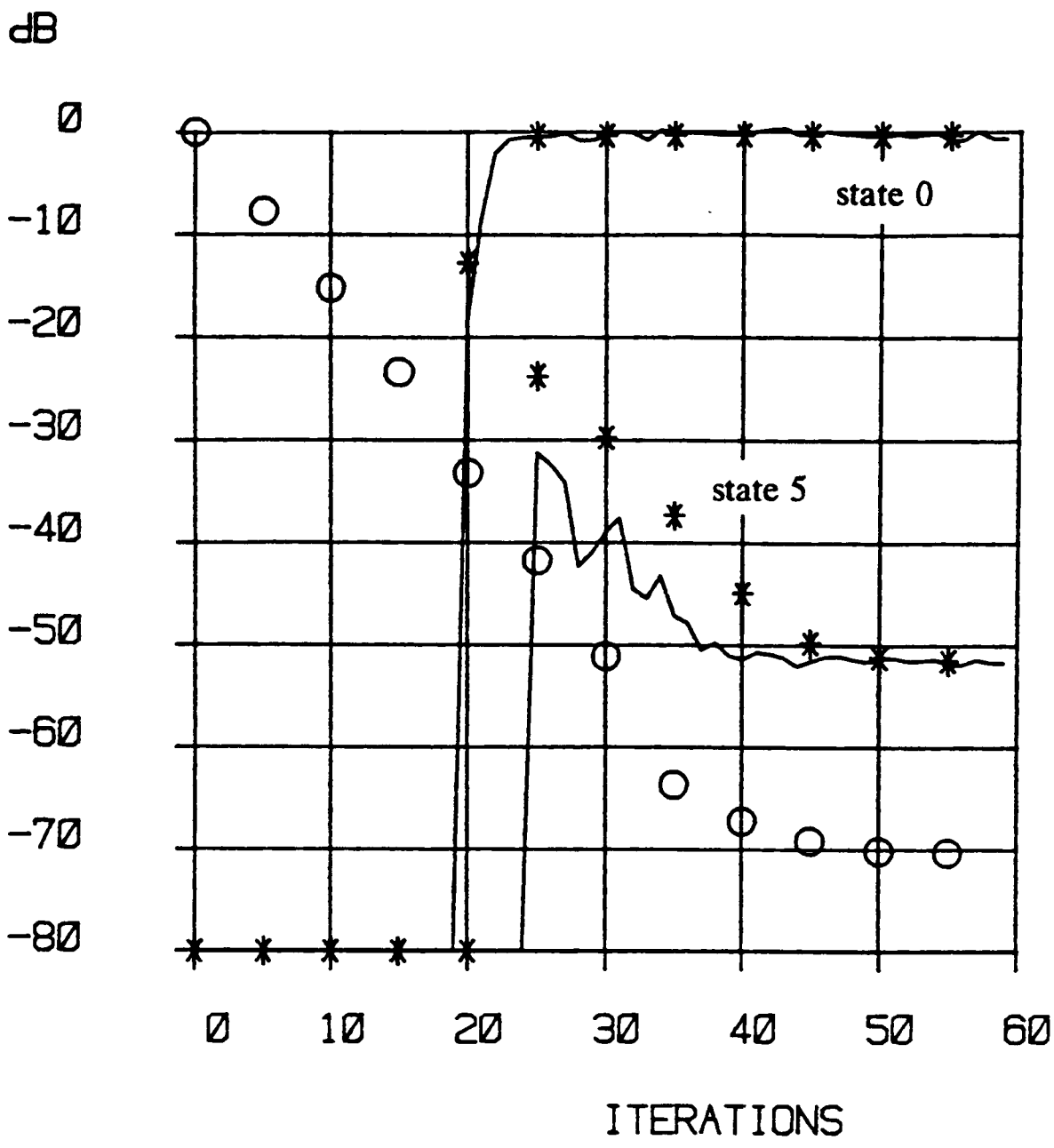
The results for channel no.1 are illustrated in Figure 6.6(a). The norm, $\rho(k)$, converges to the noise floor of -70dB at a rate that is consistent with (6.4.7). Like the channel the LMS algorithm has 3 taps. Thus the rate of convergence should be $\log_{10}(1.5)$ dB/iteration or -70dB in 40 iterations. While the elements of the error covariance matrix $\underline{V}(k/k)$ track the measured MSE values of states 0 and 5 of the Kalman equaliser closely there is a noticable seperation during the transient periods when the observation noise variance is changing. This is due to the inherent lag in the estimator of (6.4.10). However comparison of Figures 6.5 and 6.6(a) indicates that the measured MSE of state 5 of the Kalman equaliser is not altered by this seperation. A measured MSE of -50dB is achieved by state 5 of this 5th order adaptive Kalman equaliser within 38 iterations. To obtain the same MSE with a FIR equaliser, a filter of order 9 is required. The optimum lag under these conditions is 5. Using a RLS algorithm to train the FIR filter, it converges in approximately 30 iterations, Figure 6.6(b).

The results for a minimum phase channel are illustrated in Figure 6.7. The three states of the Kalman filter (states 0, 1 and 2) exhibit similar MSE performance because the channel is minimum phase. Again the elements of the error covariance matrix, $\underline{V}(k/k)$, track the measured MSE values closely. In order to achieve a MSE

value of -50dB under exactly the same conditions a FIR equaliser of order 22 and estimation lag 0 is required (Figure 6.7(a)). Using a RLS algorithm to train the filter, it converges in approximately 50 iterations (Figure 6.7(b)). The second order adaptive Kalman equaliser converges to -50dB in about 40 iterations.

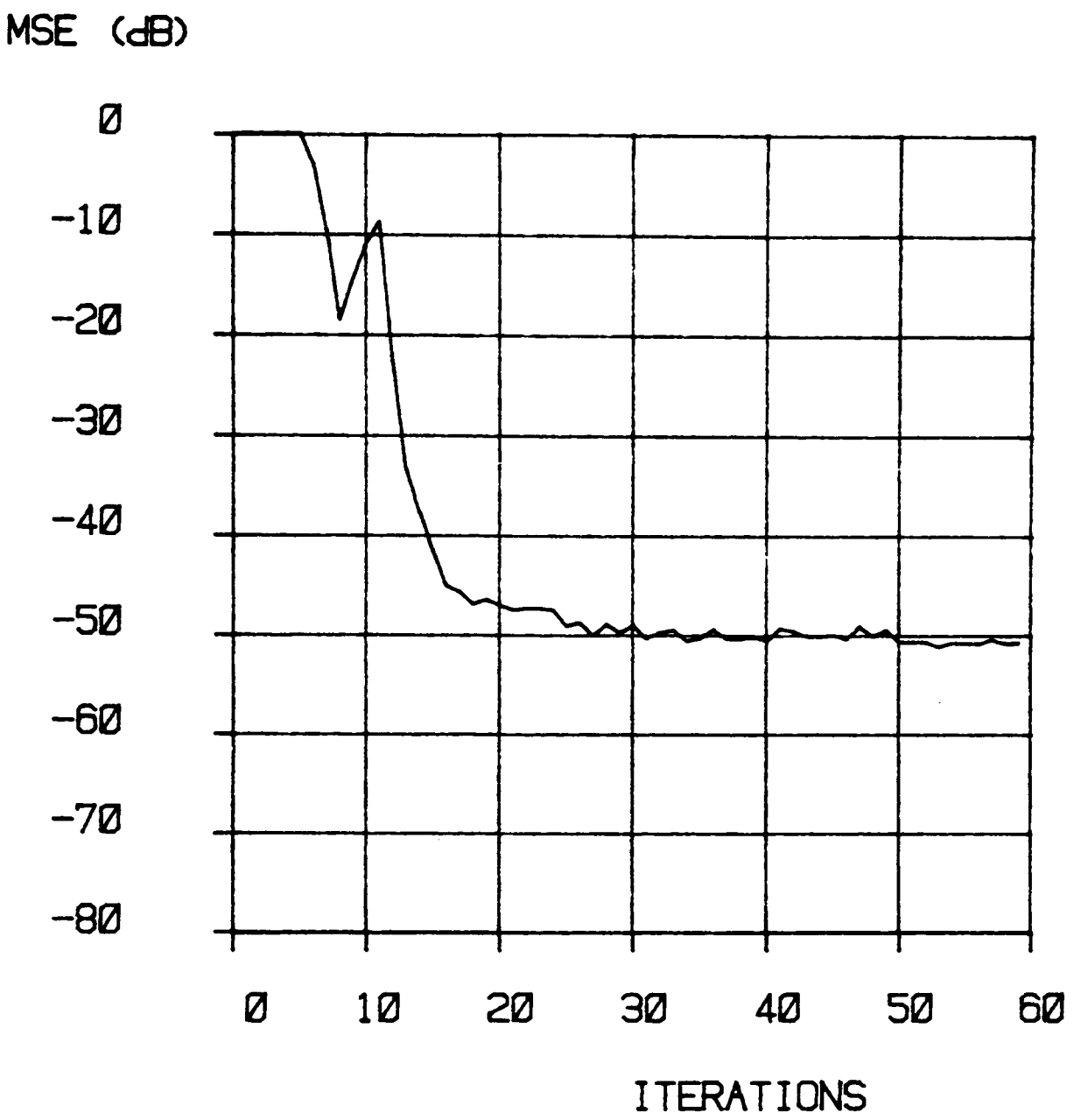
Finally the results for a second non-minimum phase channel are illustrated in Figure 6.8. An 8th order Kalman equaliser is required to obtain a MSE of -48dB. The equaliser converges in 45 iterations. Figure 6.8(b) illustrates the MSE performance of 17-tap RLS transversal equaliser with an estimation lag of 9. The RLS equaliser converges within 35 iterations.

Figure 6.6 COMPARISON WITH RLS FIR EQUALISER
non - minimum phase (channel no. 1)
additive noise = -70.0 dB
ensemble = 100



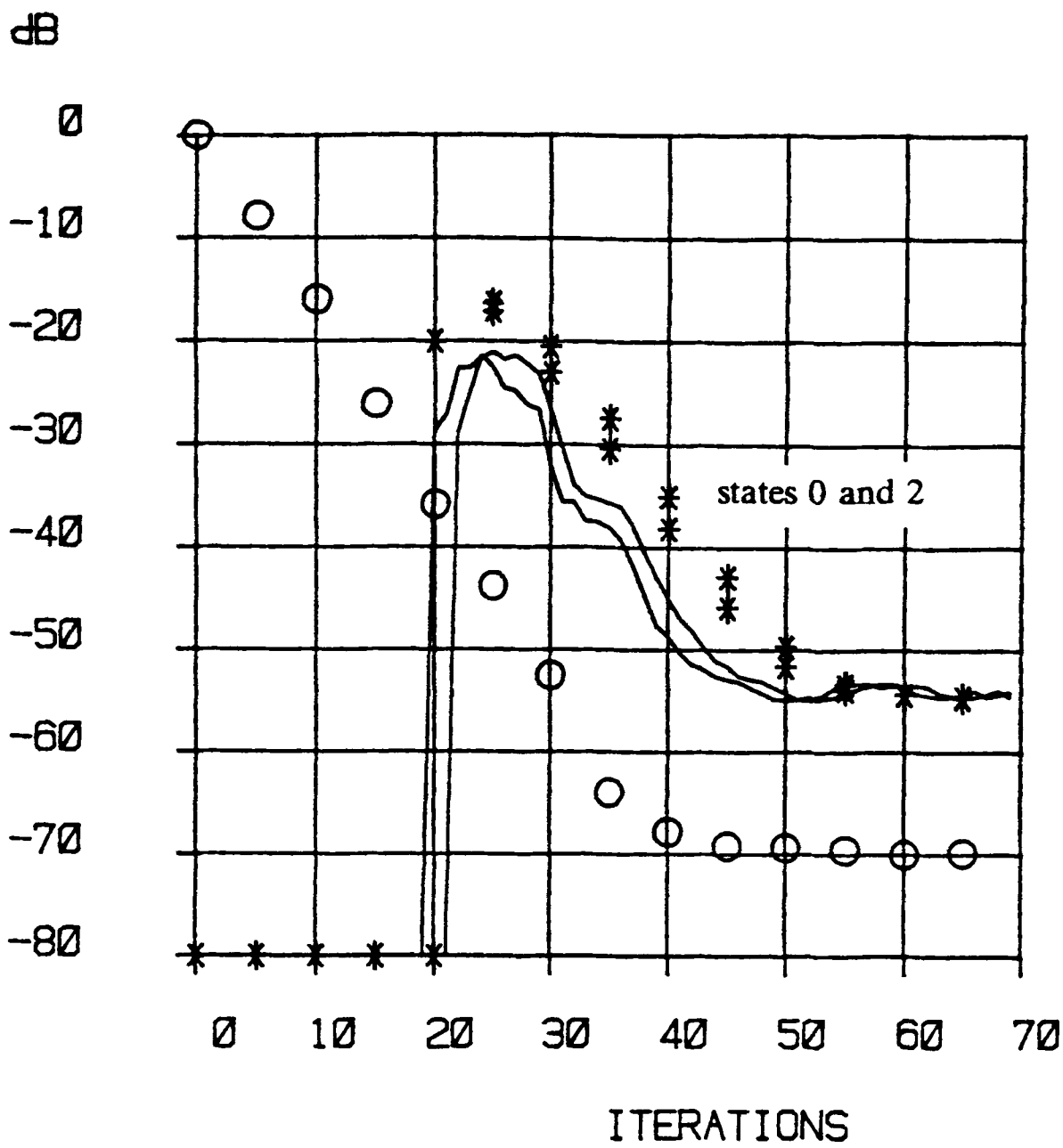
(a) Adaptive Kalman Equaliser (order = 5)

KEY	
O	measured norm of system identification
* *	error covariance
~~	measured MSE of Kalman equaliser



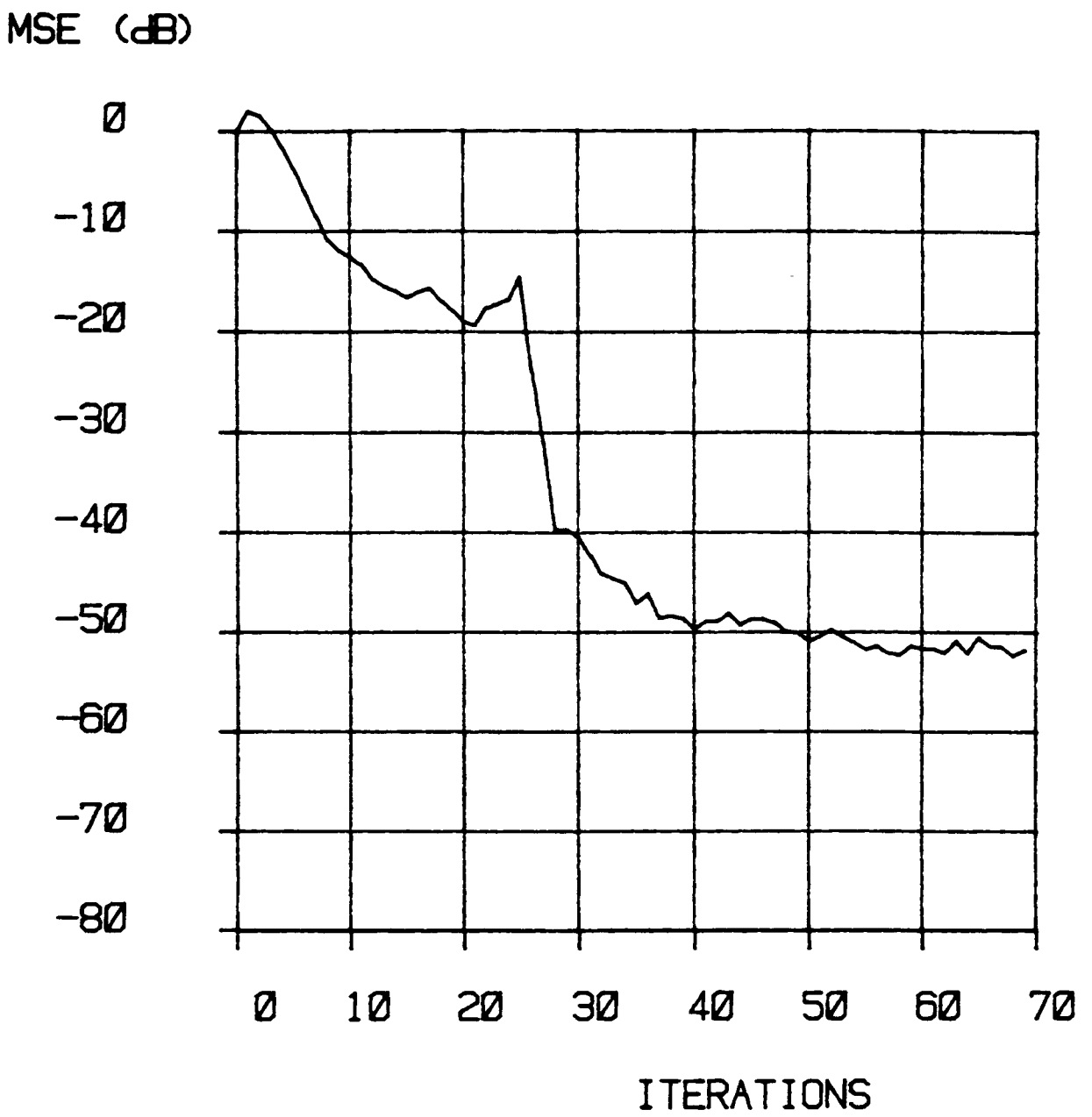
(b) RLS FIR Equaliser (order = 9, lag = 5)

Figure 6.7 COMPARISON WITH RLS FIR EQUALISER
minimum phase (channel no. 3)
additive noise = -70.0 dB
ensemble = 100



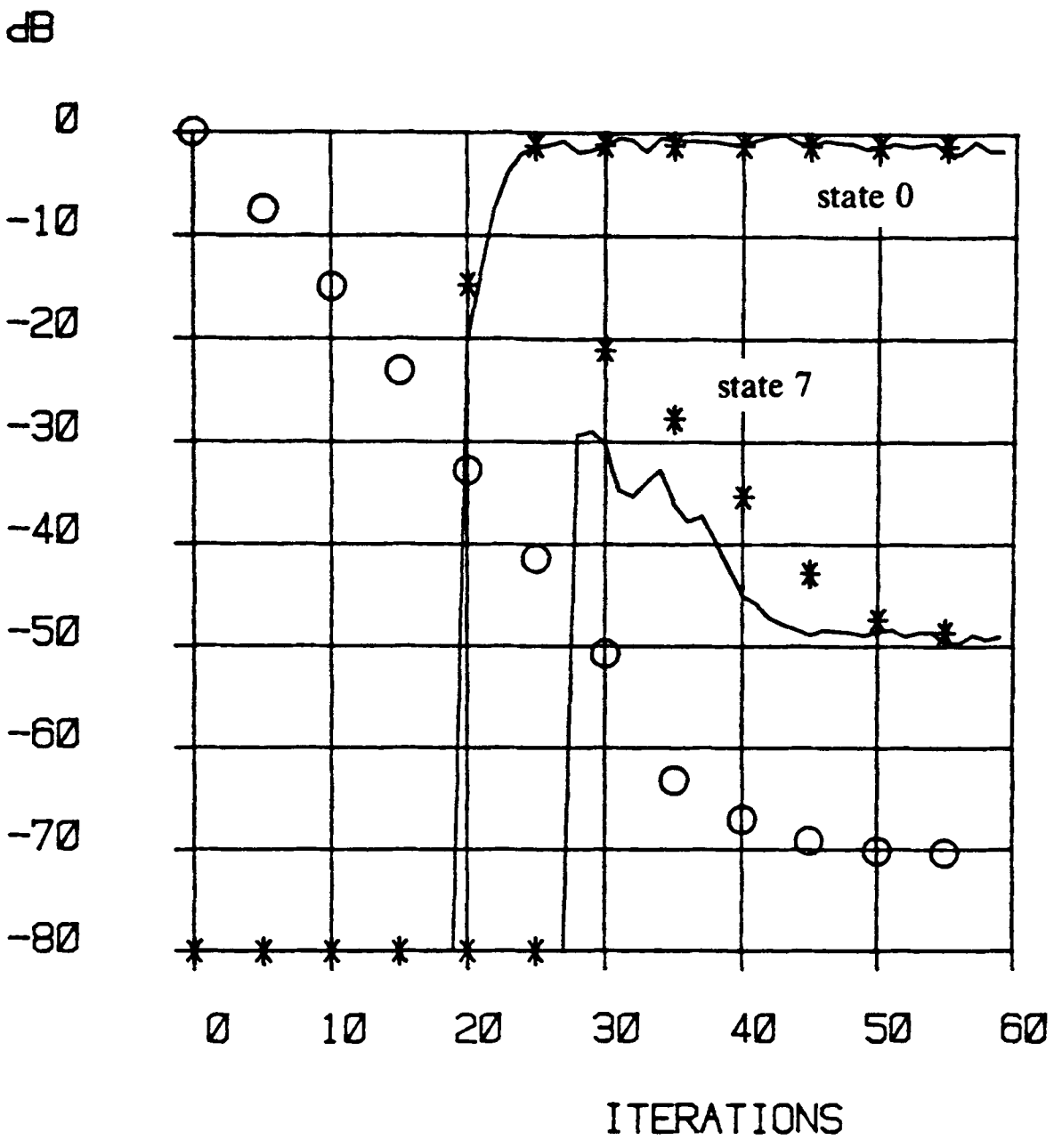
(a) Adaptive Kalman Equaliser (order = 2)

KEY	
O	measured norm of system identification
* *	error covariance
~~	measured MSE of Kalman equaliser



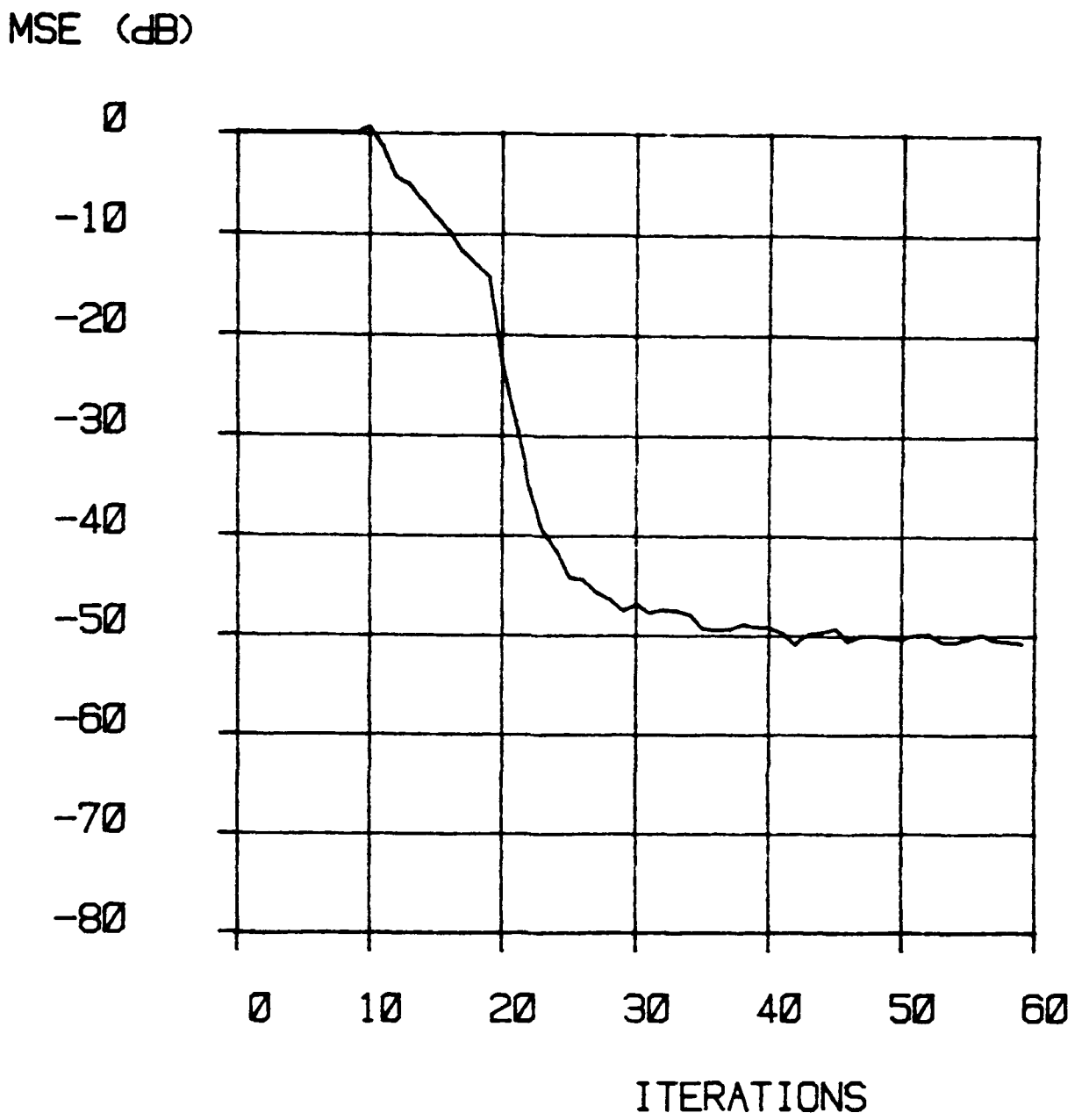
(b) RLS FIR Equaliser (order = 22, lag = 0)

Figure 6.8 COMPARISON WITH RLS FIR EQUALISER
non - minimum phase (channel no. 2)
additive noise = -70.0 dB
ensemble = 100



(a) Adaptive Kalman Equaliser (order = 8)

KEY	
O	measured norm of system identification
* *	error covariance
~~	measured MSE of Kalman equaliser



(b) RLS FIR Equaliser (order = 16, lag = 9)

6.4.5 Computational Complexity

A breakdown of the computation required to process one data point using the adaptive Kalman equaliser of Figure 6.2 is presented in Table 6.1. The complete algorithm includes the Kalman equaliser, (6.3.2) - (6.3.5), LMS system identification, (6.4.1) and (6.4.2), and estimation of the error variance $\sigma_e^2(k)$, (6.4.10). In total

$$\frac{d^2}{2} + \frac{3d}{2} + dM + 4M + 6$$

multiplications,

$$\frac{d^2}{2} - \frac{d}{2} + dM + 4M$$

additions and/or subtractions and 1 division are required to process one sample. The major computational burden of the algorithm lies in the the Kalman equaliser itself, in particular the computation of the error covariance matrix, $\underline{V}(k/k)$. Some saving can be made with respect to the general Kalman filter, (6.2.1) - (6.2.5), since the state transition matrix, $A(k)$, is a shift matrix, $\underline{a}$, and hence (6.3.5) requires no computation. Further the observation matrix, $\underline{H}(k)$, contains many zero elements.

The total number of multiplications required to implement the algorithm is illustrated in Figure 6.9 for various values of the number of channel taps, M, and the estimation lag, d. For minimum phase channels the estimation lag does not affect the performance of the adaptive Kalman equaliser and the computational complexity of the algorithm is obtained by setting $d = M - 1$. This is the minimum phase boundary of Figure 6.9. For a non-minimum phase channel a value of $d > M - 1$ may be required to obtain adequate performance. For a given number of channel taps, increasing the estimation lag increases the computational burden.

To extend the comparison with a RLS FIR equaliser from one of performance to one of computational complexity is not straightforward since the order of both structures to achieve a particular performance goal depends on the characteristics of

the channel. Because of this dependency the comparison presented here is a limited one which is derived from the performance comparison of IIR and FIR equalisers which was presented in section 5.3. Using Figures 5.3 - 5.5, values for the order of the IIR equaliser (and hence d) and the order of the FIR equaliser (and hence N) were chosen which would give the same MSE performance for a given channel and noise conditions. As in section 5.3 the FIR equaliser of optimum lag was used. Given these values for d and N, the number of computations required to process one data sample was calculated. It was assumed that the RLS FIR was implemented using the fast Kalman algorithm of Appendix A. The number of computations is given in Table 2.2. The results presented in Table 6.2 indicate the savings in computation that can be achieved using the adaptive Kalman equaliser structure. These savings range from 11-18% for the minimum phase channel (no. 3) and 50-70% for the non-minimum phase channels.

Table 6.1 ADAPTIVE KALMAN IIR EQUALISER
(Complexity)

no. of states = d+1

no. of channel taps = M

operation	mult.	add/sub.
$\underline{V}(k/k-1)\ \underline{H}(k)$	$d(M-1)+1$	$d(M-2)$
$\underline{H}^T(k)\ \underline{V}(k/k-1)\ \underline{H}(k) + \sigma_n^2$	M	M
$x(k) - \underline{H}^T(k)\ \hat{\underline{s}}(k/k-1)$	M-1	M-1
$\underline{V}(k/k-1)\ \underline{H}(k)\ [\ \underline{H}^T(k)\ \underline{V}(k/k-1)\ \underline{H}(k) + \sigma_n^2\ ]^{-1}$	d+1	
$\hat{\underline{s}}(k/k-1) + \underline{K}(k)\ [\ x(k) - \underline{H}^T(k)\ \hat{\underline{s}}(k/k-1)\ ]$	d+1	d
$\underline{K}(k)\ \underline{H}^T(k)\ V(k/k-1)$	$\frac{d^2}{2}+\frac{d}{2}$	
$\underline{V}(k/k-1) - \underline{K}(k)\ H^T(k)\ \underline{V}(k/k-1)$		$\frac{d^2}{2}+\frac{d}{2}$
$y(k) - \underline{h}^T(k-1)\ \underline{x}(k)$	M	M
$2\ \mu\ \underline{x}(k)\ (\ y(k) - \underline{h}^T(k-1)\ \underline{x}(k)\ )$	M+1	
$\underline{h}(k-1) + 2\ \mu\ \underline{x}(k)\ (\ y(k) - \underline{h}^T(k-1)\ \underline{x}(k)\ )$		M
$(\ 1 - \frac{1}{M}\ )\ \hat{\sigma}_e^2(k-1) + \frac{e^2(k)}{M}$	3	1

total mult. = $\frac{d^2}{2} + \frac{3d}{2} + dM + 4M + 6$

total add/sub. = $\frac{d^2}{2} - \frac{d}{2} + dM + 4M$

Figure 6.9 MUITIPLICATION REQUIREMENTS
(Adaptive Kalman Equaliser)

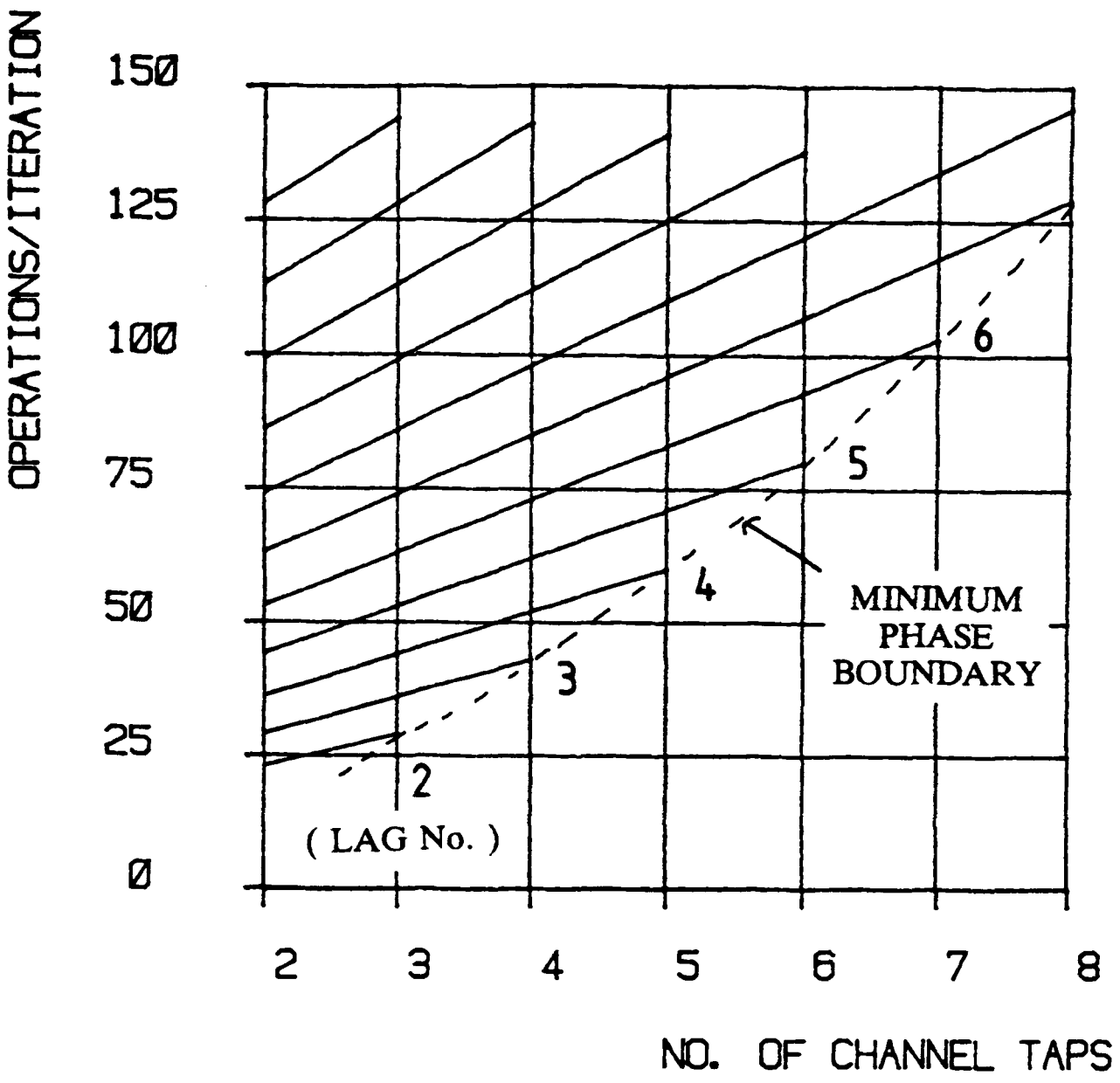


Table 6.2 COMLEXITY COMPARISON

channel		Kalman/LMS			FIR/RLS		
no.	noise	d	mult.	add/sub.	N	mult.	add/sub.
3	-40.0	2	29	19	16	161	145
3	-70.0	2	29	19	27	271	244
1	-40.0	2	29	19	4	41	37
1	-40.0	3	36	24	6	61	55
1	-70.0	4	44	30	8	81	73
1	-70.0	5	53	37	10	101	91
2	-70.0	8	86	64	17	171	154
2	-70.0	11	128	100	21	211	190

6.5 RLS SYSTEM IDENTIFICATION

From the results presented in section 3.2 and in [71], it is clear that the RLS algorithm provides faster converging system identification than the LMS algorithm. Thus, if the LMS system identification block of Figure 6.2 is replaced with a RLS system identification block, the complete adaptive Kalman equaliser will converge faster than the structure considered in section 6.4. The only issue to be resolved is how then to compensate the Kalman equaliser for the uncertainty in the channel impulse response estimate provided by the RLS algorithm and how to remove the need for a priori knowledge of the variance of the channel noise, σ_n^2 .

The approximate analysis, summarised by (2.4.17) - (2.4.19) [58], indicates that, in common with the LMS algorithm, the variance of the error sequence, $\{ e(k) \}$, associated with the RLS algorithm is a simple function of the channel noise variance and the norm, $\rho(k)$. Using (2.4.18) an expression for the norm can be derived when the channel input sequence, $\{ x(k) \}$, is white.

$$\rho(k) = tr \left(E[(\underline{h}(k) - \underline{h}) (\underline{h}(k) - \underline{h})^T] \right)$$

$$= tr \left(\frac{\sigma_n^2}{k} \Phi_{xx}^{-1} \right)$$

$$= \frac{\sigma_n^2}{k} N$$

An expression for $\sigma_e^2(k)$ is then obtained from (2.4.19).

$$\sigma_e^2(k) = \sigma_n^2 + \frac{N}{k} \sigma_n^2$$

$$= \sigma_n^2 + \rho(k)$$

This is identical to (6.4.9). Unfortunately the fast converging property of the RLS algorithm implies that the sequence $\{ e(k) \}$ is highly nonstationary and hence it would

be inappropriate to estimate $\sigma_e^2(k)$ directly from the data using a time average technique such as (6.4.10). The solution that is now proposed is to exploit some of the properties of the RLS algorithm itself in forming a recursive estimate of $\sigma_e^2(k)$.

A satisfactory estimate could be obtained if the impulse response estimate provided by the system identification algorithm was held constant for a finite number of data points. Such an estimate would have the form,

$$\hat{\sigma}_e^2(k) = \frac{1}{(k+1)} E_p(k) \tag{6.5.1}$$

where

$$E_p(k) = \sum_{n=0}^{k} (y(n) - \underline{h}^T(k-1)\, \underline{x}(n))^2 .$$

To find a time recursion for $E_p(k)$, first isolate the term due to the latest data point at $n = k$.

$$E_p(k) = \sum_{n=0}^{k-1} (y(n) - \underline{h}^T(k-1)\, \underline{x}(n))^2 + e^2(k)$$

$$= E_m(k-1) + e^2(k) \tag{6.5.2}$$

The summation, $E_m(k-1)$, is recognisable as the LS cost function evaluated using k data points, and $\underline{h}(k-1)$ is the impulse response that minimises that cost function. The minimum value of the least squares cost function, $E_m(k)$, is obtained by setting the impulse response to its optimum value ie.

$$\underline{h}(k) = \underline{r}_{xx}^{-1}(k)\, \underline{r}_{xy}(k) .$$

Thus,

$$E_m(k) = \sum_{n=0}^{k} (y(n) - \underline{h}^T(k)\, \underline{x}(k))^2$$

$$= \sum_{n=0}^{k} y^2(n) - \underline{h}^T(k)\, r_{xy}(k) \tag{6.5.3}$$

To develope a time recursion for $E_m(k)$, time recursions for each term on the right hand side of (6.5.3) are first derived.

$$\sum_{n=0}^{k} y^2(n) = \sum_{n=0}^{k-1} y^2(n) + y^2(k) \tag{6.5.4}$$

$$\underline{h}(k) = \underline{h}(k-1) + R_{xx}^{-1}(k)\, \underline{x}(k)\, e(k) \tag{6.5.5}$$

$$R_{xx}(k) = R_{xx}(k-1) + \underline{x}(k)\, \underline{x}^T(k) \tag{6.5.6}$$

$$r_{xy}(k) = r_{xy}(k-1) + \underline{x}(k)\, y(k) \tag{6.5.7}$$

Substitution of (6.5.4) - (6.5.7) in (6.5.3) yields

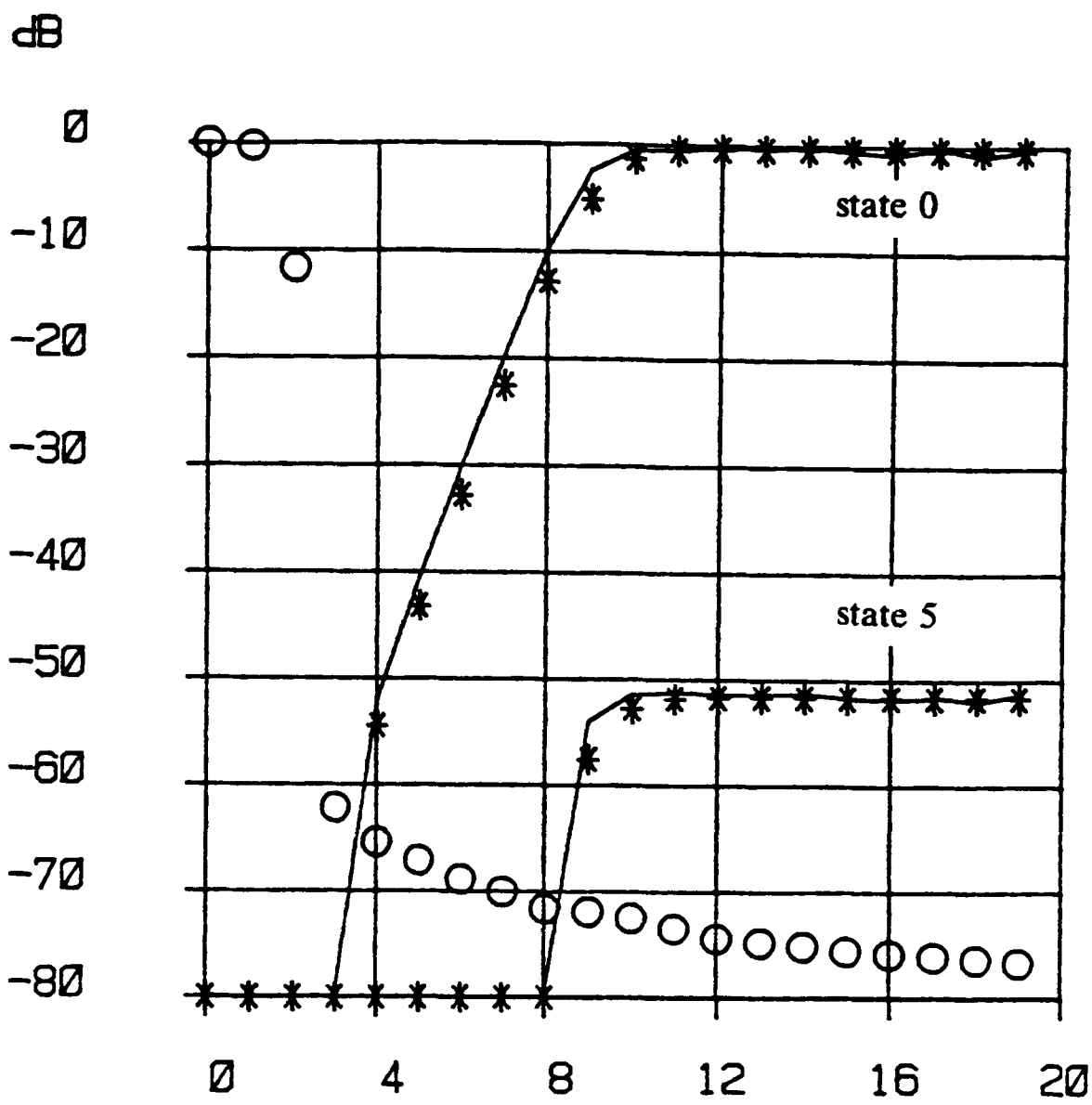
$$E_m(k) = E_m(k-1) + \frac{e^2(k)}{(\,1 + \underline{x}^T(k)\, R_{xx}^{-1}(k-1)\, \underline{x}(k)\,)} \,. \tag{6.5.8}$$

The denominator is of course directly available in the RLS algorithm. Together (6.5.1), (6.5.2) and (6.5.8) form a recursive estimate of the time varying variance $\sigma_e^2(k)$.

To assess the potential improvement in performance some of the simulations of subsection 6.4.4 were repeated with the LMS system identification replaced with RLS system identification and with the observation noise variance in the Kalman equaliser calculated using (6.5.1), (6.5.2) and (6.5.8). The results are illustrated in Figures 6.10 and 6.11 for a non-minimum phase and a minimum phase channel respectively. The most noticeable difference between these results and those of Figures 6.6(a) and 6.7(a) is the speed advantage of the RLS solution. For the minimum phase channel, the adaptive Kalman equaliser now converges to -53dB in 12 iterations compared with 50 iterations and for the non-minimum phase channel, it now converges to -53dB in 10 iteration compared with 40 iterations. Note also the close agreement between the

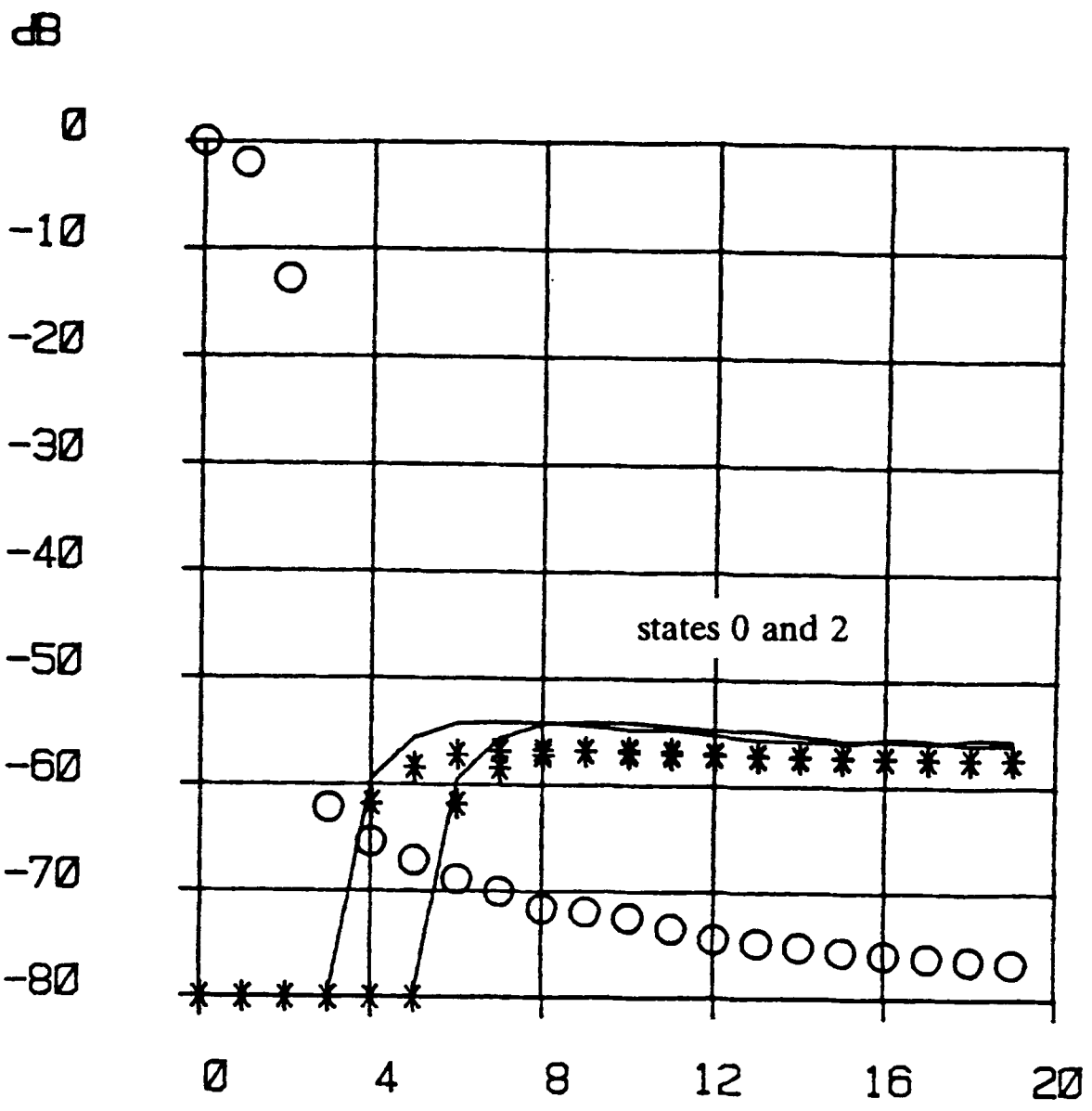
measured MSE of the states of the Kalman equaliser and the values predicted by the error covariance equations.

Figure 6.10 PERFORMANCE WITH RLS TRAINING
non - minimum phase (channel no. 1)
additive noise = -70.0 dB
ensemble = 100



KEY	
O	measured norm of system identification
* *	error covariance
~~	measured MSE of Kalman equaliser

Figure 6.11 PERFORMANCE WITH RLS TRAINING
minimum phase (channel no. 3)
additive noise = -70.0 dB
ensemble = 100



KEY	
O	measured norm of system identification
* *	error covariance
~~	measured MSE of Kalman equaliser

6.6 CONCLUSIONS

A new adaptive IIR linear equaliser [111] has been presented which is based on the Kalman filter structure of [18]. In the steady state this structure is a canonical form of the optimum IIR Wiener equaliser. The Kalman equaliser structure has been made adaptive by combining it with a LMS system identification algorithm and a novel technique, which both estimates the channel noise variance and compensates the Kalman filter for uncertainty in the channel impulse response. Simulation results have been presented showing the operation of this equaliser in the specific context of a communications channel suffering from minimum and non-minimum phase distortion.

Comparisons show that the convergence performance is roughly equivalent to a FIR equaliser which is trained with an RLS algorithm. However the order of the new filter is always lower than the FIR filter. This is particularly evident in the case of minimum phase channels. Unlike the FIR equaliser, the new filter produces estimates of the channel input at a range of lags up to the filter order. Hence the timing problem inherent in FIR based systems does not occur with this new filter structure. A comparison of the computational load of the proposed structure with an RLS FIR filter indicates that the former offers an advantage of 11-18% for a mimimum phase channel and 50-70% for non-minimum phase channels. Finally a method for improving the convergence performance of the adaptive Kalman equaliser has been described which involves replacing the LMS system identification algorithm with a RLS counterpart.

Chapter 7

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

7.1 GENERAL REMARKS

The subject of this thesis has been the design of algorithms for adaptive filtering. To this end both FIR and IIR filter structures have been considered. However, while it has been possible to look at FIR filters in a general application - independent manner in Chapters 2,3 and 4, IIR filters have only been examined in the specific application of linear equalisation in Chapters 5 and 6. This difference in approach reflects the maturity of adaptive FIR filtering, witnessed by the publication of textbooks on the subject, compared to adaptive IIR filtering. If a theme other than the title of adaptive filters unites Chapters 2,3 and 4 with Chapters 5 and 6 it is the concept of white input performance. In a FIR adaptive filter white input performance is obtained by contructing a whitening network (either explicitly or implicitly) at the input to the filter. In the adaptive IIR equaliser of section 6.3, white input performance is obtained by reconfiguring the equaliser so that the input to the adaptive filter is the transmitted signal which is usually white. The subsequent paragraphs summarise the conclusions which have been drawn in this thesis and highlight specific achievements.

7.2 SPECIFIC ACHIEVEMENTS

A broad selection of adaptive finite impulse response (FIR) filter algorithms was examined to assess their theoretical convergence performance and computational requirements. From this examination a classification system evolved in which the available algorithms were grouped into three classes according to convergence

performance and computational complexity. These three classes are: (i) stochastic gradient algorithms, (ii) self-orthogonalising algorithms and (iii) recursive least squares algorithms. Formerly classes (ii) and (iii) had been grouped together. Movement from class (i) through (ii) to (iii) improves convergence performance at the expense of increasing computational complexity.

The stochastic gradient algorithms are the least demanding computationally of all the adaptive FIR filter algorithms. Unfortunately they exhibit the poorest convergence performance since the convergence rate is dependent on the eigenvalues of the autocorrelation matrix associated with the input signal. The RLS algorithms, on the other hand, exhibit consistent fast convergence properties but are the most expensive computationally. Finally the transform domain or quasi-orthogonalising algorithms are less sensitive to the eigenvalue spread of the input autocorrelation matrix than the stochastic gradient algorithms. Thus they offer convergence performance that lies between the RLS and SG algorithms. However the sliding DFT algorithm is closer to the RLS algorithms than the SG algorithms in computational load.

Because of the difficulty in obtaining rigorous analytic results for the convergence properties of a broad selection of adaptive FIR filter algorithms, an experimental comparison was made using computer simulation. The results of these experiments confirm many of the key properties suggested by approximate analysis. In particular, the performance degradation of the SG algorithms when the input sequence is highly ill-conditioned, the fast consistent convergence of the LS algorithms, and the role of the quasi-orthogonalising algorithms as a compromise in performance between the LMS and the RLS algorithms.

Developments have been made in the area of block adaptive filtering. A unified approach to the BLMS algorithm has been presented which simplifies the application of efficient convolution algorithms other than the FFT (eg. the rectangular transform) to the construction of computationally efficient block adaptive filters. A significant contribution to the field has been the development of the self-orthogonalised block

adaptive filter which exhibits convergence performance characteristic of class (ii) in combination with computational complexity characteristic of class (i).

A closed form expression for the optimum IIR equalising filter was derived using Wiener filtering theory. This formulation highlighted the structure of the optimum IIR equaliser. A comparison of the MSE performance of FIR and IIR equalisers illustrated the inherent order advantage in using an IIR filter in this application. The adaptive IIR equaliser problem was shown to be equivalent to the identification of an ARMA plant which is embedded in an ARMAX process. Although the bias associated with RLS estimates can be avoided through resort to algorithms such as RIV, the MSE convergence of these algorithms is poorly understood and hence they cannot be considered to be robust solutions.

A new adaptive IIR linear equaliser [111] has been presented which is based on the Kalman filter structure of [18]. In the steady state this structure is a canonical form of the optimum IIR Wiener equaliser. The Kalman equaliser has been made adaptive by combining it with a LMS system identification algorithm and a novel technique, which both estimates the channel noise variance and compensates the Kalman filter for uncertainty in the channel impulse response. Comparisons show that the convergence performance is roughly equivalent to a FIR equaliser which is trained with an RLS algorithm. However the order of the new filter is always lower than the FIR filter. A comparison of the computational load of the proposed structure with an RLS FIR equaliser indicates that the former offers an advantage, which is dependent on the channel characteristics. Further a method for improving the convergence performance of the adaptive Kalman equaliser has been described which involves replacing the LMS system identification algorithm with a RLS counterpart.

7.3 LIMITATIONS AND FURTHER WORK

In Chapter 1, it was stated that adaptive filters find application in environments which are both stationary and non-stationary in nature. It is evident from subsequent chapters that the latter has not been considered in any detail. Performance in a non-stationary environment is merely implied from the convergence performance in a stationary environment. However other authors such as [112] present observations which indicate that in a continuously changing non-stationary environment such as a high frequency (HF) communications channel, this deduction is misleading and SG algorithms such as the LMS may outperform the RLS algorithm. A clear direction for future research is to study the performance of adaptive FIR filter algorithms in a continuously varying non-stationary environment using both analytic and experimental techniques.

The investigation of adaptive IIR filtering has been limited to the specific application of adaptive equalisation. The solution presented in Chapter 6 is application dependent and thus it is not believed that it could be applied to a general adaptive IIR filtering problem. However in the adaptive equaliser scenario the Kalman filter formulation makes it relatively straightforward to extend the algorithm of Chapter 6 to include decision feedback and fractionally spaced IIR equalisers. The question then arises as to how such equalisers compare in performance and complexity with more conventional decision feedback and fractionally spaced equaliser structures.

Appendix A

THE FAST KALMAN ALGORITHM

In [25] Ljung et al derived a recursion for the calculation of $\underline{k}(k)$, (2.4.14), which is an order of magnitude more efficient than (2.4.15). This recursion, which has since come to be known as the fast Kalman algorithm, depends fundamentally upon the shifting property of the vector $\underline{x}(k)$ and upon the use of forward and backwards least squares prediction to exploit this property. Before proceeding to the fast algorithm it is necessary to develop the forward and backward linear predictors and some of the variables associated with them.

An Nth order linear prediction $\hat{x}^f$ of the process x is given by the output of an N tap transversal filter.

$$\hat{x}^f(n) = \underline{a}^T \underline{x}(n-1)$$

where

$$\underline{a}^T = [\, a_0 \; a_1 \; \cdots \; a_{N-1} \,]$$

As with linear estimation, this linear prediction is considered to be least squares if the filter coefficient vector $\underline{a}$ is chosen to minimise a sum of squared errors cost function.

$$\sum_{n=0}^{k} (x(n) - \hat{x}^f(n))^2 \tag{A.1}$$

The value of $\underline{a}$ which minimises this cost function is given by

$$\underline{r}_{xx}(k-1)\, \underline{a}(k) = \underline{r}_x^f(k) \tag{A.2}$$

where

$$\underline{r}_x^f(k) = \sum_{n=0}^{k} \underline{x}(n-1)\, x(n) \tag{A.3}$$

The minimum value, $\alpha^f(k)$, of the cost function, (A.1), can then be calculated by combining (A.1) and (A.2).

$$\alpha^f(k) = r_o^f(k) - \underline{a}^T(k)\, \underline{r}_x^f(k) \tag{A.4}$$

where

$$r_o^f(k) = \sum_{n=0}^{k} x^2(n)$$

Again in common with last squares estimation, the tap vector $\underline{a}(k)$ may be generated recursively.

$$\underline{a}(k) = \underline{a}(k-1) + \underline{k}(k-1)\, e^f(k) \tag{A.5}$$

$$e^f(k) = x(k) - \underline{a}^T(k-1)\, \underline{x}(k-1) \tag{A.6}$$

Note that the recursion for $\underline{a}(k)$ requires the same gain vector $\underline{k}(k)$ as the recursion for $\underline{h}(k)$, (2.4.12).

The defining equations for backwards linear least squares prediction are analogous to those for forward linear least squares prediction. They are as follows: a backward prediction

$$\hat{x}^b(n-N) = \underline{b}^T\, \underline{x}(n)$$

a sum of squared errors cost function

$$\sum_{n=0}^{k} \left(x(n-N) - \hat{x}^b(n-N) \right)^2 \tag{A.7}$$

with solution

$$\underline{r}_{xx}(k)\, \underline{b}(k) = \underline{r}_x^b(k)$$

where

$$\underline{r}_{x}^{b}(k) = \sum_{n=0}^{k} \underline{x}(n)\, x(n-N)$$

The minimum value, $\alpha^{b}(k)$, of the cost function, (A.7), is

$$\alpha^{b}(k) = r_{o}^{b}(k) - \underline{b}^{T}(k)\, \underline{r}_{x}^{b}(k) \tag{A.8}$$

where

$$r_{o}^{b}(k) = \sum_{n=0}^{k} x^{2}(n-N)$$

a recursion for $\underline{b}(k)$

$$\underline{b}(k) = \underline{b}(k-1) + \underline{k}(k)\, e^{b}(k) \tag{A.9}$$

$$e^{b}(k) = x(k-N) - \underline{b}^{T}(k-1)\, \underline{x}(k)$$

The method used by Ljung et al to exploit the shifting property of $\underline{x}(k)$ involves first increasing the order of the system by 1 from order N to order (N + 1).

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

The vector $\bar{\underline{x}}(n)$ contains both the new data sample x(n) and the data sample x(n-N) which is disregarded. To highlight the effect of these two data points, the vector $\bar{\underline{x}}(n)$ is partitioned in two ways.

$$\bar{\underline{x}}(n) = \begin{bmatrix} x(n) \\ \underline{x}(n-1) \end{bmatrix} = \begin{bmatrix} \underline{x}(n) \\ x(n-N) \end{bmatrix} \tag{A.10}$$

This concept of increase in order and what might be called forward and backward partitions respectively is then applied to the matrix $\underline{r}_{xx}$.

$$\bar{\underline{r}}_{xx}(k) = \sum_{n=0}^{k} \bar{\underline{x}}(n)\, \bar{\underline{x}}^{T}(n)$$

$$= \begin{bmatrix} r_o^f(k) & \underline{r}_x^{f\,T}(k) \\ \underline{r}_x^f(k) & \underline{R}_{xx}(k-1) \end{bmatrix}$$

$$= \begin{bmatrix} \underline{R}_{xx}(k) & \underline{r}_x^b(k) \\ \underline{r}_x^{b\,T}(k) & r_o^b(k) \end{bmatrix}$$

This particular form assumes that the data is prewindowed. The matrix $\bar{\underline{R}}_{xx}$ may then be inverted by the use of two standard results from matrix algebra: the inversion rule for partitioned matrices [113] and the Sherman-Morrison identity. [39]

$$\bar{\underline{R}}_{xx}^{-1}(k) = \begin{bmatrix} \dfrac{1}{\alpha^f(k)} & \dfrac{\underline{a}^T(k)}{\alpha^f(k)} \\ \dfrac{\underline{a}(k)}{\alpha^f(k)} & \underline{R}_{xx}^{-1}(k-1) + \dfrac{\underline{a}(k)\,\underline{a}^T(k)}{\alpha^f(k)} \end{bmatrix}$$

$$= \begin{bmatrix} \underline{R}_{xx}^{-1}(k) + \dfrac{\underline{b}(k)\,\underline{b}^T(k)}{\alpha^b(k)} & \dfrac{\underline{b}(k)}{\alpha^b(k)} \\ \dfrac{\underline{b}^T(k)}{\alpha^b(k)} & \dfrac{1}{\alpha^b(k)} \end{bmatrix} \qquad \text{(A.11)}$$

The next step is to calculate the increased order gain vector $\bar{\underline{k}}(k+1)$ using the defining equation.

$$\bar{\underline{k}}(k+1) = \bar{\underline{R}}_{xx}^{-1}(k+1)\,\bar{\underline{x}}(k+1)$$

Application of the forward and backward forms of (A.10) and (A.11) in turn yields two expressions for $\bar{\underline{k}}(k+1)$

$$\bar{\underline{k}}(k+1) = \begin{bmatrix} 0 \\ \underline{k}(k) \end{bmatrix} - \frac{\epsilon^f(k+1)}{\alpha^f(k+1)} \begin{bmatrix} 1 \\ \underline{a}(k+1) \end{bmatrix} \qquad \text{(A.12)}$$

$$\bar{\underline{k}}(k+1) = \begin{bmatrix} \underline{k}(k+1) \\ 0 \end{bmatrix} - \frac{\epsilon^b(k+1)}{\alpha^b(k+1)} \begin{bmatrix} \underline{b}(k+1) \\ 1 \end{bmatrix} \qquad \text{(A.13)}$$

where

$$\epsilon^f(k+1) = x(k+1) - \underline{a}^T(k+1)\,\underline{x}(k)$$

and

$$\epsilon^b(k+1) = x(k-N+1) - \underline{b}^T(k+1)\,\underline{x}(k+1)$$

The variables ϵ^f and ϵ^b are known as the a posteriori forward and backward prediction errors respectively to distinguish them from e^f and e^b, the a priori errors [43]. Equations (A.12) and (A.13) give the required recursion ie. an expression for $\underline{k}(k+1)$ in terms of $\underline{k}(k)$. Using a backwards partition $\bar{\underline{k}}(k+1)$ may be rewritten.

$$\bar{\underline{k}}(k+1) = \begin{bmatrix} \underline{d}(k+1) \\ \delta(k+1) \end{bmatrix}$$

where

$$\delta(k) = \frac{-\epsilon^b(k+1)}{\alpha^b(k+1)}$$

and

$$\underline{d}(k+1) = \underline{k}(k+1) + \delta(k+1)\,\underline{b}(k+1) \tag{A.14}$$

Substitution for $\underline{b}(k+1)$ in (A.14) using (A.9) followed by some rearrangement gives

$$\underline{k}(k+1) = \frac{\underline{d}(k+1) - \delta(k+1)\,\underline{b}(k)}{1 + \delta(k+1)\,\epsilon^b(k+1)}$$

All that remains to complete the recursion is an expression for $\alpha^f(k+1)$. By definition

$$\alpha^f(k+1) = r_o^f(k+1) - \underline{a}^T(k+1)\,\underline{r}_x^f(k+1) \tag{A.15}$$

A recursion for $\underline{r}_x^f$ is obtained from (A.3).

$$\underline{r}^f(k+1) = \underline{r}^f(k) + \underline{x}(k)\,x(k+1) \tag{A.16}$$

Substitution for $\underline{a}(k+1)$ and $\underline{r}_x^f(k+1)$ in (A.15) yields

$$\alpha^f(k+1) = \alpha^f(k) + \epsilon^f(k+1)\, e^f(k+1) \ .$$

The complete algorithm is summarised in Table A.1.

Table A.1 THE FAST KALMAN ALGORITHM (PRE-WINDOWED)

$$e^f(k+1) = x(k+1) - \underline{a}^T(k)\,\underline{x}(k)$$

$$\underline{a}(k+1) = \underline{a}(k) + \underline{k}(k)\,e^f(k+1)$$

$$\epsilon^f(k+1) = x(k+1) - \underline{a}^T(k+1)\,\underline{x}(k)$$

$$\alpha^f(k+1) = \alpha^f(k) + \epsilon^f(k+1)\,e^f(k+1)\,.$$

$$\underline{\bar{k}}(k+1) = \begin{bmatrix} 0 \\ \underline{k}(k) \end{bmatrix} - \frac{\epsilon^f(k+1)}{\alpha^f(k+1)} \begin{bmatrix} 1 \\ \underline{a}(k+1) \end{bmatrix}$$

$$\underline{\bar{k}}(k+1) = \begin{bmatrix} \underline{d}(k+1) \\ \delta(k+1) \end{bmatrix}$$

$$e^b(k+1) = x(k-N+1) - \underline{b}^T(k)\,\underline{x}(k+1)$$

$$\underline{k}(k+1) = \frac{\underline{d}(k+1) - \delta(k+1)\,\underline{b}(k)}{1 + \delta(k+1)\,\epsilon^b(k+1)}$$

$$\underline{b}(k+1) = \underline{b}(k) + \underline{k}(k+1)\,e^b(k+1)$$

$$e(k+1) = y(k+1) + \underline{h}^T(k)\,\underline{x}(k+1)$$

$$\underline{h}(k+1) = \underline{h}(k) + \underline{k}(k+1)\,e(k+1)$$

Appendix B

CIRCULAR AND LINEAR CONVOLUTION

The circular convolution of the N point input vector

$$\underline{x}_r = [\, x(0)\, x(1) \;\cdots\; x(N-1)\,]^T$$

and an N-point impulse response vector

$$\underline{h} = [\, h_0\, h_1 \;\cdots\; h_{N-1}\,]^T$$

is the N-point output vector

$$\underline{y}^c = [\, y^c(0)\, y^c(1) \;\cdots\; y^c(N-1)\,]^T$$

where

$$y^c(n) = \sum_{j=0}^{N-1} h_j\, x(\,(n-j)\ mod\ N\,), \quad n = 0, 1, \cdots N-1.$$

The notation $j\ mod\ N$ indicates j modulo N ie. the remainder when j is divided by N. The circular convolution operation is indicated by the circular convolution operator, *.

$$\underline{y}^c = \underline{x}_r * \underline{h} \tag{B.1}$$

Equation (B.1) is computed efficiently using the FFT [7], RT [62] or NTT [63]. In general these three transform domain techniques may be defined in terms of two (N × M) matrices A_N and B_N and a (M × N) matrix C_N, where $M \geq N$ [85]. The subscript N is used to indicate an N-point circular convolution operation. The input vector $\underline{x}_r$ and the impulse response vector $\underline{h}$ are multiplied by the B_N and A_N matrices to form the M-point vectors $\underline{X}$ and $\underline{H}$ respectively.

$$\underline{X} = B_N\, \underline{x}_r$$

$$\underline{H} = A_N \, \underline{h}$$

Each element of the vector $\underline{X}$ is then multiplied by the corresponding element of the vector $\underline{H}$ to form the vector $\underline{Y}$.

$$Y_i = X_i \, H_i, \quad i = 0, 1, \cdots M-1$$

This point by point multiplication is denoted by the operator, $\times$.

$$\underline{Y} = \underline{X} \times \underline{H}$$

Finally the ouput vector $\underline{y}^c$ is formed by multiplying $\underline{Y}$ by the C matrix.

$$\underline{y}^c = C \, \underline{Y}$$

Turning to the linear convolution operation, the output y(n) of a N-tap FIR filter with impulse response vector $\underline{h}$ may be described by the vector inner product of two N-vectors

$$y(n) = \underline{x}^T(n) \, \underline{h} \ , \tag{B.2}$$

where

$$\underline{x}^T(n) = [\, x(n) \, x(n-1) \ \cdots \ x(n-N+1) \,] \ .$$

The output sequence $\{ y(n) \}$ is the linear convolution of the finite sequence $\{ h_i \}$, $0 \leq i < N-1$, with the infinite sequence $\{ x(n) \}$. For notational convenience the impulse response vector $\underline{h}$ is defined with its elements in time increasing order and the input vector $\underline{x}(n)$ is defined with its elements in time decreasing order. Thus the data ordering that is convenient for the representation of the filtering operation is not the same ordering that is convenient for the representation of circular convolution. To facilitate block processing it is also convenient to define an output N-vector $\underline{y}(k)$ which is contructed from N outputs of the FIR filter defined in (B.2).

$$\underline{y}(k) = [\, y(k)\; y(k-1) \;\cdots\; y(k-N+1)\,]^T$$

$$= \underline{\chi}(k)\, \underline{h}$$

The $(N \times N)$ matrix $\underline{\chi}(k)$, where

$$\underline{\chi}(k) = [\, \underline{x}(k)\; \underline{x}(k-1) \;\cdots\; \underline{x}(k-N+1)\,],$$

is symmetric.

Circular convolution may be used to perform linear convolution if it is used in conjunction with an overlap save or overlap add data sectioning technique [7]. Only overlap save will be considered here as it leads to more computationally efficient adaptive filter structures than overlap add [61]. To calculate the output vector $\underline{y}(k)$ a 2N-point circular convolution is required. First the impulse response vector of (B.2) is combined with an N-point zero vector to form a 2N-point impulse response vector.

$$\begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h} \tag{B.3}$$

The matrix I_N is an $(N \times N)$ identity matrix and the matrix O_N is an $(N \times N)$ matrix with all zero elements. Then a 2N-point input vector is formed from $\underline{x}(k)$ and $\underline{x}(k-N)$.

$$\begin{bmatrix} T_N\, \underline{x}(k-N) \\ T_N\, \underline{x}(k) \end{bmatrix} \tag{B.4}$$

The (N × N) time reversal matrix, T_N, has 1's on the secondary diagonal and zeros elsewhere. Finally the 2N-point circular convolution of the input vector of (B.4) and the impulse response vector of (B.3) is calculated. The elements of the N-point vector $\underline{y}(k)$ can be recovered from the resultant 2N-point output vector by multiplying the latter by the window matrix, $[\, O_N\; I_N\,]$. The overlap save technique is summarised by the following vector matrix equation,

$$\underline{y}_r(k) = [\, O_N \;\; I_N \,] \left\{ \begin{bmatrix} T_N \; \underline{x}(k-N) \\ T_N \; \underline{x}(k) \end{bmatrix} * \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h} \right\} \tag{B.5}$$

where

$$\underline{y}_r(k) = [\, y(k-N+1) \;\; \cdots \;\; y(k-1) \; y(k) \,]^T .$$

Since circular covolution is defined in terms of vectors in time increasing order both overlap add and overlap save techniques produce an output vector $\underline{y}_r(k)$ which is the time reverse of the vector $\underline{y}(k)$.

The circular convolution operation indicated in (B.5) can be computed efficiently using any of the 2N-point circular convolution algorithms which are defined in general by two $(2N \times M)$ matrices A_{2N} and B_{2N} and a $(M \times 2N)$ matrix C_{2N}, where $M \geq 2N$. The output vector $\underline{y}(k)$ is obtained from $\underline{y}_r(k)$ by multiplying the latter by the $(N \times N)$ reversal matrix T_N.

$$\begin{aligned} \underline{y}(k) &= T_N \; \underline{y}_r(k) \\ &= [\, O_N \;\; T_N \,]\, C_{2N} \, \{ \underline{X}(k) \times \underline{H} \} \end{aligned}$$

where

$$\underline{X}(k) = B_{2N} \begin{bmatrix} T_N \; \underline{x}(k-N) \\ T_N \; \underline{x}(k) \end{bmatrix}$$

and

$$\underline{H} = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{h}.$$

Appendix C

RELEVANT PUBLICATIONS

[1] Mulgrew, B., "The Application of Kalman Filtering to Channel Equalisation," *IEE Saraga Colloquium on Electronic Filters*, London, 21st May 1984.

[2] Mulgrew, B. and Cowan, C.F.N., "Kalman Filter Techniques in Adaptive Filtering," *IEE Colloquium on Adaptive Filters*, London, 10th Oct. 1985.

[3] * Mulgrew, B. and Cowan, C.F.N., "An Adaptive IIR Equalizer: A Kalman Filter Approach," *International Conference on Acoustics Speech and Signal Processing*, Tokyo, April 1986.

[4] * Panda, G., Mulgrew, B., Cowan, C.F.N. and Grant, P.M., "On the Rectangular Transform Approach to BLMS Adaptive Filtering," *International Conference on Acoustics Speech and Signal Processing*, Tokyo, April 1986.

[5] Mulgrew, B. and Cowan, C.F.N., "An Adaptive Infinite Impulse Response Equaliser," *U.K. Patent Application*, no. 8523286, Sept. 1985.

[6] * Panda, G., Mulgrew, B., Cowan, C.F.N. and Grant, P.M., "A Self Orthogonalising Efficient Block Adaptive Filter," *IEEE Transactions Acoustics Speech and Signal Processing*. vol. ASSP-34, no.6, pp 1573-1582, Dec. 1986.

[7] Mulgrew, B., "Kalman Filter Techniques in Adaptive Filtering," accepted for publication in *IEE Proceedings Part F*.

[8] Mulgrew, B., Cowan, C.F.N., "An Adaptive Kalman Equaliser: Structure and Performance," accepted for publication in *IEEE Transactions Acoustics Speech and Signal Processing*.

[9] Panda, G., Mulgrew, B., Cowan, C.F.N. and Grant, P.M., "Rectangular Transform Approach for Block Least Mean Squares Adaptive Filtering," accepted for publication in *IEEE Transactions Acoustics Speech and Signal Processing*.

* Reprinted at back of thesis.

REFERENCES

1. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., (1975).

2. N. Wiener, *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, John Wiley & Sons, (1962).

3. R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME, Journal of Basic Engineering*, pp. 35-45 (March 1960).

4. R. E. Kalman and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," *Trans. ASME, Journal of Basic Engineering*, pp. 95-108 (March 1961).

5. B. D. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, (1979).

6. P. F. Adams, "Adaptive Filters in Telecommunications," *Adaptive Filters*, Prentice-Hall, (1985).

7. A. V. Oppenhein and R. W. Schafer, *Digital Signal Processing*, Englewood Cliffs, (1975).

8. S. U. H. Qureshi, "Adaptive Equalisation," *Proceedings IEEE* **73**(9) pp. 1349-1387 (Sept. 1985).

9. J. G. Proakis, *Digital Communications*, McGraw-Hill, (1983).

10. J. Makhoul, "Linear Prediction a Tutorial Review," *Proceedings IEEE* **63**(4) pp. 561-580 (April 1975).

11. S. M. Kay and S. L. Marple, "Spectrum Analysis - A Modern Perspective," *Proceedings IEEE* **69**(11) pp. 1380-1419 (Nov. 1981).

12. A. H. Gray and J. D. Markel, *Linear Prediction of Speech*, Springer Verlag, (1976).

13. J. R. Zeidler, E. H. Satorius, D. M. Chabries, and H. T. Wexler, "Adaptive Enhancement of Multiple Sinsusoids in Uncorrelated Noise," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-26** pp. 240-254 (June 1978).

14. G. A. Clark, S. K. Mitra, and S. R. Parker, "Block Implementations of Adaptive Digital Filters," *IEEE Trans. Circuits and Systems* **CAS-28** pp. 584-592 (June 1981).

15. C. F. Cowan and P. M. Grant, *Adaptive Filters*, Prentice-Hall, (1985).

16. M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Kluwer Academic Publishers, (1984).

17. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, (1984).

18. R. E. Lawrence and H. Kaufman, "The Kalman Filter for the Equalization of a Digital Communications Channel," *IEEE Trans. Communucations Technology* **19**(6) pp. 1137-1141 (Dec. 1971).

19. B. Widrow, P. E. Mantey, L. J. Griffiths, and B. B. Goode, "Adaptive Antenna Systems," *Proceeding IEEE* **55** pp. 2143-2159 (Dec. 1967).

20. B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *1960 IRE WESCON Conv. Record*, pp. 96-104 (1960).

21. B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory*, ed. R. E Kalman and N. DeClaris, Holt,Rinehart & Winston, New York (1971).

22. N. Levinson, "The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction," *J. Math. Phys* **25** pp. 261-278 (Jan. 1947).

23. P. Butler and A. Cantoni, "Non-Iterative Automatic Equalization," *IEEE Trans. Communications* **COM-23**(6) pp. 621-633 (June 1975).

24. G. C. Goodwin and R. L. Payne , *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, (1977).

25. L. Ljung, M. Morf, and D. Falconer, "Fast Calculation of Gain Matrices for Recursive Estimation Schemes," *International Journal of Control* **27**(1) pp. 1-19 (Jan. 1978).

26. G. Ungerboeck, "Theory on the Speed of Convergence in Adaptive Equalisers for Digital Communication," *IBM J. of Research and Develop.* **16**(6) pp. 546-555 (Nov. 1972).

27. A. Feuer and E. Weinstein, "Convergence Analysis of LMS Filters with Uncorrelated Gaussian Data," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-33**(1) pp. 220-230 (Feb. 1985).

28. A. Feuer, "Performance Analysis of the Block Least Mean Square Algorithm," *IEEE Trans. Circuits and Systems* **CAS-32**(9)(Sept. 1985).

29. S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform Domain LMS Algorithms," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(3) pp. 609-615 (June 1983).

30. S. Y. Kung and Y. H. Hu, "A Highly Concurrent Algorithm and Piplined Architecture for Solving Toeplitz Systems," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(1) pp. 66-75 (Feb. 1983).

31. D. Yucel, N. Tepedelenlioglu, and Y. Tanik, "A Fast Non-Iterative Method for Adaptive Channel Equalisation," *IEEE Trans. Communication* **COM-31**(7) pp. 922-927 (July 1983).

32. J. S. Bendat and A. G. Piersol, *Measurement and Analysis of Random Data*, pp. 290-291 John Wiley, (1966).

33. G. M. Jenkins and D. G. Watts, *Spectral Analysis and its Applications*, pp. 171-189 Holden-day, (1968).

34. G. Strang, *Linear Algebra and its Applications*, p. Academic Press (1980). 2nd edition

35. B. Friedlander, "Lattice Filters for Adaptive Processing," *Proc. IEEE* **70**(8) pp. 829-867 (August 1982).

36. B. Friedlander, M. Morf, T. Kailath, and L. Ljung, "New Inversion Formulas for Matrices Classified in Terms of their Distance from Toeplitz," *Linear Algebra and its Applications*, pp. 31-60 Elsevier, (Oct. 1979).

37. S. L. Marple, "Efficient Least Squares System Identification," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-29**(1) pp. 62-73 (Feb. 1981).

38. J. M. Cioffi, "The Block-Processing FTF Adaptive Algorithm," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-34**(1) pp. 77-90 (Feb. 1986).

39. Broyden, *Basic Matrices*, The Macmillan Press Ltd., (1975).

40. D. Godard, "Channel Equalization Using a Kalman Filter for Fast Data Transmission," *IBM Journal Res. Develop.* **18**(3) pp. 267-273 (May 1974).

41. B. Porat, "Second-Order Equivalence of Rectangular and Exponential Windows in Least Squares Estimation of Gaussian Autoregressive Processes," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-33** pp. 1209-1212 (Oct. 1985).

42. C. C. Halkias, G. Carayannis, J. Dologlou, and D. Emmanoulopoulas, "A New Generalised Recursion for the Fast Computation of the Kalman Gain to Solve the Covariance Equations," *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 1760-1763 (May 1982).

43. G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A Fast Sequential Algorithm for Least-Squares Filtering and Prediction," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31** pp. 1394-1983 (Dec. 1983).

44. J. M. Cioffi and T. Kailath, "Fast, Recursive-Least-Squares Transversal Filters for Adaptive Filtering," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-32**(2) pp. 304-337 (April 1984).

45. M. S. Mueller, "Least Squares Algorithms for Adaptive Equalisers," *Bell Systems Technical Journal* **60**(8) pp. 1905-1925 (Oct. 1981).

46. D. W. Lin, "On a Digital Implementation of the Fast Kalman Algorithm," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-32**(5) pp. 998-1005 (Oct. 1984).

47. M. L. Honig, "Recursive Fixed Order Covariance Least Squares Algorithms," *Bell System Tech. Journal* **62** (10, part 1) pp. 2916-2922 (1983).

48. J. M. Cioffi and T. Kailath, "Windowed Fast Transversal Adaptive Algorithm with Normalisation," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-33**(3) pp. 607-625 (June 1985).

49. A. H. Gray and J. D. Markel, "Digital Lattice and Ladder Filter Synthesis," *IEEE Trans. Audio and Electroacoustics* **AU-21**(6) pp. 491-500 (1973).

50. A. H. Gray and J. D. Markel, "A Normalised Digital Filter Structure," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-23**(3) pp. 268-277 (1975).

51. J. M. Turner, "Recursive Least-Squares Estimation and Lattice Filters," pp. 91-144 in *Adaptive Filters*, ed. C. F. Cowan and P. M. Grant, Prentice-Hall, Englewood Cliffs, New Jersey (1985).

52. D. T. L. Lee, M. Morf, and B. Friedlander, "Recursive Least Squares Ladder Estimation Algorithms," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-29**(3) pp. 627-641 (June 1981).

53. M. J. Shensa, "Recursive Least Squares Lattice Algorithms - A Geometrical Approach," *IEEE Trans. Automatic Control* **AC-26**(3) pp. 675-702 (June 1981).

54. H. M. Ahmed, M. Morf, D. T. Lee, and P. H. Ang, "A Ladder Filter Speech Analysis Chip Set Utilizing Co-ordinate Rotation Arithmetic," *Proceedings IEEE International Symposium on Circuits and Systems* **3** pp. 737-741 (April 1981).

55. B. Porat, B. Friedlander, and M. Morf, "Square Root Covarience Ladder Algorithms," *IEEE Trans. Automatic Control* **AC-27**(4) pp. 813-829 (Aug. 1982).

56. C. Samson and V. U. Reddy, "Fixed Point Error Analysis of the Normalised Ladder Algorithm," *IEEE Trans. Acoutics Speech and Signal Processing* **ASSP-31**(5) pp. 1177-1191 (Oct. 1983).

57. H. Lev-Ari and T. Kailath, "Lattice Filtering Parameterization and Modelling of Non-Stationary Processes," *IEEE Trans. Information Theory* **IT-30**(1) pp. 2-16 (Jan. 1984).

58. B. Friedlander, "Adaptive Algorithms for Finite Impulse Response Filters," in *Adaptive Filters*, ed. C. F. Cowan and P. M. Grant, Prentice-Hall, Englewood Cliffs, New Jersey (1985).

59. L. Ljung, "Analysis of a Generalised Recursive Predicton Algorithm," *Automatica* **17**(1) pp. 89-99 (Jan. 1981).

60. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, (1983).

61. G. A. Clark, S. R. Parker, and S. K. Mitra, "A Unified Approach to Time- and Frequency-Domain Realisation of FIR Adaptive Digital Filters," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(5) pp. 1073-1083 (Oct. 1983).

62. R. C. Agarwal and J. W. Cooley, "New Algorithms for Digital Convolution," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-25** pp. 392-410 (Oct. 1977).

63. R. C. Agarwal and C. S. Burrus, "Fast One Dimensional Algorithms by Multidimensional Techniques," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-22** pp. 1-10 (Feb. 1974).

64. C. W. Therrien, "On the Relationship Between Triangular Matrix Decomposition and Linear Prediction," *Proceedings IEEE* **71**(12) pp. 1459-1460 (Dec. 1983).

65. L. J. Griffiths, "A Continuously Adaptive Filter Implemented as a Lattice Filter Structure," *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 683-686 (1977).

66. M. L. Honig and D. G. Messerschmitt, "Convergence Properties of an Adaptive Digital Lattice Filter," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-29** pp. 642-653 (June 1981).

67. R. P. Bitmead and B. D. O. Anderson, "Adaptive Frequency Sampling Filters," *IEEE Trans. Circuits and Systems* **CAS-28**(6) pp. 524-535 (June 1981).

68. K. M. Wong and Y. G. Jan, "Adaptive Walsh Equaliser for Data Transmission," *Proceedings IEE Part F* **130**(2) pp. 153-160 (March 1983).

69. B. Widrow, J. McCool, and M. Ball, "The Complex LMS Algorithm," *Proceedings IEEE* **63**(4) pp. 719-720 (April 1975).

70. M. L. Honig, "Convergence Models For Lattice Joint Process Estimators and Least Squares Algorithms," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(2) pp. 415-425 (April 1983).

71. M. L. Honig, "Echo Cancellation of Voiceband Data Signals Using Recursive Least Squares and Stochastic Gradient Algorithms," *IEEE Trans. Communications* **COM-33**(1) pp. 65-73 (Jan. 1984).

72. L. L.Horowitz and K. D. Senne, "Performance Advantage of Complex LMS for Controlling Narrow-Band Adaptive Arrays," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-29**(3) pp. 722-736 (June 1981).

73. J. J. Werner, "An Echo-Cancellation Based 4800 Bits/s Full-Duplex DDD Modem," *IEEE Journal on Selected Areas in Communications* **SAC-2**(5) pp. 722-730 (Sept. 1984).

74. G. Panda, B. Mulgrew, C. F. N. Cowan, and P. M. Grant, "A Self Orthogonalised Efficient Block Adaptive Filter," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-34** pp. 1573-1582 (Dec. 1986).

75. K. H. Muller and D. A. Spaulding, "A New Rapidly Converging Equalisation Technique for Synchronous Data Communication," *Bell System Technical Journal* **54**(2) pp. 369-406 (Feb. 1975).

76. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proc. IEEE* **64**(8) pp. 1151-1162 (Aug. 1976).

77. R. D. Gitlin and F. R. Magee, "Self-Orthogonalising Adaptive Equalisation Algorithms," *IEEE Trans. Communications* **COM-23**(July 1977).

78. K. H. Muller, "A New Fast-Converging Mean-Square Algorithm for Adaptive Equalisers with Partial Response Signalling," *Bell System Technical Journal* **54**(1) pp. 143-153 (Jan. 1975).

79. J. W. Cooley and J. W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series," *Math. Comput.* **19** pp. 297-301 (April 1965).

80. D. Mansour and A. H. Gray, "Unconstrained Frequency-Domain Adaptive Filter," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-30**(5) pp. 726-734 (Oct. 1982).

81. G. Picchi and G. Prati, "Self-Orthogonalised Adaptive Equalisation in the Frequency Domain," *IEEE Trans. Communications* **COM-32**(4) pp. 371-379 (April 1984).

82. E. R. Ferrara, "Fast Implementation of LMS Adaptive Filters," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-28**(4) pp. 474-475 (Aug. 1980).

83. R. C. Agarwal and C. S. Burrus, "Fast Convolution using Fermat Number Transforms with Applications to Digital Filtering," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-22** pp. 87-97 (April 1974).

84. J. C. Lee, B. K. Min, and M. Suk, "Realization of Adaptive Digital Filters using the Fermat Number Transform," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-33**(4) pp. 1036-1039 (Aug. 1985).

85. G. Panda, B. Mulgrew, C. F. N. Cowan, and P. M. Grant, "On the Rectangular Transform Approah for BLMS Adaptive Filtering," *International Conference on Acoustics Speech and Signal Processing* **4** pp. 2955-2958 (1986).

86. R. Kumar, "A Fast Algorithm for Solving Toeplitz System of Equations," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-33**(1) pp. 254-267 (Feb. 1985).

87. R. C. Singleton, "An Algorithm for Computing the Mixed Radix Fourier Transform," *IEEE Trans. Audio Electroacoust.* **AU-17** pp. 93-103 (June 1969).

88. M. Morf, "Doubling Algorithm for Toeplitz and Related Equations," *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 956-959 (1980).

89. F. G. Gustavson and D. Y. Y. Yun, "Fast Algorithms for Rational Hermite Approximation and Solution of Toeplitz Systems," *IEEE Trans Circuits and Systems* **CAS-26** pp. 750-755 (Sept. 1979).

90. S. Haykin, "Radar Signal Processing," *IEEE ASSP Magazine*, pp. 2-18 (April 1985).

91. J. R. Treichler, "Adaptive Algorithms for Infinite Impulse Response Filters," *Adaptive Filters*, Prentice-Hall, (1985).

92. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, (1985).

93. C. R. Johnson, "Adaptive IIR Filtering: Current Results and Open Issues," *IEEE Trans. Information Theory* **IT-30**(2) pp. 237-250 (March 1984).

94. C. A. Belfiore and J. H. Park, "Decision Feedback Equalisation," *Proceedings IEEE* **67**(8) pp. 1143-1156 (Aug. 1979).

95. J. J. O'Reilly and A. M. de Oliveira Duarte, "Error Propagation in Decision Feedback Receivers," *IEE Proceedings Part F* **132**(7) pp. 561-566 (Dec. 1985).

96. M. D. Srinath and P. K. Rajasekaron, *An Introduction to Statistical Processing with Applications*, John Wiley & Sons, (1979).

97. D. T. Sherwood and L. R. Weill, "Kalman Filter Deconvolution of a Non-Minimum Phase Multipath Channel," *Conf. Records 17th Asilomar Conf. on Circuits and Systems and Computers* , pp. 516-520 (1983).

98. D. G. Manolakis, N. Kalouptsidis, and G. Carayannis, "Fast Algorithm for Discrete Time Wiener Filters and Optimum Lag," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(1) pp. 168-179 (Feb. 1983).

99. B. Friedlander, "A Modified Prefilter for Some Recursive Parameter Estimation Algorithms," *IEEE Trans. Automatic Control* **AC-27**(1) pp. 232-235 (Feb. 1982).

100. T. Soderstrom and P. Stoica, "Comparison of some Instrumental Variable Methods - Consistency and Accuracy Aspects," *Automatica* **17** pp. 101-115 (Jan. 1981).

101. B. Friedlander, "Instrumental Variable Methods for ARMA Spectral Estimation," *IEEE Trans. Acoustics Speech and Signal Processing* **ASSP-31**(2) pp. 404-415 (April 1983).

102. L. Ljung, "Asymptotic Behaviour of the Extended Kalman Filter as a Parameter Estimator for Linear Systems," *IEEE Trans. Automatic Control* **AC-24**(1) pp. 36-58 (Feb. 1979).

103. S. B. Kleibanov, V. B. Privalski, and I. V. Time, "Kalman Filter for Equalization of Digital Communications Channel," *Automation and Remote Control* **35, part 1** pp. 1097-1102 (1974). (English Translation)

104. J. W. Mark, "A Note on the Modified Kalman Filter for Channel Equalization," *Proc. IEEE* **6**(4) pp. 481-482 (April 1973).

105. E. D. Mese and G. Corsini, "Adaptive Kalman Filter Equaliser," *Electronic Letters* **16**(14) pp. 547-549 (3rd July 1980).

106. A. Luvison and G. Pirani, "Design and Performance of an Adaptive Kalman Receiver for Synchronous Data Transmission," *IEEE Trans. Aerospace and Electronic Systems* **AES-15**(5) pp. 635-648 (Sept. 1979).

107. A. Luvison and G. Pirani, "A State-Variable Approach to the Equalization of Multilevel Digital Signals," *CSELT Report Tecnici* **2** pp. 3-22 (April 1974).

108. S. Benedetto and E. Biglieri, "On Linear Receivers for Digital Transmission Systems," *IEEE Trans. Communications* **COM-22**(9) pp. 1205-1215 (Sept. 1974).

109. H. Heffes, "The Effect of Erroneous Models on the Kalman Filter Response," *IEEE. Trans. Automatic Control* **AC-11**(3) pp. 541-543 (July 1966).

110. M. J. Grimble, "Optimal Control of Linear Uncertain Multivariable Stochastic Systems," *IEE Proceedings Part D* **129**(6) pp. 263-270 (Nov. 1982).

111. B. Mulgrew and C. F. N. Cowan, "An Adaptive Infinite Impulse Response Equaliser," *U. K. Patent Application*, (8523286)(Sept. 1985).

112. A. P. Clark and R. Harun, "Assessment of Kalman Filter Channel Estimators for an HF Radio Link," *IEE Proceedings Part F* **133** pp. 513-521 (Oct. 1986).

113. F. Ayres, "Theory and Problems of Matrices," *Schaum's Outline Series*, McGraw Hill, (1974).

AN ADAPTIVE IIR CHANNEL EQUALISER: A KALMAN FILTER APPROACH

Bernard Mulgrew, and Colin F.N.Cowan,
Department of Electrical Engineering,
University of Edinburgh,
Mayfield Road,
Edinburgh, EH9 3JL,
SCOTLAND.

ABSTRACT

Using discrete time Wiener filtering theory a closed form for the optimum mean-square error (MSE) infinite impulse response (IIR) linear equaliser is derived. The minimum phase spectral factorisation, which is an integral part of the derivation of the IIR equaliser, may be circumvented through the use of a Kalman equaliser such as that originally proposed by Lawrence and Kaufman. The structure is made adaptive by using a system identification algorithm operating in parallel with a Kalman equaliser. In common with Luvison & Pirani, a least mean squares (LMS) algorithm was chosen for the system identification because the input to the channel is white. A new technique is introduced which both estimates the variance of channel noise and compensates the Kalman filter for errors in the estimate of the channel impulse response.

1. INTRODUCTION

A digital communications channel with inter-symbol interference may be modelled by an equivalent discrete time transversal filter with additive white noise [1]. Thus the channel output $x(k)$ may be written in terms of the channel input $s(k)$ and the noise $n(k)$ as,

$$x(k) = \underline{h}^T \underline{s}(k) + n(k) \qquad (1)$$

where $\underline{h}$ is the M point impulse response vector and the vector $\underline{s}(k)$ contains the last M inputs to the channel.

$$\underline{s}^T(k) = [\ s(k)\ s(k-1)\ \ldots\ s(k-M+1)\ ]$$

The superscript T denotes transpose. A linear equaliser consists of a linear filter section followed by a non-linear slicer or decision circuit. The linear filter is designed to minimise the error between the filter output and the input to the channel. A mean-square error cost function, L, is usually used.

$$L = E[\ (\ s(k-d) - \hat{s}(k-d)\ )^2\ ]$$

The superscript $\hat{}$ denotes an estimate. The delay term d, $d \geq 0$, allows for the possibility of fixed lag smoothing. The non-linear slicer makes decisions on a symbol by symbol basis. From the above it is clear that the major design effort for this form of equaliser is concentrated on the linear filter section, where linear estimation theory is applied with a view to minimising the mean-square error L at the input to the decision circuit.

2. OPTIMUM LINEAR ESTIMATION

Usually the linear filter takes the form of a FIR transversal filter of order M-1 [1]. In deriving the optimum transversal equaliser the mean-square error L is minimised subject to the constraint that the impulse response is finite, causal and stable. If this condition is replaced with a less stringent one, i.e. the filter should be causal and stable, the solution to the minimisation problem is provided by the IIR Wiener filter. The optimum IIR filter is defined in terms of the z transform, G(z), of its impulse response sequence $\{\ g(k)\ \}$ [2].

$$G(z) = \frac{1}{\left(\ H(z)H(z^{-1})\sigma_s^2 + \sigma_n^2\ \right)^+} \left[\frac{z^{-d}\ H(z^{-1})\ \sigma_s^2}{\left(\ H(z)H(z^{-1})\sigma_s^2 + \sigma_n^2\ \right)^-} \right]_+$$

where $E[\ n(k)\ n(k+l)\ ] = \sigma_n^2\ \delta(l)$

$$E[\ s(k)\ s(k+l)\ ] = \sigma_s^2\ \delta(l)$$

$\delta(l)$ is the Dirac delta function. The notation $[.]_+$ represents the causal part of [.]. The power spectrum (.) is factorised ,

$$(.) = (.)^+\ (.)^-$$

where the term with the superscript + is minimum phase.

In the absence of noise and with the input signal power normalised to unity the IIR Wiener equaliser may be expressed in terms of the zeros of the channel frequency response H(z), which is assumed to have P zeros inside the unit circle at z_i^+ and Q zeros outside the unit circle at z_j^-.

$$G(z) = W(z)\ [\ B(z)\ ]_+$$

where

$$W(z) = \frac{z^{M-1}}{h_0 \prod_{i=1}^{P} \left(z - z_i^+ \right) \prod_{j=1}^{Q} \left(-z + \frac{1}{z_j^-} \right)}$$

40. 3. 1

ICASSP 86, TOKYO CH2243-4/86/0000-2099 $1.00 © 1986 IEEE

and

$$B(z) = \frac{\prod_{j=1}^{Q}\left[-z + \frac{1}{z_j^-}\right]}{z^d \prod_{j=1}^{Q}\left[z - z_j^-\right]}$$

The transfer function W(z) is of an autoregressive noise whitening filter of order M-1 . The transfer function $[\ B(z)\ ]_+$ represents a FIR filter of order d.

3. AN ADAPTIVE KALMAN EQUALISER

While the IIR Wiener filter exhibits a distinct performance advantage over a FIR filter of the same order when used to equalise a known channel, significant problems are encountered with the former when the channel is unknown or time-varying and an adaptive filter structure is required. An initial approach to the problem might be to postulate an adaptive algorithm such as those suggested in [3] that would recursively estimate the coefficients of the IIR Wiener filter in the same manner as the LMS algorithm [3] is used to estimate the coefficients of the FIR Wiener filter. However, in the process of adaptation, there is a finite probability that the poles of the filter will move outside the unit circle in the z-plane. This can lead to instability if the poles remain outside the unit circle for an extended period [3]. An alternative approach is to follow Lawrence and Kaufman [4] and use a Kalman filter to directly estimate the vector $\underline{s}$. For unknown channels the state vector is then augmented to include the channel impulse response. The result of this formulation is a non-linear estimation problem to which an extended Kalman filter (EKF) is applied. However the convergence of the EKF is not guaranteed [5]. The structure that is considered here consists of a system identification algorithm in parallel with a Kalman equaliser (Fig.1). In essence it is the combination of the adaptive Kalman filter structure of [6] with the non-adaptive smoothing filter of [7]. but unlike [6], the full form of the Kalman filter gain equations are used in order to exploit the full capacity of the filter to handle nonstationary environments. This combination produces an adaptive structure which is capable of equalising both minimum and non-minimum phase channels.

The adaptive equaliser, (Fig.1), operates in the following manner. During an initial training period a predetermined sequence is transmitted and the system identification algorithm forms an estimate of the channel impulse response vector. The estimate is passed to the Kalman filter which is then initialised and transmission of data begins. Because a training sequence is employed, initialisation of the Kalman filter is exact. The state vector at time zero, consists of the last d samples of the training sequence. The error covariance matrix is the zero matrix since the state vector is known with probability one. The Kalman filter forms estimates of the signal at all lags up to lag d. In addition the variance of these estimates is available on the leading diagonal of the error covariance matrix. Thus once the Kalman filter has reached a steady state, the element of the state vector which achieves a predetermined performance bound is used as the input to the decision circuit or slicer. During data transmission it is possible to operate the adaptive equaliser in decision directed mode. In this mode, the output of the slicer is used as an input to the system identification algorithm. Thus, provided the slicer makes correct decisions, the system identification algorithm and hence the adaptive equaliser will be capable of tracking time varying channels. When the channel is non-minimum phase adequate

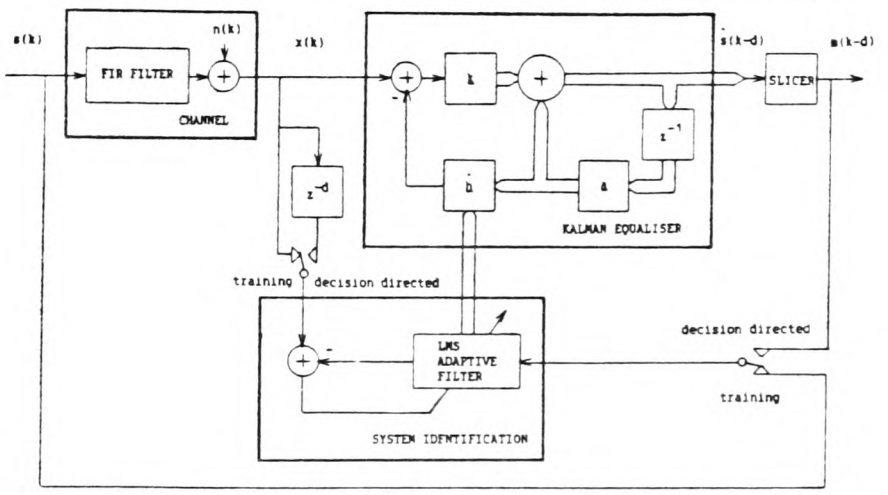


Fig.1: An Adaptive Kalman Equaliser

performance may only be achieved by using a fixed lag d. Under these conditions the output of the slicer will represent the channel input d samples ago. In order that the system identification algorithm produce correct results a delay d must be introduced between the channel output and the training input to the system identification algorithm.

3.1 System Identification

Since the channel input sequence is white, its autocorrelation matrix is diagonal with equal eigenvalues. Under these conditions a stochastic gradient LMS algorithm achieves its best performance and hence it is well suited to perform the system identification task. Of course, a recursive least squares (RLS) adaptive transversal filter [3] converges faster than the LMS algorithm even under white input conditions [8]. However this performance gain is achieved at the expense of increased complexity and since a Kalman algorithm has already been postulated to perform the equalisation task, it is natural to choose the simpler LMS algorithm for initial study.

Standard theoretical analysis of the LMS algorithm [9] yields the following equation which summarises the convergence properties when the input signal is white and the step size μ is set for fastest convergence.

$$\varrho(k) = \left(1 - \frac{1}{M} \right) \varrho(k-1) + \frac{\sigma_n^2}{M} \qquad (2)$$

The norm $\varrho(k)$ is defined as

$$\varrho(k) = E[(\underline{\hat{h}}(k) - \underline{h})^T (\underline{\hat{h}}(k) - \underline{h})]$$

and the M-vector $\underline{\hat{h}}$ is the estimated impulse response of the channel. Although the analysis relies on assumptions that are often invalid, it produces figures which agree well with experimental results and hence is useful in the understanding and operation of the adaptive Kalman equaliser. Equation (2) provides two important results: (i) the fastest convergence rate that can be obtained using the LMS algorithm is

$$10 \log_{10} \left(\frac{M}{M-1} \right) \quad \text{dB/iteration}$$

and (ii) the final value of the norm under these conditions is determined by the variance of the noise i.e.

$$\lim_{k \to \infty} \varrho(k) = \sigma_n^2$$

3.2 Model Uncertainty

The two parameters in the channel model (1) of which there is uncertainty or imperfect knowledge are the channel impulse response vector and the additive noise variance. If the noise variance used in the Kalman filter is greater than the actual noise variance, then the diagonal elements of the error covariance matrix are an upper bound on the MSE performance of the filter [2]. This result suggests a simple solution to the problem of uncertainty in the observation noise i.e. the noise variance in the Kalman filter should be set to a maximum or worst case value that is expected in a particular application. There are however two disadvantages with this strategy: (i) the performance of the Kalman filter will be degraded if only slightly, and (ii) the diagonal elements of the error covariance matrix will be greater than the actual MSE and hence will not be useful as indicators of the equaliser performance. An alternative solution is to estimate the noise variance directly from the channel. Before considering this solution, the effect of uncertainty in the channel tap vector $\underline{h}$ on the performance of the Kalman filter will be considered briefly.

In the previous section it was shown that the lower bound on the norm is the additive noise variance, when the convergence factor μ is set for fastest convergence. Experiment indicates that this is the point at which the uncertainty in the tap vector $\underline{h}$ starts to degrade the performance of the Kalman filter (Fig. 2). For this experiment the estimated tap vector used in the Kalman filter was formed by adding a suitably scaled random noise term to each coefficient of the channel tap vector. The technique that is now presented provides a method for both estimating the additive noise variance and compensating the Kalman filter for the uncertainty in the channel impulse response vector.

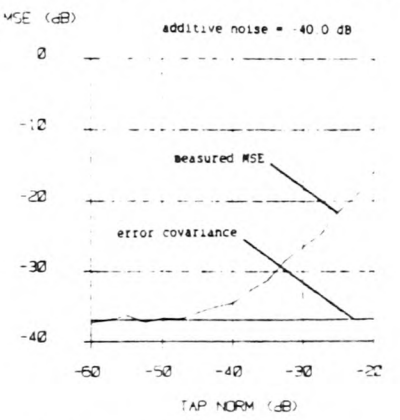


Fig.2: The Effect of Modelling Error on Kalman Equaliser Performance.

The definition of the error e(k+1), when rearranged yields.

$$x(k+1) = \underline{\hat{h}}^T(k) \, \underline{s}(k+1) + e(k+1) \qquad (3)$$

If, to a first order approximation, { e(k+1) } is considered to be a white noise process, then (3) provides an alternative interpretation of how the observation sequence { x(k+1) } was formed. The variance of the error e can be written down directly from [9].

$$E[e^2(k+1)] = \sigma_e^2 = \sigma_s^2 \, \varrho(k) + \sigma_n^2$$

In words, x(k+1) is formed by adding the output of a time varying FIR filter to a time varying white

noise sequence { e(k) }. The variance of the sequence { e(k) } combines both the effects of model uncertainty, represented by the norm , and the variance of the additive noise. The parameter e(k+1) is of course directly available as it is an output of the LMS system identification algorithm and thus its variance may be readily estimated on-line. A suitable recursive estimate is.

$$\hat{\sigma}^2_e(k+1) = \left[1 - \frac{1}{M} \right] \hat{\sigma}^2_e(k) + \frac{e^2(k+1)}{M} \qquad (4)$$

3.3 Simulation Results

In this section, the results of a simulation study of the adaptive Kalman equaliser are presented (Fig.3). For this simulation, the channel was modelled by (1), a zero mean binary white sequence of unit variance was used as the channel input, and a zero mean white Gaussian sequence was used for the additive noise. The channel had impulse response [0.26, 0.93, 0.26]. The trace labelled ' ρ ' is the measured value of the norm ρ(k). The traces labelled ' 0 ', and ' 5 ' are the measured MSE of the states 0, and 5 respectively of the Kalman filter. The traces labelled ' 0' ', and ' 5' ' are the elements on the leading diagonal of the error covariance matrix, being [0,0], and [5,5] respectively. The norm ρ(k) converges to the noise floor of -70dB at a rate that is consistent with (2). Like the channel the LMS algorithm has 3 taps. Thus the rate of convergence should be $\log_{10}(1.5)$ dB/iteration or -70dB in 40 iterations. The Kalman filter is initialised after 20 iterations of the LMS algorithm with a MSE of zero. After initialisation the Kalman filter diverges to MSE values that are determined by the level of additive noise and of the norm. It then reconverges due to the subsequent reduction in the norm until a final MSE value of -50dB for state 5 is achieved. Because the channel is non-minimum phase, the MSE of the states decreases with increasing state number and the total number of states is greater than the number of coefficients in the channel vector. There is a noticeable lag between the error covariance and the measured MSE of state 5, which is due to the inherent lag in the estimate of (4). To obtain the same MSE with a FIR equaliser, 10 taps are required. Using a RLS algorithm to train the FIR filter, it converges in approximately 30 iterations (Fig. 4), compared with 38 iterations required by the adaptive IIR filter.

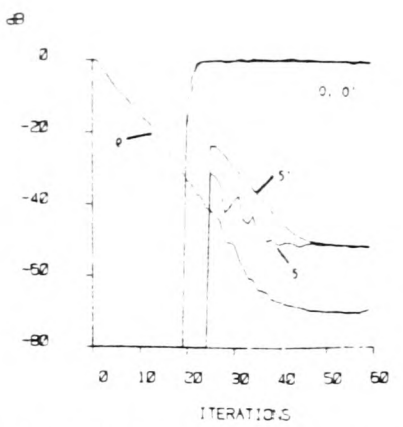


Fig 3: Convergence Performance of Adaptive Kalman Equaliser.

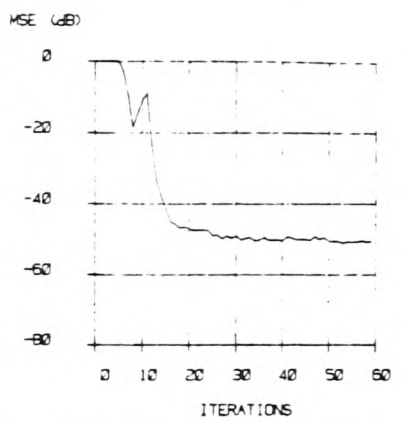


Fig.4: Convergence Performance of RLS Transversal Equaliser.

4. CONCLUSIONS

The adaptive IIR equaliser presented here converges to a given MSE performance goal in the same timescale as an RLS FIR equaliser. However the order of the IIR filter required to achieve that goal is always lower than the order of the FIR filter.

ACKNOWLEDGEMENT

This work was funded by the Science and Engineering Research Council of Great Britain.

REFERENCES

[1] J. G. Proakis, "Digital Communications," McGraw-Hill, 1983.
[2] B. D. Anderson, and J. B. Moore, "Optimal Filtering," Prentice-Hall, Englewood Cliffs, NJ, 1979.
[3] C. F. N. Cowan, and P. M. Grant, (eds.), "Adaptive Filters," Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
[4] R. E. Lawrence, and H. Kaufman, IEEE Trans. Comm. Tech., 19, no.6. pp. 1137-1141, Dec. 1971.
[5] L. Ljung, IEEE Trans. AC, 24, no.1, pp. 36-58, Feb. 1979.
[6] A. Luvison, and G. Pirani, IEEE Trans. AES, 15, no.5, pp. 635-648, Sept. 1979.
[7] S. B. Kleibanov, V. B. Privalskii, I. V. Time, Automation and Remote Control, vol .35, part 1, pp. 1097-1102, 1974.
[8] M. L. Honig, IEEE Trans. Comm., 33, no.1, pp. 65-73, Jan. 1984.
[9] G. Ungerboeck, IBM Journal Res. Develop., 16, no.6, pp. 546-555, Nov. 1972.

ON THE RECTANGULAR TRANSFORM APPROACH FOR BLMS ADAPTIVE FILTERING.

Ganapati Panda ,Bernard Mulgrew ,
Colin F.N.Cowan ,Member IEEE ,
Peter M.Grant ,Senior Member IEEE

University of Edinburgh,
Department of Electrical Engineering,
King's Buildings, Mayfield Road, Edinburgh,
EH9 3JL ,U.K.

ABSTRACT

This paper presents a new Block Least Mean Squares(BLMS) adaptive filter structure by exploiting the use of an efficient circular convolution algorithm known as the rectangular transform(RT). The BLMS algorithm is presented as an approximation to a recursive block least squares (RBLS) algorithm. The analysis is used to develop the computationally superior overlap save adaptive filter structure even though the structure of the overlap add adaptive filter may be easily worked out in a similar manner. The computational requirements of the proposed filter are investigated and compared with an FFT based filter. The proposed filter is found to be a useful alternative to its FFT counterpart. For completeness , the convergence properties of the filter obtained from simulations are also included.

I. INTRODUCTION

In recent years continuous effort has been expended on the design and implementation of frequency domain adaptive finite impulse response (FIR) digital filters [1-4]. These filters implement a block form of the least mean squares (LMS) algorithm which was proposed by Widrow [5]. The main motive behind the development of frequency domain adaptive filters is to exploit the efficiency of the fast Fourier Transform (FFT) as a circular convolution algorithm. An efficient circular convolution algorithm, known as the rectangular transform (RT) [6], has been successfully employed to develop adaptive filter that performs circular convolution [7-8]. In this paper, the RT is chosen as a basis for the development of a new BLMS linear adaptive filter structure. A detailed version of this paper has been submitted for publication [9]. In section II ,the rectangular transform circular convolution algorithm is briefly described. Modifications are made to the basic RT algorithms in order to facilitate efficient adaptive filter operation. In section III, two RT based linear convolution algorithms are described. The development of the BLMS as a simplification of the recursive block least squares (RBLS) is presented in section IV. In section V ,a detailed examination of the computational requirements of the overlap save structure is made. A direct comparison is made between an FFT based and RT based overlap save structure for a wide range of filter lengths.A brief examination of the convergence properties of the proposed algorithm is presented in section VI. Finally, in section VII ,the simulation results of the proposed RT BLMS adaptive filters are presented.

II. THE RECTANGULAR TRANSFORM

The N-point circular convolution $\{ y^c(i) \}$ of two finite sequences $\{ x(i) \}$ and $\{ h_j \}$ is defined as

$$y^c(i) = \sum_{j=0}^{N-1} h_j \cdot x([i-j] \bmod N) \qquad (1)$$

$i = 0, 1, \ldots\ldots, N-1; \; j = 0, 1, \ldots\ldots, N-1$

The rectangular transform is a dedicated convolution algorithm which efficiently computes (1) in the following steps [6].

$$\underline{X} = B\,\underline{x} \; ; \; \underline{H} = A\,\underline{h} \; ; \; \underline{Y} = \underline{X} \bullet \underline{H} \; ; \; \underline{y}^c = C\,\underline{Y}$$

where the column vectors $\underline{x}$ and $\underline{h}$ contain the elements of the input sequence $\{ x(i) \}$ and the impulse response sequence $\{ h_j \}$ respectively. The matrices A, B and C are real and rectangular and contain only simple elements. The symbol $\bullet$ denotes point-by-point multiplication of two vectors.The input, the output and the impulse response vectors are defined with their elements in time increasing order. The A, B and C matrices for small N values of 2, 3, 4, 5, 7, 8 and 9 are given in the literature [6]. To obtain convolution of long sequences, matrices of relatively prime Ns must be suitably nested . For a multifactor RT the A,B and C matrices have the form:

$$D = D_t \,\Delta\, D_{t-1} \,\Delta \ldots \Delta\, D_1$$

where Δ is the Kronecker product of matrices and t denotes the number of stages to be nested. The transform matrices being real, the convolution of two real sequences by this approach involve only real processing and this simplifies the overall filter structure. Besides ,this transform is more attractive than the FFT approach as it has been shown to be computationally more efficient up to a length of 420 [6]. In general, the existing A and B matrices are not equal. But in adaptive filtering ,computation with A and B matrices are required more than once. Therefore ,for each short

ICASSP 86, TOKYO CH2243-4/86/0000-2955 $1.00 © 1986 IEEE

transform, it is advantageous to make A = B by suitably modifying C.

III. LINEAR CONVOLUTION USING RT

The output y(k) of a finite impulse response (FIR) filter with impulse response vector $\underline{h}$ may be described by the vector inner product of two N-vectors.

$$y(k) = \underline{x}_R^T(k).\underline{h} \quad \text{where } \underline{h}^T = [\, h_0\ h_1\ \ldots\ h_{N-1}\,]$$

$$\text{and} \quad \underline{x}_R^T(k) = [\, x(k)\ x(k-1)\ \ldots\ x(k-N+1)\,]$$

For notational convenience the impulse response vector $\underline{h}$ is defined with its elements in time increasing order and the input vector $\underline{x}_R$ is defined with its elements in time decreasing order. To facilitate block processing it is also convenient to define an output N-vector $\underline{y}_R(k)$ as

$$\underline{y}_R(k) = [\, y(k)\ y(k-1)\ \ldots\ y(k-N+1)\,]^T = \underline{s}(k).\underline{h}$$

$$\text{where } \underline{s}^T(k) = [\, \underline{x}_R(k)\ \underline{x}_R(k-1)\ \ldots\ \underline{x}_R(k-N+1)\,]$$

The matrix $\underline{s}(k)$ is symmetric. It is well known that a circular convolution machine may be used to perform linear convolution if used in conjunction with an overlap save or overlap add data sectioning technique. The overlap save may be described succinctly in matrix notation.

Overlap save

$$\underline{y}(k) = [\, 0_N\ I_N\,] \begin{bmatrix} \underline{x}(k) \\ \underline{x}(k+N) \end{bmatrix} \bullet \begin{bmatrix} I_N \\ 0_N \end{bmatrix} \underline{h}$$

Using the RT-matrices, we have

$$\underline{y}(k) = [\, 0_N\ I_N\,]\, C_{2N}\, [\, \underline{X}(k+N) \bullet \underline{H}(k)\,]$$

where

$$\underline{X}(k+N) = B_{2N} \begin{bmatrix} \underline{x}(k) \\ \underline{x}(k+N) \end{bmatrix}, \text{and } \underline{H}(k) = A_{2N} \begin{bmatrix} I_N \\ 0_N \end{bmatrix} \underline{h}(k)$$

and $\underline{y}(k) = [\, y(k-N+1)\ \ldots\ y(k-1)\ y(k)\,]^T$

The symbol $\bullet$ represents circular convolution. The matrix I_N is an (NxN) identity matrix and the matrix 0_N is an (NxN) matrix with all zero elements. The overlap save technique produces an output vector y(k) which is the time reverse of the vector $\underline{y}_R(k)$.

IV. A BLOCK ADAPTIVE LINEAR CONVOLUTION ALGORITHM

The available literature on the BLMS algorithms is somewhat unclear on the subject of data ordering and thus does not lend itself to the formulation of RT based BLMS adaptive filter, since RT does not exhibit all the symmetry properties which are associated with the discrete Fourier transform(DFT).In this text ,a recursive block least squares (RBLS) approach to the formulation of an RT based BLMS adaptive filter which circumvents the above problem is presented. The formulation proceeds as follows. A linear estimate $\hat{y}$ of the process y is given by the output of an N-tap finite impulse response (FIR) filter.

$$\hat{y}(n) = \underline{x}_R^T(n).\underline{\hat{h}} \quad \text{where } \underline{\hat{h}}^T = [\, \hat{h}_0\ \hat{h}_1\ \ldots \hat{h}_{N-1}\,]$$

This estimate is considered to be least squares if the filter impulse response vector $\underline{\hat{h}}$ is chosen to minimise the sum of squared errors ,

$$\sum_{n=0}^{k} [\, y(n) - \hat{y}(n)\,]^2 \qquad (2)$$

given the two data sequences $\{x(n)\}$ and $\{y(n)\}$, where $n = 0,1,2,\ldots,k$. The value of $\underline{\hat{h}}$ which minimises the sum of the squared errors is given by the Wiener-Hopf equation.

$$\Gamma_{xx}(k).\underline{\hat{h}}(k) = \underline{\Gamma}_{xy}(k) \qquad (3)$$

$$\text{where} \quad \Gamma_{xx}(k) = \sum_{n=0}^{k} \underline{x}_R(n).\underline{x}_R^T(n)$$

$$\text{and} \quad \underline{\Gamma}_{xy}(k) = \sum_{n=0}^{k} \underline{x}_R(n).y(n)$$

One block of data later at sample (k+N)

$$\Gamma_{xx}(k+N).\underline{\hat{h}}(k+N) = \underline{\Gamma}_{xy}(k+N) \qquad (4)$$

Using (3) and (4) ,a block recursion for $\underline{\hat{h}}(k)$ may be obtained.

$$\underline{\hat{h}}(k+N) = \underline{\hat{h}}(k) + \Gamma_{xx}^{-1}(k+N).\underline{s}(k+N).\underline{e}_R(k+N) \qquad (5)$$

$$\text{where} \quad \underline{e}_R(k+N) = \underline{y}_R(k+N) - \underline{\hat{y}}_R(k+N)$$

$$\text{and} \quad \underline{\hat{y}}_R(k+N) = \underline{s}^T(k+N)\ .\ \underline{\hat{h}}(k) \qquad (6)$$

The vectors $\underline{y}_R(k+N)$ and $\underline{\hat{y}}_R(k+N)$ are defined with their elements in time decreasing order. Equations (5), and (6) do not completely describe a RBLS algorithm as a recursion for Γ_{xx}^{-1} has been neglected but they are sufficient to develop a BLMS algorithm. For a stationary data sequence $\{x(n)\}$,

$$\lim_{k\to\infty} \frac{1}{k}\, \Gamma_{xx}(k) = E[\, \underline{x}_R(k).\underline{x}_R^T(k)\,]$$

If the limiting operation is removed, the approximation results in a self orthogonalising algorithm [11] in block form. Further, if the sequence $\{x(n)\}$ is assumed to be white, the input autocorrelation matrix, $\underline{\Phi}_{xx}$,

$$\underline{\Phi}_{xx} = E[\, \underline{x}_R(k).\underline{x}_R^T(k)\,] = \sigma_x^2\, I_N$$

,where σ_x^2 is the power of the input signal x(k). The tap vector update equation for the BLMS algorithm [3,4] is then obtained as,

$$\underline{\hat{h}}(k+N) = \underline{\hat{h}}(k) + 2\mu\, \underline{s}(k+N).\underline{e}_R(k+N) \qquad (7)$$

The BLMS algorithm is summarised by (5),(6) and (7).Convergence in the mean of this algorithm is guaranteed provided the step size μ falls with in the limits

$$0 < \mu < \frac{1}{N\, \lambda_{max}}$$

where λ_{max} is the maxium eigenvalue of the input

signal autocorrelation matrix, $\underline{\Phi}_{YY}$. The algorithm contains two linear convolutions.

$$\underline{Y}_R(k+N) = \underline{s}(k+N).\underline{h}(k)$$

and

$$\underline{C}_R(k+N) = \underline{s}(k+N).\underline{e}_R(k+N)$$

There is no necessity to distinguish between convolution and correlation operations as in [3,4]. The ordering of the data is completely described by the definitions of the relevant vectors. The two linear convolutions may be generated by using an RT circular convolution machine in conjunction with an overlap add or overlap save data sectioning technique. The complete overlap save algorithm is summarised in Table 1. The development of an overlap add adaptive filter proceeds along similar lines. The (N x N) , time reversal matrix ,T_N, has 1's on the secondary diagonal and zero's elsewhere. It is used to convert vectors in natural order to vectors in time reversed order and vice versa.

V. COMPUTATIONAL REQUIREMENTS

The computational requirements of the linear adaptive filter proposed here are dealt with in this section. The total computational load on an adaptive filter employing a t-factors RT may be expressed as

overlap save

$$\text{Additions} = A = 2N + A_{N_1}N_2N_3\ldots N_t + M_{M_1}A_{N_2}N_3N_4\ldots N_t + M_{M_1}M_{M_2}\ldots M_{M_{t-1}}A_{N_t} \quad (8)$$

where $A_{N_i} = A_i + A_{B_i} + A_{C_i}$;i=1,2,....t.

$$\text{Multiplications} = M = 2(M_M + M_{C_1}N_2N_3\ldots N_t + M_{M_1}M_{C_2}N_3\ldots N_t + M_{M_1}M_{M_2}\ldots M_{M_{t-1}}M_{C_t}) \quad (9)$$

where $2N = N_1 \times N_2$ and $M_M = M_{M_1} \times M_{M_2}$.

M_{M_1} and M_{M_2} are the number of multiplications involved in the multiplication stage of N_1 and N_2 respectively. A_i, A_{B_i}, A_{C_i} and M_{C_i} represent the total number of additions, additions in the B stage, additions in the C stage and multiplications in the C stage respectively for an N_1 point short transform.

Optimum ordering of short transforms

The order of nesting of the short RT modules affects the number of operations. Therefore to obtain minimum computation proper ordering of the short modules is essential. Following the technique dealt in [6], we derive an optimum ordering of the modules which states that the transformation to be performed first is the one for which the quantity $(M_M - N)/(A+A_B + A_C)$ is smaller. Computation of this quantity suggests the order of nesting to be 2, 6, 4, 3, 8, 9, 5 and 7.

To compare the computational efficiency of the RT-based adaptive filters, the FFT-based linear adaptive filter proposed in the literatures [3,],[4] is considered. Further, the FFT employs radix-2 arithmetic and efficiently exploits the real input signal situation. The calculation also assumes that one complex multiplication is implemented through four real multiplications and two real additions. To obtain one block of N-point output, the FFT-based adaptive filter requires

$$M = 10N \log_2 N + 8N + 22 \quad (10)$$

$$A = 15N \log_2 N + 30N + 10 \quad (11)$$

The average number of operation counts per single output sample(iteration) for various order RT and FFT BLMS adaptive filters, computed from (8), (9), (10), and (11), are illustrated in Fig.1. This figure also includes the plots of the operation counts for the LMS algorithm [5] which requires 2N multiplications per iteration and 2N additions per iteration. The results can be summarised by the following remarks. The RT filter requires less multiplications than the FFT filter if the filter length is less than ~1000. Except for short filter length, i.e. N < 16, where the LMS algorithm is probably the best choice, the RT filter requires more additions than the FFT filter. By using a RT BLMS structure instead of a FFT BLMS structure the number of multiplications is reduced by increasing the number of additions. The RT BLMS adaptive filter structure is thus a useful alternative in applications where filter length of moderate order, e.g. 60, 126, 252, are required but multiplications are more expensive than additions.

VII. SIMULATION RESULTS

To justify the validity of the proposed adaptive filter structure ,the convergence rates for both white and correlated inputs were studied. To facilitate such a study the channel equalisation problem was chosen for simulation. The channel input was assumed to be a zero mean white binary sequence. The white sequence is convolved with a 3-tap symmetric channel having an eigenvalue ratio of 11.84 . If the impulse response of the channel is adjusted to an impulse, then the input to the equaliser filter becomes white. The equaliser under consideration is a 15-tap RT block LMS adaptive filter. The desired signal is obtained by delaying the channel input sequence by 8 samples. The simulation results for both input situations are illustrated in Fig.2. From this the characteristic dependency of a BLMS adaptive filter on input statistics can be clearly observed.

VIII. CONCLUSIONS

The use of rectangular transform (RT) has been demonstrated in the implementation of a transform domain linear adaptive filter structure. The adaptive filter update recursions were derived by using a simplification of a formulation based on a block recursive least squares algorithm to yield a block least mean squares filter structure. An evaluation of computational requirements shows this structure to be superior to equivalent structure realised using fast Fourier transform (FFT) algorithms.

ACKNOWLEDGEMENT

This work was funded by the Science and Engineering Research Council.

REFERENCES

[1] E.R. Ferrara, "Fast implementation of LMS adaptive filters," IEEE Trans. ASSP, vol.ASSP-28, pp.474-475,Aug.1980.
[2] D. Mansour and A.H. Gray, Jr, "Unconstrained frequency-domain adaptive filter," IEEE Trans. ASSP, vol.ASSP-30, pp.726-734, October 1982.
[3] G.A. Clark, S.K. Mitra, and S.R. Parker, "Block implementation of adaptive digital filters," IEEE Trans. CAS, vol.CAS-28, pp.584-592, June 1981, and vol.ASSP-29, pp.744-752, June 1981.
[4] G.A. Clark, S.R. Parker, and S.K. Mitra, "A unified approach to time- and frequency-domain realisation of FIR adaptive digital filters," IEEE Trans. ASSP, vol.ASSP-31, pp.1073-1083,Oct. 1983.
[5] B. Widrow et al, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," Proc. IEEE, vol.64, pp.1151-1162, August 1976.
[6] R.C. Agarwal and J.W. Cooley, "New algorithms for digital convolution," IEEE Trans. ASSP, vol.ASSP-25 ,pp.392-410,Oct.1977.
[7] G.Panda and P.M.Grant,"Rectangular-transformed-based adaptive filter,"Electronic Letters, vol.21, pp.301-303, March 1985.
[8] G. Panda et al, "A transform domain circular convolution algorithm for adaptive filtering,"submitted to IEEE Trans.on ASSP.
[9] G.Panda et al,"A rectangular transform structure for block least mean squares adaptive filtering,"submitted to IEEE Trans. on ASSP.
[10] C.F.N. Cowan and P.M. Grant, "Adaptive Filters," Prentice-Hall, New Jersey, pp.145-152, 1985.
[11] R.D.Gitlin and F.R.Magee,Jr,"Self-orthogonalising adaptive equalisation algorithms," IEEE Trans.Com., vol.COM-25, pp.666-672, July 1977.

Table 1 Overlap Save RT BLMS Adaptive Filter

$$\underline{X}(k+N) = B_{2N} \begin{bmatrix} \underline{x}(k) \\ \underline{x}(k+N) \end{bmatrix}$$

$$\dot{\underline{y}}_R(k+N) = \begin{bmatrix} O_N & T_N \end{bmatrix} C_{2N} \left[\underline{X}(k+N) \bullet \dot{\underline{H}}(k) \right]$$

$$\underline{e}_R(k+N) = \underline{y}_R(k+N) - \dot{\underline{y}}_R(k+N)$$

$$\underline{E}(k+N) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \underline{e}_R(k+N)$$

$$\underline{c}_R(k+N) = \begin{bmatrix} O_N & T_N \end{bmatrix} C_{2N} \left[\underline{X}(k+N) \bullet \underline{E}(k+N) \right]$$

$$\dot{\underline{h}}(k+N) = \dot{\underline{h}}(k) + 2\mu\, \underline{c}_R(k+N)$$

$$\dot{\underline{H}}(k+N) = A_{2N} \begin{bmatrix} I_N \\ O_N \end{bmatrix} \dot{\underline{h}}(k+N)$$

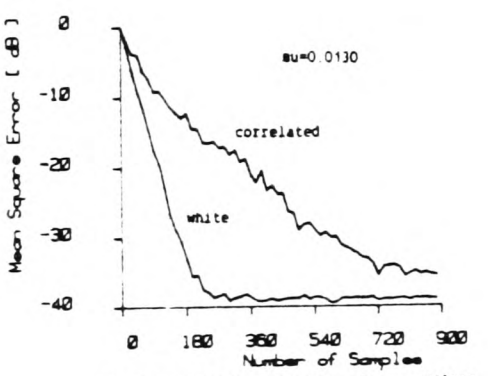


Fig 2 Convergence of RT BLMS Adaptive Filter

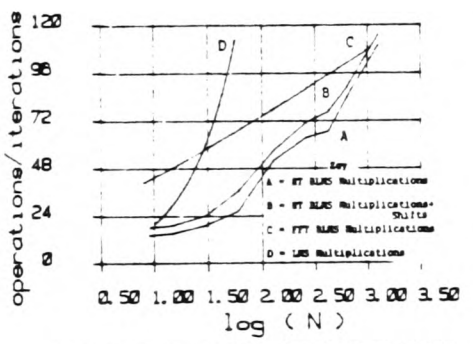


Fig 1(a) Comparison of Number of Multiplications

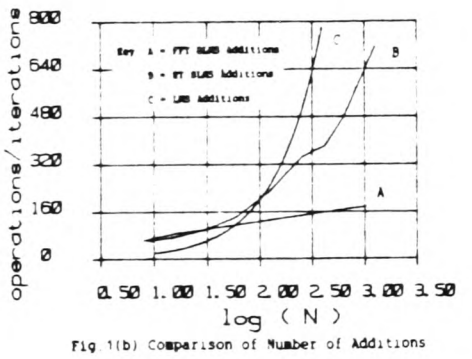


Fig 1(b) Comparison of Number of Additions

A Self-Orthogonalizing Efficient Block Adaptive Filter

GANAPATI PANDA, BERNARD MULGREW, COLIN F. N. COWAN, MEMBER, IEEE, AND PETER M. GRANT, SENIOR MEMBER, IEEE

Abstract—**This paper deals with the development of a unique self-orthogonalizing block adaptive filter (SOBAF) algorithm that yields efficient finite impulse response (FIR) adaptive filter structures. Computationally, the SOBAF is shown to be superior to the least mean squares (LMS) algorithm. The consistent convergence performance which it provides lies between that of the LMS and the recursive least squares (RLS) algorithm, but, unlike the LMS, is virtually independent of input statistics. The block nature of the SOBAF exploits the use of efficient circular convolution algorithms such as the FFT, the rectangular transform (RT), the Fermat number transform (FNT), and the fast polynomial transform (FPT). In performance, the SOBAF achieves the mean squared error (MSE) convergence of a self-orthogonalizing structure, that is, the adaptive filter converges under any input conditions, at the same rate as an LMS algorithm would under white input conditions. Furthermore, the selection of the step size for the SOBAF is straightforward as the range and the optimum value of the step size are independent of the input statistics.**

I. Introduction

In the area of adaptive filtering, the recursive least squares (RLS) [1] and the least mean squares (LMS) [2] are the two major alternatives in a tradeoff of convergence performance against computational complexity. The conventional RLS algorithm requires a number of computations per new data point that is a function of the square of the number of coefficients (N) in the finite impulse response (FIR) filter, i.e., order N^2 ($O(N^2)$). By exploiting the shift invariance properties [3], this has been reduced to $O(N)$. This and subsequent developments [4]–[7] make available the consistent rapid mean square error (MSE) convergence properties of the RLS algorithms at a computational cost, which is of the same order as the more commonly used LMS algorithm, whose convergence properties are generally poor [8]. However, the RLS algorithm still represents a computational load which is significantly higher than the LMS algorithm, typical figures being $10N$ multiplications per new data point for the RLS algorithm, compared to $2N$ for the LMS algorithm. The original aim of the work that is reported here was to find an algorithm that lay between the RLS and the LMS in both computational complexity and performance, and whose rate of convergence was independent of the input signal conditioning. In fact, the algorithm that has been developed goes beyond this initial goal in that it represents a significant reduction in computational load compared to an LMS algorithm for moderate to large values of N.

The first question to pose is what algorithms already exist which might provide a combination of computational complexity and performance which is between that of the LMS and RLS algorithms? A survey of the literature rapidly yields the term "self-orthogonalizing." The concept originated in [9]–[11] and was a result of the convergence analysis of the LMS algorithm and the recognition of the associated dependence of the rate of convergence of the LMS algorithm on the eigenvalues of the input autocorrelation matrix [2]. A self-orthogonalized algorithm involves constructing a linear operator (transform or preprocessor) which maps the input N-vector $\boldsymbol{x}$ to an N-vector $\boldsymbol{u}$ such that the elements of $\boldsymbol{u}$ are mutually orthogonal. Given this, the matrix $E[\boldsymbol{u}\boldsymbol{u}^T]$ is a diagonal whose eigenvalue spread can be normalized to unity by dividing each element of $\boldsymbol{u}$ by the square root of the appropriate eigenvalue. The resultant N-vector z is white with unit variance, i.e., $E[zz^T]$ is an $N \times N$ identity matrix. If the vector z forms the input to an LMS algorithm, it is straightforward to predict, using the convergence analyses of [8] and [12], that the complete structure (linear operator + eigenvalue normalization + LMS) will converge (in a mean square error (MSE) sense) under any input conditions at the same rate as an LMS algorithm would under white input conditions. This technique is equivalent to multiplying the gradient term in an LMS algorithm by the inverse of the input autocorrelation matrix $E[\boldsymbol{x}\boldsymbol{x}^T]$ [11], [13].

Only in a limited number of applications such as [14] is the input autocorrelation matrix, or equivalently an orthogonalizing operator, known *a priori*, and hence, for general purpose applications two suboptimum techniques have been suggested. In the first, a fixed linear operator such as a discrete Fourier transform (DFT) or discrete cosine transform (DCT) is chosen that performs an approximate orthogonalization of the input vector [13], [15]. The subsequent processing proceeds as if the orthogonaliza-

Manuscript received May 14, 1986. This work was supported by the Science and Engineering Research Council of Great Britain.

G. Panda was with the Department of Electrical Engineering, University of Edinburgh, Edinburgh EH9 3JL, Scotland. He is now with Sambalpur University, Sambalpur, India.

B. Mulgrew, C.F.N. Cowan, and P. M. Grant are with the Department of Electrical Engineering, University of Edinburgh, Edinburgh EH9 3JL, Scotland.

IEEE Log Number 8610482.

tion was exact. In the second, the autocorrelation matrix is estimated directly from the data, inverted, and used to multiply the gradient estimate [11]. These two techniques might be classified as explicit and implicit orthogonalization, respectively. However, it should be noted that the form of estimate for the input autocorrelation matrix identified in [11] as the ideal self-orthogonalizing algorithm gives rise to the RLS algorithm. It is clear from [16] that even under white input conditions, the RLS algorithm may outperform the LMS algorithm. Consequently, it must be concluded that an RLS algorithm does more than merely orthogonalize the input signal, and therefore it should not be classified as self-orthogonalizing algorithm.

Explicit orthogonalization techniques have been applied to fast Fourier transform (FFT) [17] based block adaptive filters [18], [19]. The FFT is used in the block LMS (BLMS) algorithms of [18], [20]–[22], and [42] to provide fast convolution and fast estimation of the gradient. As a by-product of this implementation, the FFT of an augmented input vector is available, i.e., a fixed transform that performs approximate orthogonalization. Bartlett spectral estimation has also been considered in an attempt to improve the quality of this approximation [19]. If, however, other fast convolution algorithms such as the Fermat number transform (FNT) [23] or the rectangular transform (RT) [24] are to be applied to a self-orthogonalized block adaptive filter as they have been applied to the BLMS algorithm [25], [26], then since they cannot be assumed to exhibit even approximate orthogonalizing properties, an implicit approach must be considered.

The self-orthogonalizing block adaptive filter (SOBAF) that is presented here is a unique alternative to the RLS and LMS algorithms. It provides a combination of computational load, which is significantly less than the LMS algorithm, and consistent convergence performance, which lies between that of the LMS and RLS algorithms but, unlike the LMS, is virtually independent of the input statistics. Therefore, it is well suited to applications where neither the LMS nor the RLS algorithm can provide the correct tradeoff of computational load against convergence performance. The computational efficiency is achieved by using a block filtering structure which is similar to the BLMS algorithm [20], [21] and, hence, may exploit either FFT [17] or RT [24]. The rapid convergence performance is achieved by using an implicit self-orthogonalizing technique, which ensures that the algorithm will converge under any input conditions at the same rate as an LMS algorithm would under white input conditions.

The paper is subdivided in the following manner. In Section II the theoretical development of the algorithm is presented and the results verified by computer simulation. Section III contains the arguments that lead to a practical SOBAF algorithm, along with details of how it can be implemented efficiently. In Section IV the computational load of the proposed filter is assessed. Finally, in Section V, results from computer simulations are presented which confirm that the practical SOBAF achieves the convergence performance that was promised in the theoretical considerations of Section II.

II. Theoretical Considerations

A. Theory

Consider a stationary sequence of N-vectors $\{\boldsymbol{x}(i)\}$, which is zero mean uncorrelated in time and jointly Gaussian. The sequence is completely described by the $N \times N$) autocorrelation matrix $\boldsymbol{\varphi}_{xx}$, where

$$\boldsymbol{\varphi}_{xx} = E[\boldsymbol{x}(i)\, \boldsymbol{x}^T(i)].$$

The superscript T denotes transpose. Although the matrix is positive semidefinite, in many applications it can be assumed to be positive definite, in which case there exists an $(N \times N)$ matrix Q such that

$$Q^T \boldsymbol{\varphi}_{xx} Q = I_N$$

where I_N is an $(N \times N)$ identity matrix. The matrix Q is not unique [27]. A second sequence of N-vectors, $\{\boldsymbol{z}(i)\}$, which is uncorrelated in time and zero mean, may be generated from $\{\boldsymbol{x}(i)\}$ using the matrix Q^T.

$$\boldsymbol{z}(i) = Q^T \boldsymbol{x}(i), \tag{2.1}$$

therefore, Q^T may be considered to be a whitening filter, since

$$\boldsymbol{\varphi}_{zz} = E[\boldsymbol{z}(i)\, \boldsymbol{z}^T(i)] = I_N.$$

Because of the special structure of $\boldsymbol{\varphi}_{zz}$, i.e., it is diagonal with equal eigenvalues, a stochastic gradient search adaptive filter with input $\{\boldsymbol{z}(i)\}$ will achieve rapid consistent convergence rates. Furthermore, since $\{\boldsymbol{z}(i)\}$ is also uncorrelated in time, zero mean, and jointly Gaussian [1], the theoretical results of [12] and [28] may be applied directly to give 1) bounds on the step size μ that ensure convergence in a MSE sense, and 2) a single optimum value μ_{opt} for fastest convergence.

To the sequence of N-vectors $\{\boldsymbol{z}(i)\}$, apply a BLMS algorithm [21] of block length L to form an estimate $\{\hat{y}(i)\}$ of the stationary scalar sequence $\{y(i)\}$. The aim is to minimize the MSE

$$E[(y(i) - \hat{y}(i))^2].$$

The estimate $\hat{y}(i)$ is linear and is formed using the weight N-vector $\boldsymbol{w}$, i.e.,

$$\hat{y}(i) = \boldsymbol{z}^T(i)\boldsymbol{w}. \tag{2.2}$$

The optimum solution which minimizes the MSE is given by

$$\boldsymbol{w}_{opt} = \boldsymbol{\varphi}_{zz}^{-1}\, E[\boldsymbol{z}(i)\, y(i)]$$
$$= E[\boldsymbol{z}(i)\, y(i)].$$

The BLMS algorithm is defined by the following three equations [21].

$$\hat{\boldsymbol{y}}(j) = \boldsymbol{\zeta}(j)\, \boldsymbol{w}(j-1) \tag{2.3}$$

$$\boldsymbol{e}(j) = \boldsymbol{y}(j) - \hat{\boldsymbol{y}}(j) \tag{2.4}$$

$$\boldsymbol{w}(j) = \boldsymbol{w}(j-1) + \frac{2\mu}{L}\zeta^T(j)\boldsymbol{e}(j) \tag{2.5}$$

and

$$\boldsymbol{y}(j) = [y(jL)\ y(jL-1)\ \cdots\ y(jL-L+1)]^T$$
$$\hat{\boldsymbol{y}}(j) = [\hat{y}(jL)\ \hat{y}(jL-1)\ \cdots\ \hat{y}(jL-L+1)]^T$$

and

$$\zeta^T(j) = [z(jL)\ z(jL-1)\ \cdots\ z(jL-L+1)].$$

The index j is known as the block number, where a block contains L data vectors. The weight vector $\boldsymbol{w}$ is only updated once per block.

Using the theoretical results of [28], a recursive relationship for the block MSE σ_ϵ^2 is obtained.

$$\sigma_\epsilon^2(j) = \left(1 - 4\mu + \frac{4\mu^2}{L}(L+N+1)\right)\sigma_\epsilon^2(j-1) + \left(4\mu - \frac{4\mu^2}{L}(L+1)\right)\sigma_{\text{opt}}^2 \tag{2.6}$$

where

$$\sigma_\epsilon^2(j) = \frac{1}{L}E[\boldsymbol{e}^T(j)\ \boldsymbol{e}(j)]$$

and σ_{opt}^2 is the minimum MSE that is obtained when the weight vector $\boldsymbol{w}_{\text{opt}}$ is used.

$$\sigma_{\text{opt}}^2 = E[(y(i) - \boldsymbol{w}_{\text{opt}}^T\boldsymbol{z}(i))^2].$$

Equation (2.6) yields bounds that ensure MSE convergence,

$$0 \le \mu < \frac{L}{(L+N+1)}. \tag{2.7}$$

and a value μ_{opt} which gives fastest convergence [28].

$$\mu_{\text{opt}} = \frac{L}{2(L+N+1)}. \tag{2.8}$$

Equations (2.1), (2.3), (2.4), and (2.5) thus define a self-orthogonalizing block adaptive filter whose MSE convergence is ensured, provided μ is chosen within the limits of (2.7), and whose rate of convergence is independent of the eigenvalues of the autocorrelation matrix $\boldsymbol{\varphi}_{zz}$, (2.6).

This self-orthogonalizing block adaptive filter may be reformulated in terms of an overall weight vector $\boldsymbol{h}$ where

$$\hat{y}(i) = \boldsymbol{x}^T\boldsymbol{h} \tag{2.9}$$

such that explicit knowledge of the matrix Q is unnecessary. Combining (2.9) and (2.1),

$$\hat{y}(i) = \boldsymbol{x}^T(i)Q\boldsymbol{w}$$

from which a relationship between $\boldsymbol{h}$ and $\boldsymbol{w}$ is obtained.

$$\boldsymbol{h} = Q\boldsymbol{w}. \tag{2.10}$$

Application of (2.1) and (2.10) to (2.3) and (2.5) yields

$$\hat{\boldsymbol{y}}(j) = \boldsymbol{\chi}(j)\boldsymbol{h}(j-1) \tag{2.11}$$

and

$$\boldsymbol{h}(j) = \boldsymbol{h}(j-1) + \frac{2\mu}{L}\boldsymbol{\varphi}_{zz}^{-1}\boldsymbol{\chi}^T(j)\boldsymbol{e}(j) \tag{2.12}$$

where

$$\boldsymbol{\chi}^T(j) = [\boldsymbol{x}(jL)\boldsymbol{x}(jL-1)\ \cdots\ \boldsymbol{x}(jL-L+1)].$$

B. Comparison with Simulation

The theoretical results presented above rely on the assumption that the sequence $\{\boldsymbol{x}(i)\}$ is uncorrelated in time and jointly Gaussian. For an adaptive transversal filter application, the sequence $\{\boldsymbol{x}(i)\}$ is never uncorrelated in time, since

$$\boldsymbol{x}(i) = [x(i)\ x(i-1)\ \cdots\ x(i-N+1)]^T$$

and

$$\boldsymbol{x}(i-1) = [x(i-1)\ x(i-2)\ \cdots\ x(i-N)]^T$$

and is rarely exactly Gaussian. The aim of this section is to test the validity of the theoretical convergence results, summarized in (2.6), for the particular case of an adaptive transversal communcations channel equalizer where neither assumption is true.

A typical equalizer scenario is illustrated in Fig. 1. The digital message is a zero-mean binary distributed white random sequence $\{u(i)\}$. The channel is modeled by an FIR filter whose output is corrupted by a zero mean white Gaussian sequence $\{n(i)\}$. The role of the adaptive filter is to form a fixed lag estimate of the channel input. The training signal for the adaptive filter is thus

$$y(i) = u(i-d)$$

where d is a positive integer. For the purposes of the simulations presented here, a three-tap channel was used (channel 2, Table I). The signal-noise ratio, defined as $E[u^2]/E[n^2]$, was set at 40 dB. The self-orthogonalizing adaptive algorithm, defined by (2.4), (2.7), (2.11), and (2.12), was used to update a 15-tap transversal equalizer. The block length L was set at 15. Under these conditions the autocorrelation matrix, $\boldsymbol{\varphi}_{zz}$, has a maximum/minimum eigenvalue ratio of approximately 11. The convergence performance of the algorithm is illustrated in Fig. 2, on which is shown both a measured MSE calculated from an ensemble of 20 runs and a theoretical MSE calculated from (2.6). This clearly emphasizes the close agreement between theory and practice.

III. A Practical Algorithm

In this section a new block adaptive filter algorithm is described. This algorithm is a unique combination of three concepts. The first involves the performance goal, which is chosen to be the MSE convergence of a self-orthogonalized stochastic gradient search algorithm, summarized in (2.6). In words, the adaptive filter should converge, in an MSE sense, under any input conditions, as the same rate as it would if the input sequence was white. Thus, the self-orthogonalized algorithm will not by definition

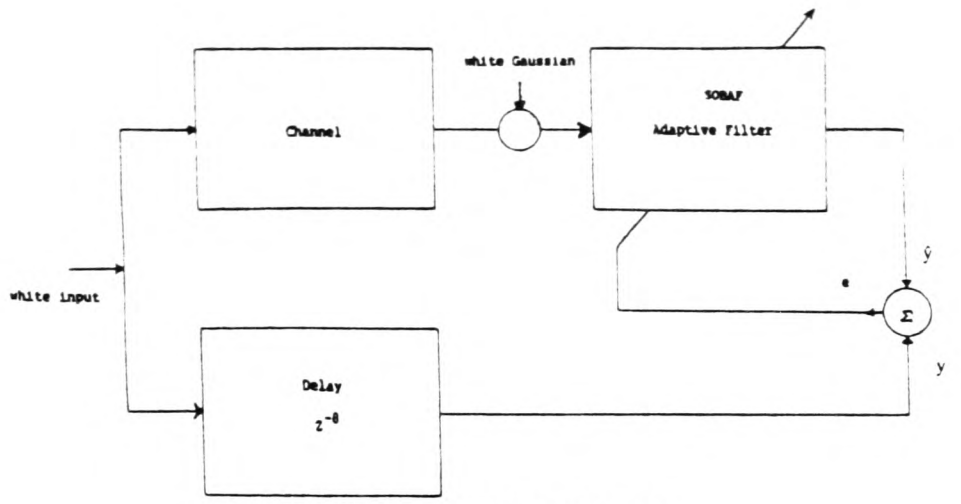


Fig. 1. A typical equalizer block diagram.

TABLE I
CHANNEL IMPULSE RESPONSES

Channel Number	Impulse Response	Eigenvalue Ratio (15-tap filter)
1	$1.0 + 0.0z^{-1} + 0.0z^{-2}$	1.0
2	$0.2602 + 0.9298z^{-1} + 0.2602z^{-2}$	11.8
3	$0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$	68.6

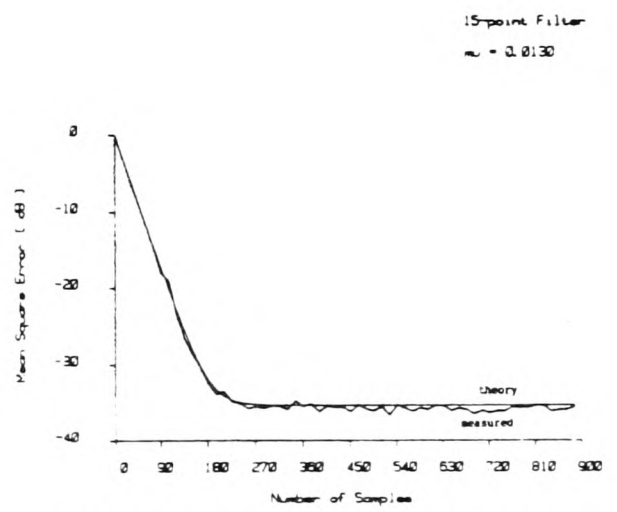


Fig. 2. Comparison between theoretical and measured MSE.

exhibit a sensitivity to the eigenvalue spread of the autocorrelation matrix $\boldsymbol{\varphi}_{xx}$ that is a characteristic of both LMS [8], [12] and BLMS [28] algorithms. However, it should also be noted that this performance goal is not equivalent to the performance of a RLS algorithm, since even under white input conditions the RLS algorithm will outperform a stochastic gradient search algorithm [16].

The second concept involves the choice of an estimator for the autorcorrelation matrix. In a general adaptive filter application, the autocorrelation matrix is unknown, and hence, the tap weight update, (2.12), must be replaced by

$$\boldsymbol{h}(j) = \boldsymbol{h}(j-1) + \frac{2\mu}{L}\hat{\boldsymbol{\Phi}}_{xx}^{-1}(j)\,\boldsymbol{\chi}^T(j)\boldsymbol{e}(j) \qquad (3.1)$$

where $\hat{\boldsymbol{\Phi}}_{xx}(j)$ is an estimate of $\boldsymbol{\varphi}_{xx}$ at block j. Several possible estimates of $\boldsymbol{\varphi}_{xx}$ exist in the literature. The most notable is

$$\hat{\boldsymbol{\Phi}}_{xx}(j) = \frac{1}{jL}\sum_{i=1}^{jL}\boldsymbol{x}(i)\boldsymbol{x}^T(i). \qquad (3.2)$$

However, the use of this estimate would produce an RLS block adaptive filter structure [26]. Hence, it is considered inappropriate here, as its convergence performance would not be that of a self-orthogonalized stochastic gradient filter. Furthermore, computationally efficient RLS block adaptive filter algorithms already exist [29]. In the estimation technique that is considered here, the matrix $\hat{\boldsymbol{\Phi}}_{xx}(j)$ is assumed to be symmetric Toeplitz. Thus, each element may be generated from knowledge of the first column $\hat{\boldsymbol{p}}(j)$, where

$$\hat{\boldsymbol{p}}(j) = [\hat{p}_0(jL)\hat{p}_1(jL)\cdots\hat{p}_{N-1}(jL)]^T$$

and

$$\hat{p}_k(jL) = \frac{1}{jL}\sum_{i=1}^{jL}x(i)\,x(i-k) \qquad 0 \le k < N.$$

The vector $\hat{\boldsymbol{p}}(j)$ may thus be updated on a block-by-block basis.

$$\hat{\boldsymbol{p}}'(j) = \hat{\boldsymbol{p}}'(j-1) + \frac{1}{L}\delta\boldsymbol{p}(j) \qquad (3.3)$$

$$\hat{\boldsymbol{p}}(j) = \frac{1}{j}\hat{\boldsymbol{p}}'(j) \qquad (3.4)$$

where

$$\delta\boldsymbol{p}(j) = \boldsymbol{\chi}^T(j)\,\boldsymbol{x}_L(j) \qquad (3.5)$$

and

$$\boldsymbol{x}_L = [x(jL)\ x(jL-1) \cdots x(jL-L+1)]^T.$$

The Toeplitz assumption also allows the application of computationally efficient techniques such as the Levinson recursion in [30] and more recently [31] for the solution of

$$\hat{\boldsymbol{\Phi}}_{xx}(j)\,\delta\boldsymbol{h}(j) = \boldsymbol{c}(j) \tag{3.6}$$

where

$$\boldsymbol{c}(j) = \boldsymbol{\chi}^T(j)\,\boldsymbol{e}(j). \tag{3.7}$$

It is well known in equalizer [32] and spectral estimation [33] applications that the Toeplitz assumption produces poorer performance than the estimate of (3.2). Hence, the choice of a Toeplitz assumption here may be interpreted as reflecting a desire to degrade the performance of the algorithm from that of an RLS structure to that of a self-orthogonalized structure.

The third concept involves the application of computationally efficient circular convolution algorithms, of which the fast Fourier transform (FFT) [17] and the rectangular transform (RT) [24] are but two examples, to produce an adaptive filter algorithm which is itself computationally efficient. This technique has been the motivation behind the development of the BLMS algorithm, which is computationally superior to the LMS algorithm [20], [21], [25], [26]. To utilize this technique, the linear convolution operations are first identified. In this case they are (2.11), (3.5), and (3.7). Of these three, (2.11) and (3.7) are common to both the BLMS and the self-orthogonalized structures. The existence of (3.5) is a direct result of the Toeplitz assumption on $\hat{\boldsymbol{\Phi}}_{xx}$, and is a significant factor in making that assumption. Each of these linear convolution operations is then performed using a combination of either overlap-add or overlap-save data sectioning [34] and a circular convolution algorithm. To simplify the notation, only overlap-save and a block length $L = N + 1$ will be considered here, as these are known to produce the most efficient adaptive filter structures [21]. This does not, however, detract from the generality of the results.

Let the circular convolution of two N-vectors $\boldsymbol{\alpha}(j)$ and $\boldsymbol{\beta}(j)$ be defined by the N-vector $\boldsymbol{\gamma}(j)$, where $\boldsymbol{\alpha}(j)$ and $\boldsymbol{\beta}(j)$ contain the last N samples from the scalar sequences $\{\alpha(i)\}$ and $\{\beta(i)\}$, respectively, in time-increasing order; thus,

$$\boldsymbol{\gamma}(j) = \boldsymbol{\alpha}(j) * \boldsymbol{\beta}(j)$$

where

$$\boldsymbol{\alpha}(j) = [\alpha(jN-N+1) \cdots \alpha(jN-1)\ \alpha(jN)]^T$$

$$\boldsymbol{\beta}(j) = [\beta(jN-N+1) \cdots \beta(jN-1)\ \beta(jN)]^T.$$

The symbol $*$ denotes circular convolution. This operation may be performed using a transform based processor, which is defined by the two $(M \times N)$ matrices $\boldsymbol{A}_N$ and $\boldsymbol{B}_N$ and the $(N \times M)$ matrix $\boldsymbol{C}_N$.

$$\boldsymbol{\gamma}(j) = \boldsymbol{C}_N\,(\boldsymbol{A}_N\boldsymbol{\alpha}(j) \otimes \boldsymbol{B}_N\boldsymbol{\beta}(j)).$$

TABLE II
SELF-ORTHOGONALIZING BLOCK ADAPTIVE FILTER

$$\boldsymbol{X}(j) = \boldsymbol{B}_{2N}\begin{bmatrix} T_N\boldsymbol{x}(j-1) \\ T_N\boldsymbol{x}(j) \end{bmatrix}$$

$$\boldsymbol{H}(j-1) = \boldsymbol{A}_{2N}\begin{bmatrix} I_N \\ O_N \end{bmatrix}\boldsymbol{h}(j-1)$$

$$\hat{\boldsymbol{y}}(j) = [O_N T_N]\,\boldsymbol{C}_{2N}\{\boldsymbol{X}(j) \otimes \boldsymbol{H}(j-1)\}$$

$$\boldsymbol{e}(j) = \boldsymbol{y}(j) - \hat{\boldsymbol{y}}(j)$$

$$\boldsymbol{E}(j) = \boldsymbol{A}_{2N}\begin{bmatrix} I_N \\ O_N \end{bmatrix}\boldsymbol{e}(j)$$

$$\boldsymbol{c}(j) = [O_N T_N]\,\boldsymbol{C}_{2N}\{\boldsymbol{X}(j) \otimes \boldsymbol{E}(j)\}$$

$$\boldsymbol{X}'(j) = \boldsymbol{A}_{2N}\begin{bmatrix} I_N \\ O_N \end{bmatrix}\boldsymbol{x}(j)$$

$$\delta\boldsymbol{p}(j) = [O_N T_N]\,\boldsymbol{C}_{2N}\{\boldsymbol{X}(j) \otimes \boldsymbol{X}'(j)\}$$

$$\boldsymbol{p}'(j) = \boldsymbol{p}'(j-1) + \frac{1}{N}\delta\boldsymbol{p}(j)$$

$$\boldsymbol{p}(j) = \frac{1}{j}\boldsymbol{p}'(j)$$

$$\delta\boldsymbol{h}(j) = \hat{\boldsymbol{\Phi}}_{xx}^{-1}(j)\,\boldsymbol{c}(j)$$

$$\boldsymbol{h}(j) = \boldsymbol{h}(j-1) + \frac{2\mu}{N}\delta\boldsymbol{h}(j)$$

The symbol $\otimes$ denotes the point-by-point multiplication of the M-vectors $\boldsymbol{A}_N\boldsymbol{\alpha}(j)$ and $\boldsymbol{B}_N\boldsymbol{\beta}(j)$, $M \geq N$. This is a generalized transform based circular convolution processor. For example, an FFT structure would be obtained if $M = N$, $\boldsymbol{A}_N = \boldsymbol{B}_N = \boldsymbol{F}_N$, and $\boldsymbol{C}_N = \boldsymbol{F}_N^{-1}/N$, where $\boldsymbol{F}_N$ is the $(N \times N)$ discrete Fourier transform matrix [35].

The linear convolutions of (2.11), (3.5), and (3.7) are obtained in the following way [26]. Taking (2.11) as an example,

$$\begin{aligned}\hat{\boldsymbol{y}}(j) &= \boldsymbol{\chi}(j)\boldsymbol{h}(j-1) \\ &= [O_N T_N]\left\{\begin{bmatrix} T_N\boldsymbol{x}(j-1) \\ T_N\boldsymbol{x}(j) \end{bmatrix} * \begin{bmatrix} I_N \\ O_N \end{bmatrix}\boldsymbol{h}(j-1)\right\} \\ &= [O_N T_N]\boldsymbol{C}_{2N}\{\boldsymbol{X}(j) \otimes \boldsymbol{H}(j-1)\}\end{aligned}$$

where

$$\boldsymbol{X}(j) = \boldsymbol{B}_{2N}\begin{bmatrix} T_N\boldsymbol{x}(j-1) \\ T_N\boldsymbol{x}(j) \end{bmatrix}$$

and

$$\boldsymbol{H}(j-1) = \boldsymbol{A}_{2N}\begin{bmatrix} I_N \\ O_N \end{bmatrix}\boldsymbol{h}(j-1).$$

The two $(M \times 2N)$ matrices $\boldsymbol{A}_{2N}$ and $\boldsymbol{B}_{2N}$ and the $(2N \times M)$ matrix $\boldsymbol{C}_{2N}$ define a circular convolution machine which operates on $2N$-vectors, $M \geq 2N$. The matrix I_N is an $(N \times N)$ identity matrix, the matrix O_N is $(N \times N)$ with all zero elements, and the $(N \times N)$ time reversal matrix T_N has 1's on the secondary diagonal and zeros elsewhere. The complete algorithm is summarized in Table II and illustrated in Fig. 3.

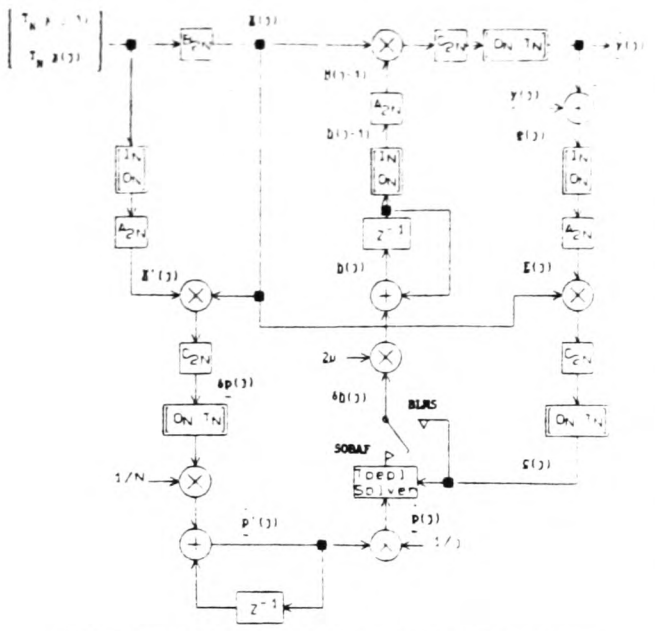


Fig. 3. A generalized overlap-save SOBAF adaptive filter.

IV. Computational Complexity

As mentioned earlier, efficient structures for the proposed adaptive filters are obtained when block convolution algorithms such as the RT and the FFT are employed in conjunction with a 50 percent overlap-save technique. These efficient structures are considered here for the evaluation of computational loads. In case of the RT-based filter, the algorithms in which $A = B$ are chosen to find the overall number of operations. To facilitate the evaluation procedure, the adaptive filter structure is divided into three main parts, namely, the BLMS, the autocorrelation estimator, and Levinson's recursion [30]. The computation of the number of operations assumes the filter input to be real. The filter under consideration has N-taps, and hence, the matrices of the algorithms correspond to $2N$-points. For an RT of transform length $2N$, let there be t numbers of nested RT modules, N_i, related by

$$2N = \prod_{i=1}^{t} N_i.$$

Each short module N_i has the following parameters:

- A_{B_i} number of additions in the B-stage
- A_{C_i} number of additions in the C-stage
- M_{M_i} number of multiplications in the multiplication (M)-stage
- M_{C_i} number of multiplications in the C-stage.

The total number of multiplications encountered in the M-stage of a $2N$-point RT is given by

$$M_M = \prod_{i=1}^{t} M_{M_i}.$$

Using this notation, the number of operations in each part of the RT-based adaptive filter is obtained as follows.

For processing one block of N-point data, the BLMS part requires computation of three B and two C matrices. Besides, it includes $2M_M$ multiplications for weighting and updating, and $2N$ additions for weighting and comparison. The autocorrelation estimator involves computation of one C and two B matrices. It also requires $(M_M + 1)$ multiplications and N additions for averaging and weighting. It is worth mentioning here that the computation of one B matrix is common to both the parts. For processing two N-point vectors, the Levinson recursion requires $2N^2 - 3N$ additions, $2N^2 - N - 3$ multiplications, and $N - 1$ divisions. Combining all these, the overall computational effort required for processing each sample of an iteration of an N-point RT filter may be expressed as

$$A = 2N\left[N + \sum_{i=1}^{t} \frac{A_i}{N_i}\left\{\prod_{k=0}^{i-1} \frac{M_k}{N_k}\right\}\right] \tag{4.1}$$

$$M = 2N - 1 + \frac{3M_M - 2}{N} + 6\left[\sum_{i=1}^{t} \frac{M_{C_i}}{N_i}\left\{\prod_{k=0}^{i-1} \frac{M_k}{N_k}\right\}\right] \tag{4.2}$$

$$D = 1 - \frac{1}{N} \tag{4.3}$$

where

$$A_i = 4A_{B_i} + 3A_{C_i}.$$

The symbols A, M, and D denote the total number of additions, multiplications, and divisions, respectively. In (4.1) and (4.2), the initial value M_0/N_0 is assumed to be unity. As pointed out in [26], the total number of operations depend on the ordering of the nested modules. Therefore, there exists an optimum order which yields a minimum overall operations. Based on their results, the order of nesting of the short RT modules is given as 2, 6, 4, 3, 8, 9, 5, 7.

In an identical manner, the computational requirements of the FFT-based orthogonalized BLMS adaptive filter may be obtained. The FFT algorithm chosen here is based on the radix-2 formulation and efficiently employs real inputs [35]. It is also assumed that a complex multiplication is implemented through four real multiplications and two real additions. Considering all these, the processing load on each sample of an iteration of the N-point FFT based adaptive filter is computed as

$$A = 2N + 21\log_2 N + 41 + \frac{14}{N} \tag{4.4}$$

$$M = 2N + 14\log_2 N + 11 + \frac{28}{N} \tag{4.5}$$

$$D = 1 - \frac{1}{N} \tag{4.6}$$

where the symbols are defined earlier.

There are a variety of algorithms [30], [36]–[38] which may be employed to solve the Toeplitz system of equations. In computing the number of operations in the filter structure, we have used only the well-known Levinson recursion [30]. But significant further computational savings can be achieved by exploiting the use of recently reported fast algorithms for solving the Toeplitz system of equations [31], [39], [40]. The technique dealt with in [31] is of particular interest to us, since it uses the FFT algorithm to perform block convolution. This entire algorithm of [31] requires

$$2.5N \log_2 N \log_2 N + 11.5N \log_2 N + 6N$$

multiplications and the same number of additions to solve a Toeplitz system of N equations. It may be pointed out here that the other block convolution algorithms such as the RT [24], FNT [23], and the fast polynomial transform (FPT) [41] may be efficiently used for the same purpose. As an illustration, let us apply this technique to the FFT-based filter and compute the number of operations. The average number of operations per single output sample of the FFT based filter is found to be

$$A = \log_2 N[2.5 \log_2 N + 25.5] + 18 + \frac{31}{N} \quad (4.7)$$

$$M = \log_2 N[2.5 \log_2 N + 32.5] + 47 + \frac{14}{N}. \quad (4.8)$$

In assessing the computational load of the SOBAF, it is easier at first to consider an FFT-based formulation, since this yields simple closed-form expressions; see (4.4) and (4.5). The computational load of an LMS algorithm is $O(N)$, i.e., the number of operations increases linearly with the number of taps in the transversal filter. An FFT-based BLMS algorithm, on the other hand, is $O(\log N)$. Thus, an FFT-based BLMS algorithm has a significant advantage in computational efficiency over an LMS algorithm for moderate to large N. As mentioned already, the SOBAF requires the solution of a Toeplitz set of equations. Using the Levinson recursion this requires $O(N^2)$ operations, which reduces to $O(N)$ since the equations are only solved once per block. Although the remaining operations in (4.4) and (4.5) are at most $O(\log N)$, the linear term will dominate, and hence, the overall computational load is $O(N)$, which is the same as an LMS algorithm. A more detailed comparison is illustrated in Figs. 4 and 5 for the LMS algorithm and RT-based BLMS and SOBAF algorithms.

If, however, the fast inversion technique of [31] is applied to the solution of the Toeplitz equations, then the computational load of the SOBAF is $O(\log N)$; see (4.7) and (4.8). In Figs. 6 and 7 the computational requirements of two FFT-based SOBAF filter structures are illustrated. For one adaptive filter, the set of Toeplitz equations is solved using the Levinson recursion, and for the other the fast inversion technique of [31] is used. These graphs clearly illustrate the dramatic reduction in computational load that can be achieved when the fast inversion technique is employed in the SOBAF. Thus, in common with the BLMS algorithm, the SOBAF can exhibit a significant decrease in computational load with respect to the LMS algorithm for moderate to large N.

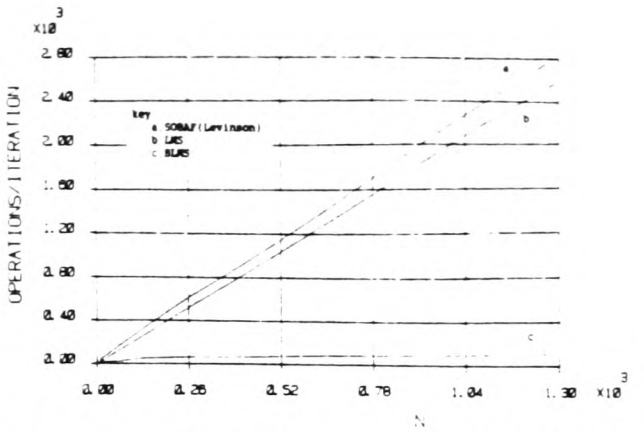


Fig. 4. Comparison of number of multiplications between SOBAF, LMS, and BLMS.

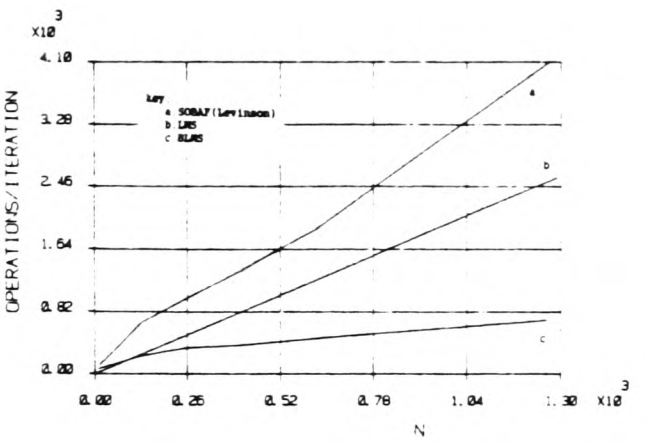


Fig. 5. Comparison of number of additions between SOBAF, LMS, and BLMS.

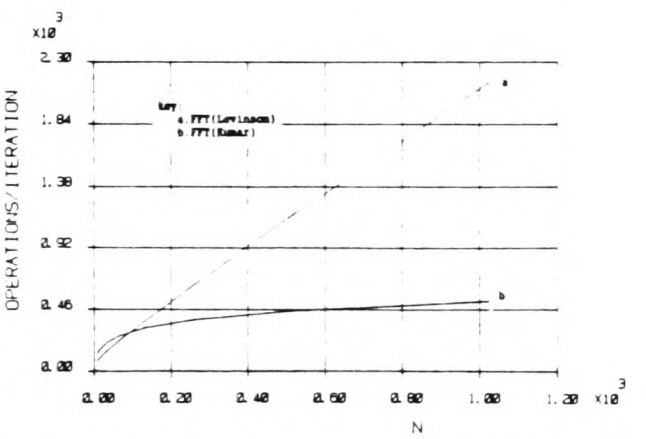


Fig. 6. Comparison of number of additions between FFT (Levinson's technique) and FFT (Kumar's technique).

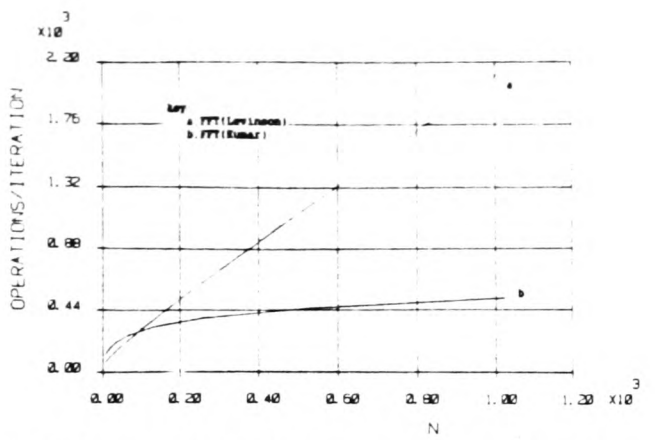


Fig. 7. Comparison of number of multiplications between FFT (Levinson's technique) and FFT (Kumar's technique).

V. Performance Comparison Simulations

The performance of the SOBAF, which is summarized in Table II, was examined using the equalizer scenario described in Section II-B and illustrated in Fig. 1. Three different channels were used in order to vary the maximum/minimum eigenvalue ratio of the autocorrelation matrix, Φ_{xx}. The three channels are defined in Table I. For all the experiments, the signal-noise ratio was set at 40 dB, and the MSE was calculated from an ensemble of 20 runs. All block adaptive filter algorithms were coded in RT form.

When the algorithm of Table II is first switched on, the quality of the autocorrelation estimate will be poor and may lead to initial divergence; see Fig. 8. This problem may be avoided by using a BLMS algorithm for the first few blocks while the quality of the autocorrelation estimte improves. The overall performance of the algorithm is not significantly degraded by this measure (Fig. 8). In all subsequent simulations the SOBAF of Table II is initialized by using a BLMS algorithm for the first four blocks of data.

It is to be expected that the performance of the practical algorithm of Table II will not be as good as the theoretical algorithm defined by (2.4), (2.7), (2.11), and (2.12), since in the former the autocorrelation matrix is estimated and in the the latter it is known *a priori*. Experiment indicates that the performance is indeed degraded but not by a significant amount; see Fig. 9.

Finally, the performance of the SOBAF was compared to that of a BLMS algorithm for the three channels described in Table I. The results are illustrated in Figs. 10–12. As the eigenvalue ratio of the input autocorrelation matrix increases, the performance of the BLMS algorithm gets poorer. The performance of the SOBAF, on the other hand, changes very little as the eigenvalue ratio is increased. This indicates that the SOBAF is insensitive to the eigenvalue spread of the input autocorrelation matrix and has an MSE convergence performance equivalent to a BLMS algorithm under white input conditions.

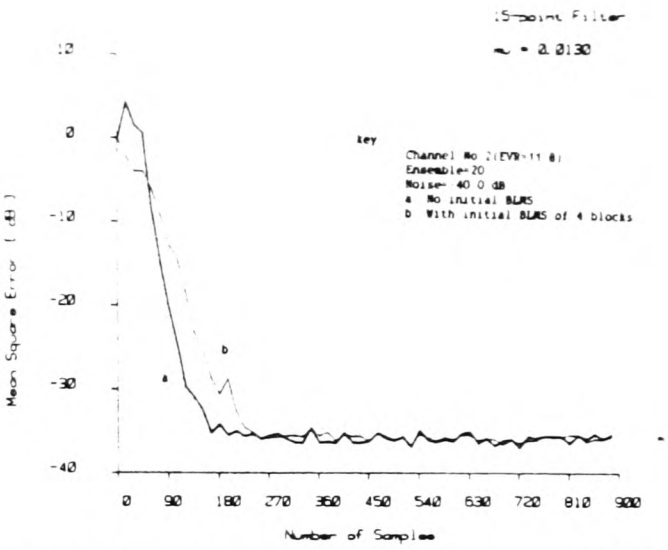


Fig. 8. Initialization effects on the convergence characteristics of SOBAF

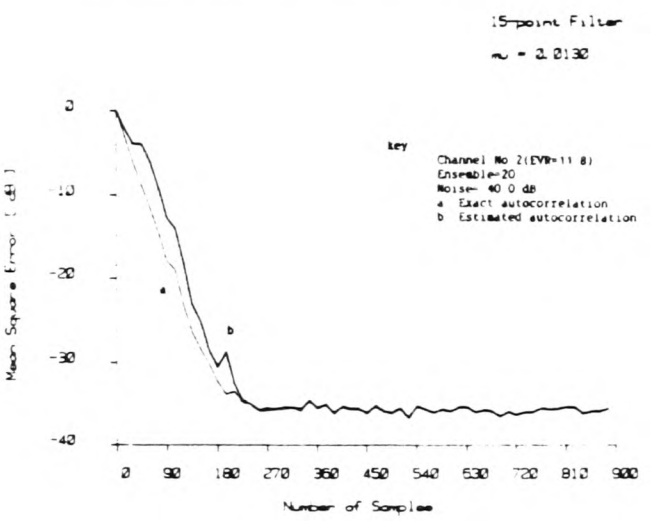


Fig. 9. Performance of exact and estimated autocorrelation.

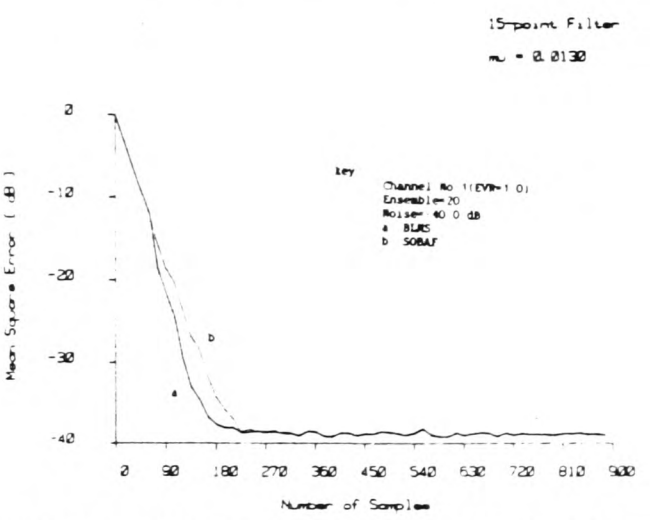


Fig. 10. Convergence characteristics of BLMS and SOBAF for channel 1

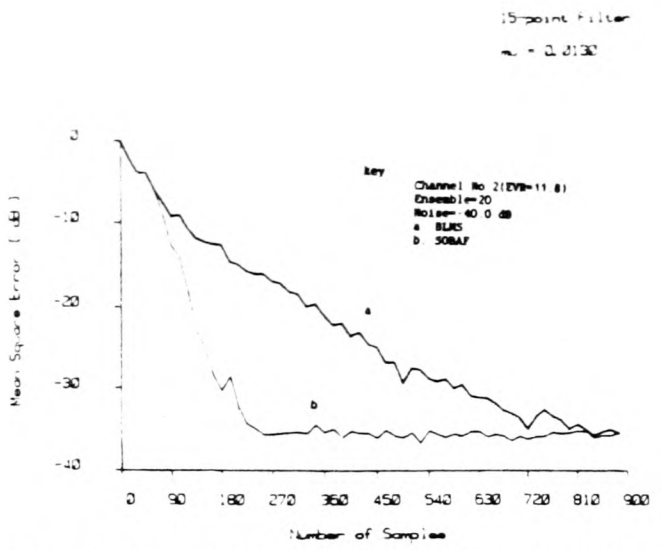


Fig. 11. Convergence characteristics of BLMS and SOBAF for channel 2.

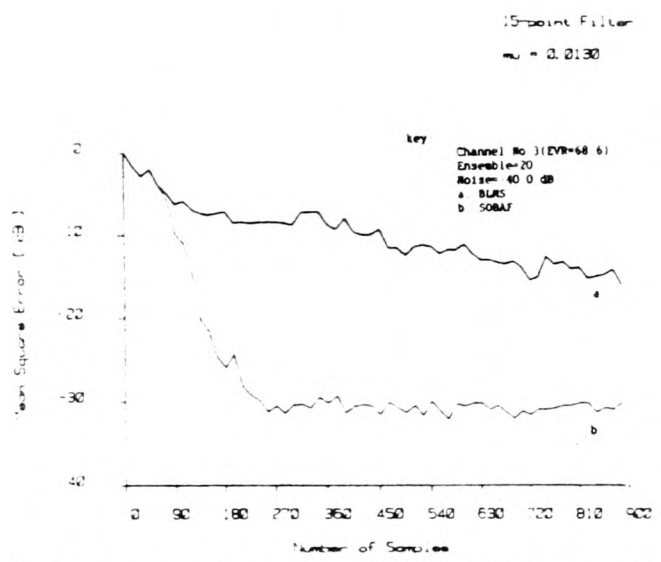


Fig. 12. Convergence characteristics of BLMS and SOBAF for channel 3.

VI. Conclusions

The SOBAF of Table II is a unique adaptive filter algorithm. In computational load, an FFT-based SOBAF is superior to an LMS algorithm for moderate to large N, being of the same order as a BLMS algorithm, i.e., $O(\log N)$. The SOBAF is thus a very efficient algorithm, computationally. The block nature of the SOBAF also permits the use of other efficient circular convolution algorithms such as the RT and the FNT. In performance, the SOBAF achieves the MSE convergence of a self-orthogonalized structure, i.e., the adaptive filter converges under any input conditions at the same rate as it would if the input was white. Further, the selection of the step size μ is more straightforward than for LMS and BLMS algorithms. This is because both the range of μ that ensures MSE convergence and the value of μ for fastest convergence are independent of the input autocorrelation matrix. In fact, for a given application, the approximate performance of the algorithm is easily predicted *a priori* from (2.6).

References

[1] G. C. Goodwin and R. L. Payne, *Dynamic System Identification. Experiment Design and Data Analysis*. New York: Academic, 1977.

[2] B. Widrow, "Adaptive filters," in *Aspects of Network and System Theory*, R. E. Kalman and N. Declaris, Eds. New York: Holt, Rinehart, and Winston, 1971.

[3] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. Contr.*, vol. 27, pp. 1-19, Jan. 1978.

[4] D. T. L. Lee, M. Morf, and B. Friedlander, "Recursive least squares ladder algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 627-641, June 1981.

[5] B. Porat, B. Friedlander, and M. Morf, "Square root covariance ladder algorithms," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 813-829, Aug. 1982.

[6] M. L. Honig, "Recursive fixed order covariance least squares algorithm," *Bell Syst. Tech. J.*, vol. 62, pp. 2916-2922, Dec. 1983, part 1.

[7] J. M. Cioffi and T. Kailath, "Windowed fast transversal adaptive algorithm with normalization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 607-625, June 1985.

[8] G. Ungerboeck, "Theory on the speed of convergence in adaptive equalizers for digital communication," *IBM J. Res. Develop.*, vol. 16, pp. 546-555, Nov. 1972.

[9] K. H. Muller and D. A Spaulding, "A new rapidly converging equalization technique for synchronous data communication," *Bell Syst. Tech. J.*, vol. 54, pp. 369-406, Feb. 1975.

[10] B. Widrow, J. McCool, M. Larimore, and C. Johnson, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151-1162, Aug. 1976.

[11] R. D. Gitlin and F. R. Magee, "Self-orthogonalizing adaptive equalization algorithms," *IEEE Trans. Commun.*, vol. COM-23, July 1977.

[12] A. Feuer and E. Weinstein, "Convergence analysis of LMS filters with uncorrelated Gaussian data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 220-230, Feb. 1985.

[13] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 609-615, June 1983.

[14] K. H. Muller, "A new fast-converging mean-square algorithm for adaptive equalizers with partial response signalling," *Bell Syst. Tech. J.*, vol. 54, pp. 143-153, Jan. 1975.

[15] R. P. Bitmead and B. D. O. Anderson, "Adaptive frequency sampling filters," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 524-534, June 1981.

[16] M. L. Honig, "Echo cancellation of voiceband data signals using recursive least squares and stochastic gradient algorithms," *IEEE Trans. Commun.*, vol. COM-33, pp. 65-73, Jan. 1984.

[17] J. W. Cooley and J. W. Tukey, "An algorithm for machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297-301, Apr. 1965.

[18] D. Mansour and A. H. Gray, Jr., "Unconstrained frequency-domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 726-734, Oct. 1982.

[19] G. Picchi and G. Prati, "Self-orthogonalizing adaptive equalization in the discrete frequency domain," *IEEE Trans. Commun.*, vol. COM-32, pp. 371-379, Apr. 1984.

[20] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 744-752, June 1981.

[21] G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realisation of FIR digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1073-1083, Oct. 1983.

[22] E. R. Ferrara, "Fast implementation of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, Aug. 1980.

[23] R. C. Agarwal and C. S. Burrus, "Fast convolution using Fermat number transforms with applications to digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 87-97, Apr. 1974.

[24] R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 392-410, Oct. 1977.

[25] J. C. Lee, B. K. Min, and M. Suk, "Realization of adaptive digital filters using the Fermat number transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1036-1039, Aug. 1985.
[26] G. Panda, B. Mulgrew, C. F. N. Cowan, and P. M. Grant, "On rectangular transform approach for BLMS adaptive filtering," presented at ICASSP '86, Tokyo, Japan, Apr. 1986.
[27] C. G. Broyden, *Basic Matrices*. London, England: Macmillan, 1975.
[28] A. Feuer, "Performance analysis of the block least mean squares algorithm," *IEEE Trans. Circuits Syst.*, vol. CAS-32, Sept. 1985.
[29] J. M. Cioffi, "The block-processing FTF adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 77-90, Feb. 1986.
[30] N. Levinson, "The Weiner RMS (root mean square) error criterion in filter design and prediction," *J. Math. Phys.*, vol. 25, pp. 261-278, Jan. 1947.
[31] R. Kumar, "A fast algorithm for solving Toeplitz system of equations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 254-267, Feb. 1985.
[32] D. Yucel, N. Tepedelenlioglu, and Y. Tanik, "A fast noniterative method for adaptive channel equalization," *IEEE Trans. Commun.*, vol. COM-31, pp. 922-927, July 1983.
[33] S. M. Kay and S. L. Marple, Jr., "Spectrum analysis—A modern perspective," *Proc. IEEE*, vol. 69, pp. 1380-1419, Nov. 1981.
[34] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.
[35] R. C. Singleton, "An algorithm for computing the mixed radix Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 93-103, June 1969.
[36] J. Durbin, "The filtering time series models," *Rev. Int. Statist. Inst.*, vol. 28, pp. 233-244, 1960.
[37] W. F. Trench, "An algorithm for the inversion of finite Toeplitz matrices," *J. SIAM*, vol. 12, pp. 515-522, Sept. 1964.
[38] S. Zohar, "The solution of Toeplitz set of linear equations," *J. ACM*, vol. 21, pp. 272-276, Apr. 1974.
[39] M. Morf, "Doubling algorithm for Toeplitz and related equations," in *Proc. ICASSP '80*, Denver, CO, pp. 956-959.
[40] F. G. Gustavson and D. Y. Y. Yun, "Fast algorithms for rational Hermite approximation and solution of Toeplitz systems," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 750-755, Sept. 1979.
[41] H. J. Nussbaumer, "Digital filtering using polynomial transform," *Electron. Lett.*, vol. 13, pp. 386-387, June 1977.
[42] J. C. Lee and C. K. Un, "Block realization of multirate adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 105-117, Feb. 1986.

Ganapati Panda received the B.Sc.Eng. (first class honours) degree in electrical engineering and the M.Sc.Eng. degree in communication systems from Sambalpur University, Sambalpur, India, in 1971 and 1977, respectively, and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, in 1982.

During 1972 he worked as an Electrical Engineer with the Industrial Corporation, Orissa, India. From 1972 to 1979 he was a Lecturer at Sambalpur University, first in the Department of Electrical Engineering and then in the Department of Electronics and Telecommunication Engineering. Subsequently in 1979 he was promoted to the post of Reader. From October 1984 to 1986 he was a Commonwealth Staff Fellow in the Department of Electrical Engineering at the University of Edinburgh, Edinburgh, Scotland, where he was involved in SERC-sponsored research contracts on digital adaptive systems, and his research interests were fast DFT and convolution algorithms, spectral estimation, and adaptive signal processing. He has a number of national and international publications in this field. He is now with Sambalpur University, Sambalpur, India.

Dr. Panda is a member of the Institution of Electronics and Telecommunication Engineers (India).

Bernard Mulgrew was born in Belfast, Northern Ireland, on August 3, 1958. He received the B.Sc. (Hons. first class) degree in electrical and electronic engineering from Queen's University, Belfast, in 1979.

He is currently completing the requirements for the Ph.D. degree in electrical engineering at the University of Edinburgh, Edinburgh, Scotland. From 1979 to 1983, he worked in the Radar Systems Department at Ferranti plc, Edinburgh, on digital hardware design and tracking algorithms for airborne radar. Since 1983 he has been a Research Associate in the Department of Electrical Engineering at the University of Edinburgh. His research interests are in adaptive signal processing and estimation theory and in their application to radar and communications systems.

Mr. Mulgrew is an associate member of the Institution of Electrical Engineers (London).

Colin F. N. Cowan (M'82) was born in Newry, Ireland, on December 6, 1955. He received the B.Sc. and Ph.D. degrees in electrical and electronic engineering from the University of Edinburgh, Edinburgh, Scotland, in 1977 and 1980, respectively.

In 1980 he was appointed as a Lecturer in the Department of Electrical Engineering, University of Edinburgh. Since that time he has been active in research on adaptive algorithms and applications and image processing. He has published some 50 papers and articles in this field and is the Co-editor of the text *Adaptive Filters* (Englewood Cliffs, NJ: Prentice-Hall, 1985). From 1984 to 1986 he was a Consultant to Hewlett Packard Ltd., Queensferry Telecommunication Division.

Dr. Cowan is a member of the Institution of Electrical Engineers (London).

Peter M. Grant (M'77-SM'83) was born in St. Andrews, Scotland, on June 20, 1944. He received the B.Sc. degree in electronic engineering from the Heriot-Watt University, Edinburgh, Scotland, in 1966, and the Ph.D. degree from the University of Edinburgh in 1975.

From 1966 to 1970 he worked as a Development Engineer with the Plessey Company Ltd., England, at both the Allen Clark Research Centre, Towcester, and Avionics and Communications Division, Havant, designing frequency synthesizers and standards for mobile military communications. In 1971 he was appointed to a research fellowship at the University of Edinburgh to study the applications of surface acoustic wave (SAW) and charge-coupled devices in communication systems. He was subsequently appointed to a lectureship and promoted to a readership with responsibility for teaching electronic circuits, signal processing, and communication systems. He leads the Signal Processing Research Group with personal involvement in adaptive filtering and pattern recognition. During 1977-1978, he was the recipient of a James Caird Travelling Scholarship, and as a Visiting Assistant Professor he researched in acoustic imaging and surface acoustic wave storage correlator applications at the Ginzton Laboratory, Stanford University, Stanford, CA. In 1985-1986 he was appointed as a Visiting Staff Member at the M.I.T. Lincoln Laboratory, Lexington, MA, studying the application of neural networks to pattern recognition.

Dr. Grant is a member of the Institution of Electrical Engineers (London), where he serves as one of the Honorary Editors of the *Proceedings of the IEE-Part F, Communications, Radar and Signal Processing*.