UNIVERSITY OF ADELAIDE LIBRARY
9/11/84

THE UNIVERSITY OF ADELAIDE

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

ADAPTIVE DIGITAL FILTERS

by

William George COWLEY, B.Sc., B.E.(Hons.)

A thesis submitted in fulfilment of the requirement for the degree of Doctor of Philosophy.

Adelaide

February, 1984

"To live effectively is to live with adequate information."

Norbert Wiener, The Human Use of Human Beings (1954)

to Margaret-Anne

and young Alexander

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Summary

Some aspects of the behaviour and applications of adaptive digital filters are considered. The structures examined are finite impulse response transversal and lattice filters, using both gradient and optimum least square adaptive algorithms. Most of the material is concerned with the lattice, rather than the transversal, structure. Topics investigated include: comparisons between different adaptive lattice algorithms, convergence models for lattice structures, an analysis of the bias and variance of all-pole spectral estimates derived from transversal and lattice structures, several radar applications of adaptive lattice filters, and some implementation considerations relevant to very large scale integration.

Statement of Originality

This thesis contains no material which has been accepted for the award of any degree or diploma in any University. To the best of the author's knowledge and belief it contains no material previously published or written by any other person, except where due reference is made in the text.

(W. G. Cowley)

Acknowledgements

The author is pleased to acknowledge the assistance of the following people:

Dr. B. R. Davis, the thesis supervisor, for many useful discussions, encouragement and expert guidance during the course of work for this thesis.

Other members of the Adelaide University EEE Department; in particular several post-graduate students for their support and stimulating discussions, and some final-year undergraduates (see Appendix D) who assisted with the ALPS project.

Mr. N. Bryans of the Microwave Radar Group, Defence Research Centre Salisbury, for his comments on the material in Chapter 7 and assistance with the data collection.

Mr. A. Trevorrow of the Adelaide University Computing Centre for his help with TEX, the typesetting system used for this thesis.

Mr. G. R. Cowley for his assistance in proofreading.

Main mathematical symbols used

$A(\omega)$	adaptive filter transfer function (subscripts r and i indicate real and imaginary parts) (*)
a_i^n	$i-$th forward linear prediction coefficient for the $n-$th order predictor (if the superscript is omitted the predictor is assumed to be $p-$th order) (*)
b_i^p	$i-$th backward linear prediction coefficient for the $n-$th (or $p-$th) order predictor
$e_{n,t}$	current forward prediction error, $n-$th order
e_t	current $p-$th order forward prediction error
$F_Q(q)$	distribution function of all-pole spectral estimates
f_n	frequency, normalised by the sampling frequency
$K_{i,t}$	current $i-$th reflection coefficient (single coefficient per lattice stage)
$K_{i,t}^e$	current $i-$th forward reflection coefficient
$K_{i,t}^r$	current $i-$th backward reflection coefficient
N	number of samples in rectangular window for same LPC statistics (see section 5.1)
N_{eff}	time constant for prediction error covariances (see section 3.2)

(*) A prime superscript ($'$) may be used with these variables to explicitly indicate they are derived from imperfect (noisy) estimates of the Linear Prediction Coefficients e.g. $Q'(\omega)$ in section 5.1.

mathematical symbols (continued)

p	order of the all-pole model
$Q(\omega)$	relative all-pole spectral estimate (*)
R	autocorrelation matrix of signal y_t
$R_t^c(i,j)$	composite autocorrelation function (see section 3.3)
$R_{n,t}^e$	current $n-$th stage forward error covariance (ELS-UP)
$R_{n,t}^r$	current $n-$th stage backward error covariance (ELS-UP)
$r(\omega)$	variance of $A'(\omega)$
r_i	autocorrelation of signal y_t at $i-$th lag
$r_{n,t}$	current $n-$th stage backward prediction error
S	scaled autocorrelation matrix inverse, see (5.2b)
$s(\omega)$	function of a_i' and frequency, see (5.7)
t	time, integer values represent multiples of the sampling interval
$v_{n,t}$	current $n-$th stage partial cross-correlation (GRAD-HM)
$v_{n,t}^f$	current $n-$th stage partial cross-correlation (GRAD-F+B)
$w_{n,t}$	current $n-$th stage mean error covariance (GRAD-HM)
$w_{n,t}^e$	current $n-$th stage forward error covariance (GRAD-F+B)
$w_{n,t}^r$	current $n-$th stage backward error covariance (GRAD-F+B)
y_t	current ($t-$th) sample of zero-mean time series
$\alpha(\omega)$	correction function for Q' statistics, see section 5.1
$\gamma_{n,t}^c$	current $n-$th stage complementary gamma term (ELS-UP)
$\Delta_{n,t}$	current $n-$th stage partial autocorrelation (ELS-UP)
$\delta(\omega)$	normalised variance of $A'(\omega)$, see (5.9b)

mathematical symbols (continued)

η	magnitude of $A(\omega_m)$ at peak ω_m
λ	exponential weighting factor
σ_i^2	$i-$th order prediction error power ($p-$th order if subscript is missing)
σ_{im}^2	variance of $A_i'(\omega)$
σ_{re}^2	variance of $A_r'(\omega)$
σ_w^2	weight variance in LMS adaptive filter
ω	frequency in radians

List of Abbreviations

ALPS	Adaptive Linear Predictor System
ALU	Arithmetic and Logic Unit
AMTI	Airborne Moving Target Indicator
CIF	Caltech Intermediate Format
CMOS	Complementary Metal Oxide Semiconductor
DFT	Discrete Fourier Transform
DMR	Delocalised Multiplier Register
ELS	Exact Least Square
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMS	Least Mean Square
LPC	Linear Prediction Coefficient
LS	Least Square
MEM	Maximum Entropy Method
MPC	Multi-Project Chip
MSE	Mean Square Error
MTI	Moving Target Indicator
NMOS	n-channel Metal Oxide Semiconductor
PEF	Prediction Error Filter
PRF	Pulse Repetition Frequency
PSD	Power Spectral Density
RSR	Recirculating Shift Register
SCV	Sub-Clutter Visibility
TFB	Transform and Filtering "Brick"
WOSA	Weighted Overlap Segment Averaging

List of Figures

List of Figures (continued)

List of Figures (continued)

UNIVERSITY OF ADELAIDE LIBRARY

Chapter 1 Introduction

1.1 Subject Coverage

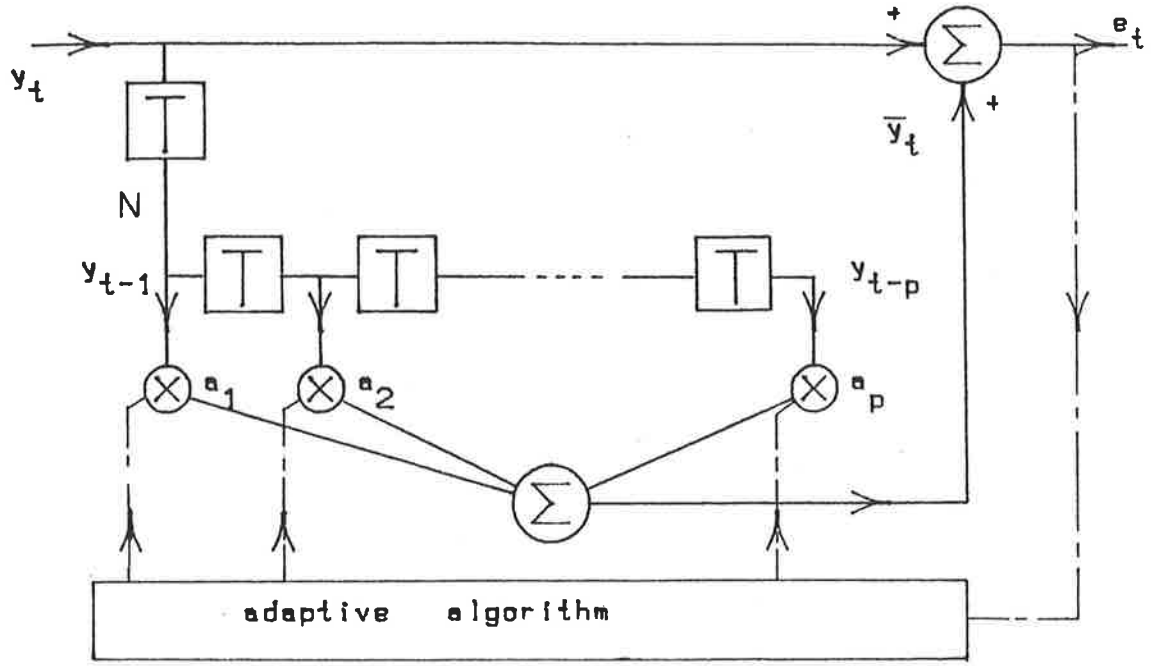
During the last two decades adaptive digital filters have become increasing important in a wide range of signal processing applications. This thesis is concerned with some aspects of these devices from a theoretical and practical point of view. Most of the material concerns understanding the behaviour of, and differences between, adaptive filters, either by analysis or simulation. In addition some implementation aspects relevant to recent advances in Very Large Scale Integration (VLSI) are considered, plus some interesting applications to radar. This chapter seeks to provide a largely descriptive introduction to the operation and uses of adaptive filters, together with some background information to the topics covered. A summary of other chapter's contents is also given.

The adaptive filters under consideration form a small subset of all digital filters. They are adaptive because their coefficients are continuously updated in response to a time sampled input signal (time series), to achieve a specified criterion, such as minimum mean square error output. Two finite-impulse-response structures are considered: the transversal and lattice adaptive filters. The second of these receives more treatment, as it has a number of important advantages over the older transversal type but is less well understood. Several algorithms for updating the filter coefficients are considered with respect to the balance between algorithm complexity (i.e. number and type of arithmetic operations) and performance according to various criteria. This is done using the theory of linear prediction and least square estimation.

1.2 Review of Adaptive Filter Operation and Applications

Adaptive transversal filters preceded their lattice counterparts and have been in use since about the 1960s. Figure 1.1 shows an example of their structure, in this case incorporated into a one step predictor. The input signal is represented by the

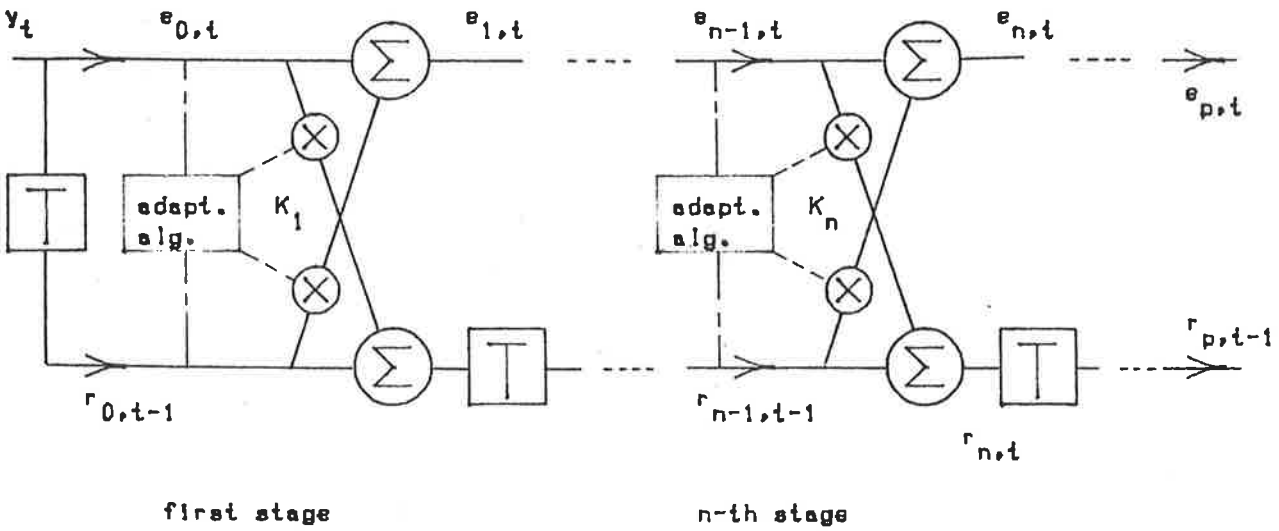
Figure 1.1 Adaptive transversal filter in linear predictor arrangement



sequence $y_0, \ldots y_t$. As these samples propagate through the tapped delay line, they are multiplied by variable coefficients $a_1, \ldots a_p$, where p is the number of stages, or order, of the filter. The weighted samples are summed and the resulting linear combination of past data samples form an estimate, $\overline{y}_t$, of the current data sample y_t. The prediction error e_t between the current data sample and its estimate is minimised by adjusting the multipliers after each new data sample, to achieve the minimum mean square error. This adjustment is carried out according to some adaptive algorithm, the simplest and most widespread of which requires only two multiplications and additions per data sample per filter stage.

The reduction of the error output may be viewed as a removal from the current sample of those correlated components which may be predicted from the last p samples. The output will then ideally be completely uncorrelated (i.e. white noise), and the filter acts as an adaptive whitening filter. The success of this whitening depends on the validity of the assumption that the current data may be represented as a linear

Figure 1.2 Adaptive lattice filter



combination of the last p samples. Assuming that the output sequence is white, its power spectral density is constant. The power spectrum of the input is therefore proportional to the reciprocal of the filter transfer function magnitude squared. This provides a means of power spectral estimation which is examined in some detail in later chapters.

The transversal filter described thus far is a one step linear predictor. Its output is a forward prediction error because its prediction is forward in time. A backwards predictor may be similarly developed. If the forward and backward predictors are combined stage by stage, a lattice structure is produced (see Figure 1.2). The multipliers are now called reflection coefficients, and again must be adjusted to minimise the error outputs from each stage. The optimum sets of forward and backward prediction errors in the lattice ($e_{0,t}, \ldots e_{p,t}$ and $r_{0,t-1}, \ldots r_{p,t-1}$) are the same as would be produced from optimum forward and backward transversal predictors. In practice the lattice exhibits some well documented advantages [9,12] over the tranversal adaptive filter, including better convergence and stability and less noisy and less biased coefficients.

There are a large number of "gradient" algorithms for updating the coefficients of adaptive filters. These algorithms are so called because the adjustments are made according to the slope of the error surface with respect to the coefficients, so as to minimise the mean square error. Whilst these algorithms may be relatively easy to implement, and might perform adequately for the application in hand, they are sub-optimum in the sense that the total mean square error is greater than that achieved by alternative block processing techniques. Recently a new class of lattice algorithms has been derived [6,10,12] which achieve the "exact least square" error solution. Comparisons of lattice performance using different adaptive algorithms, including spectral estimation capabilities, are made in later chapters.

Adaptive filters have found numerous applications. These can be broadly divided into those which use the information provided by the converged coefficients for signal modelling, spectral estimation and event detection, and those which use the error sequence for prewhitening applications. Examples of the former areas include speech analysis [8,16], seismic event detection [15] and clock synchronization [39], whilst in the latter category there are sonar [20,76] and radar (see Chapter 7) uses. Variations on the basic structure allow applications involving more than one input signal. In Figure 1.1 the input to the tapped delay line at point "N" may be fed with the second input instead of the one step delayed primary signal. This allows correlated components present in the second input to be removed from the first, and has been used for adaptive noise cancelling [3] (see Figure 1.3a). System identification or related applications are shown in Figure 1.3b and 1.3c. The former configuration is used in data communication systems to compensate for imperfect communications channels, and is called adaptive equalisation [66,31,44]. The reference input in this situation is a pre-stored training sequence. Figure 1.3c shows an arrangement used in long distance telephone circuits to cancel echoes. The echoes arise because imperfect hybrids allow part of the received signal to be returned to the sender; the trans-hybrib response represents the unknown transfer function in this case. The echo may be removed, or at least mean-square min-

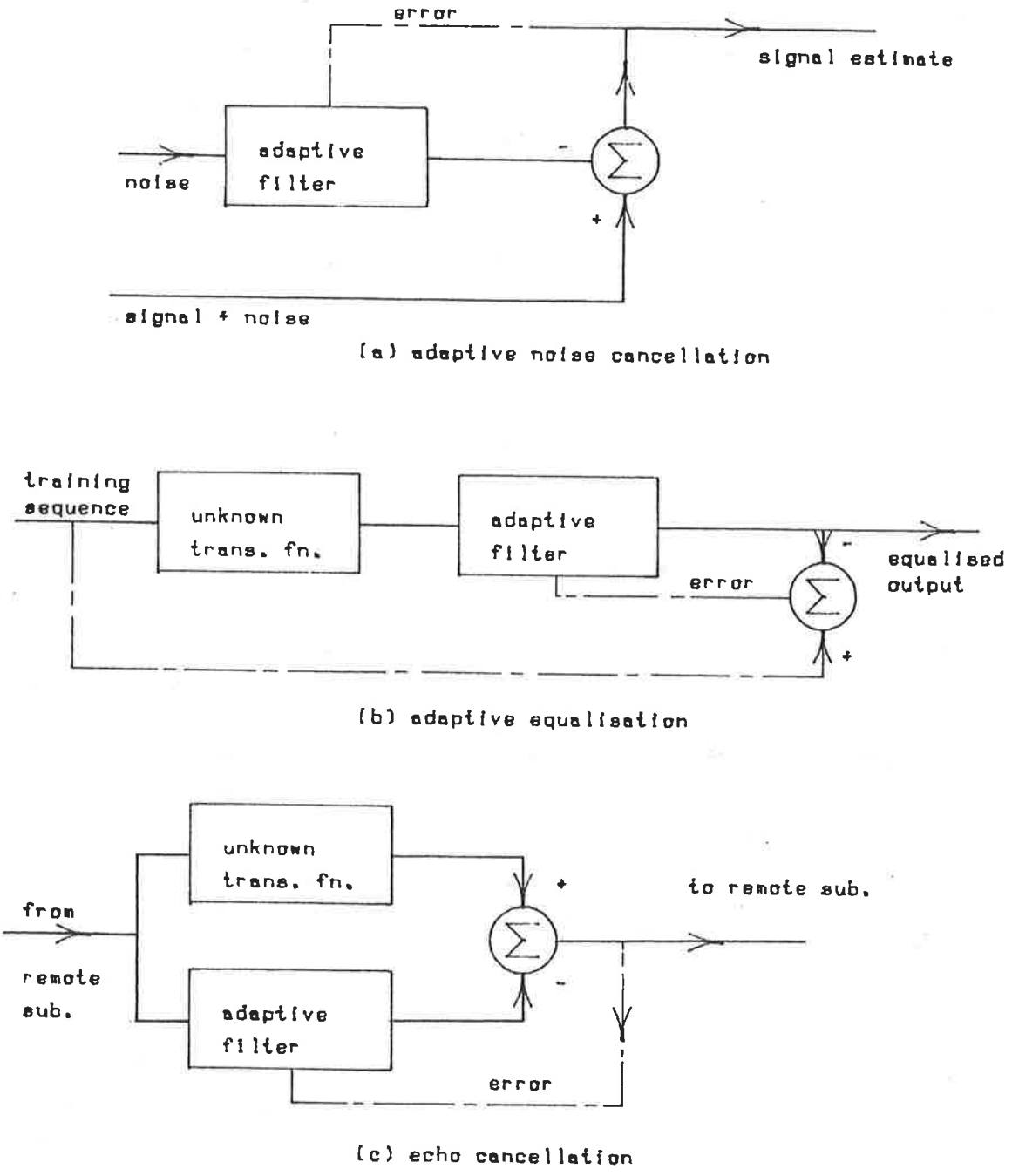


Figure 1.3 Alternative Adaptive Filter Configurations

imised, with an adaptive filter [18,19]. The same system modelling technique may be used for time delay estimation [11]. Both transversal and lattice structures have been used for many of these applications.

Another class of applications stems from the duality between time series data

and signals sampled from an array of receiving elements. Electromagnetic (or other) signals reaching the antenna array from different directions may be separated spatially by beamforming techniques, analogous to the resolution of time series components by transformation to the frequency domain. Adaptive noise cancelling methods offer alternatives to some conventional array processing methods. The weighting coefficients which linearly combine the array elements may be adjusted adaptively to cancel interference from unwanted sources [73,77]. The algorithms may include constraints to maintain the desired look direction and frequency response [72].

For many adaptive whitening and noise cancellation applications there are few alternative signal processing techniques. In applications not using the error output, adaptive filters may provide alternatives to block processing techniques. The continuously adaptive methods may be preferred because the application requires continuous estimates of system parameters, or because of better suitability for hardware implementation. The latter advantage has become more important with the advent of very large scale integration.

Some of the principle references and figures associated with the adaptive filters considered in this thesis are now listed. Widrow et al. first studied a number of issues relevant to adaptive tranversal filters such as stability, convergence rates, weight noise, excess mean square error and noise cancellation performance. Many of these results may be found in [2,3]. This stimulated much further work of the behaviour and applications of these devices [20,74,75,76], most of which used the Widrow-Hoff Least Mean Square (LMS) algorithm (see Chapter 2). Itakura and Saito were early proponents of adaptive lattice filters for speech applications [35], and recognised the orthogonality and stability characteristics of lattice structures. Makhoul, Griffiths, Haykin (see references in the author list) and others were interested in gradient algorithms for lattice structures. More recently Morf and colleagues have developed the Exact Least Square (ELS) algorithms mentioned above, and proposed many extensions and interconnections with other fields [6,12,13,24,43].

1.3 Outline of Thesis

Chapter two first presents some important concepts from the theory of linear prediction and all-pole modelling which are useful for understanding the operation of adaptive filters. These are given briefly, with references to tutorial papers where necessary. Adaptive transversal filters using the LMS algorithm are then introduced, and a summary of some new results concerning the statistics of all-pole spectral estimates is given. The discussion and derivation of these results is presented in Appendix A, reprinted from the IEEE Transactions on Acoustics, Speech and Signal Processing.

The third chapter introduces the lattice structure and is concerned with gradient algorithms. An attempt is made to show the derivation and relationships between gradient algorithms, in order to understand which might perform best. Section 3.3 examines error and coefficient convergence in adaptive lattice structures. A new method of predicting convergence, which is relevant to ELS and gradient algorithms, is discussed and tested with several computer simulations.

Chapter four initially provides a brief summary of different methods of recursive least square estimation applied to the all-pole modelling problem. This is done to give some perspective to the exact least square lattices. The derivation of these algorithms is lengthy and readily available elsewhere, so only an overview is given in section 4.2. Appendix C contains listings of other ELS lattice variants, plus an example of their behaviour. Extraction of the linear prediction coefficients from different lattices is discussed in section 4.3. A comparison is then made between an ELS and a gradient lattice algorithm, including implementation differences and performance examples using various simulated signals.

Some results concerning the all-pole spectral estimates from lattice filters are obtained in Chapter 5. These are related to the earlier results summarised in section 2.4. The emphasis is on enabling lattice parameters to be adjusted so that spectral estimates of negligible bias and suitably low variance may be obtained. For two stage

lattices simple results are possible. When low variance spectral estimates can not be achieved the all-pole spectral estimate distribution function is considered. A simulation compares the predicted distribution function to that measured from various adaptive lattices, and indicates some interesting differences.

The next two chapters mostly describe work done with real adaptive filters or on real, rather than simulated, data. Chapter 6 is a short description of a hardware implementation of an adaptive transversal filter using the LMS algorithm. This was partly constructed with a custom built VLSI circuit. Comments are made on the possibilities of using the same techniques to realise other signal processing functions, including lattice filters. Appendix D contains some supplementary information.

Some proposed radar applications of adaptive lattice filters are then discussed. This work was prompted by an existing study [17] of a lattice method for suppressing radar clutter. However difficulties arose using this method, and the extensions and new results are reported in Chapter 7. A new application of adaptive lattices, target classification by signature analysis of the radar return, is introduced. Some mention is also made of ways in which the proposed techniques could be implemented. The last chapter summarises new results and attempts to draw conclusions where appropriate.

The computer software used for the simulations of Chapters 3 to 5 forms a useful set of signal processing programs for adaptive filter investigations. Signal generation and display routines are included. Listings are not included in this thesis (over 4000 lines of code are involved) but Appendix B gives a specification of the software capabilities for potential users. Appendix E contains a brief description of the "TFB" signal processor chip currently under design at Adelaide University. This device will be well suited to adaptive filter applications.

Chapter 2 All-Pole Models and Adaptive Transversal Filters

Some basic results from the Linear Prediction approach to all-pole modelling of time series are described briefly in sections 2.1 and 2.3. These results are used frequently in the following chapters. Comprehensive treatment of this material is available in several excellent review articles [1,24,28]. The remainder of the chapter examines adaptive transversal filters, as a prelude to the adaptive lattice structures considered in later chapters. Section 2.4 summarises some new results on all-pole spectral estimates obtained from adaptive transversal filters.

2.1 All-pole signal model and Yule-Walker equations

The adaptive structures under consideration assume that the current data sample y_t of a zero-mean time series may be represented as a linear combination of past data samples $y_{t-1}, \ldots y_{t-p}$, plus an error:

$$y_t = -\sum_{i=1}^{p} a_i y_{t-i} + e_t \tag{2.1}$$

The coefficients $a_1, \ldots a_p$ are called Linear Prediction Coefficients (LPCs). The error between the actual value of the current data sample and its predicted value from the past p samples is e_t. The error sequence is assumed to be uncorrelated white noise of uniform power spectral density. This widely used model of the signal is called the *all-pole* or autoregressive model. It assumes that the signal may be generated by an all-pole (recursive) digital filter whose input is the white sequence $e_0, \ldots e_t$. Figure 2.1 shows the all-pole model generating filter. Taking the z transform of (2.1) gives the all-pole model transfer function.

$$H(z) = \frac{Y(z)}{E(z)} = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{2.2}$$

In general the linear prediction coefficients are unknown apriori, and must be determined from the data. This may be done by " block" methods, in which a block of

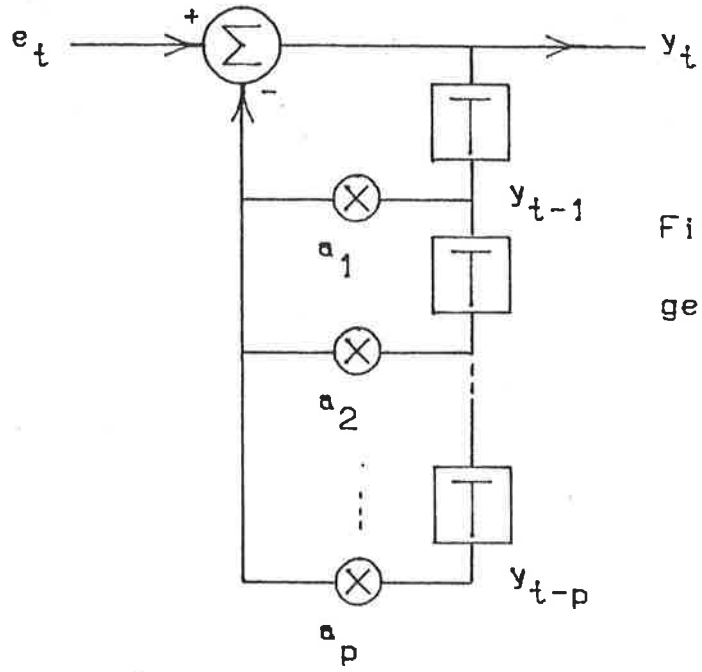


Figure 2.1 All-pole generating model

samples is analysed to give one estimate of the model parameters, or by continuously adaptive methods, in which an updated estimate is generated for each new data sample. The adaptive filters under discussion form an important means of performing the latter function. However an analysis of the block methods is useful in understanding and deriving the adaptive solutions.

In passing it should be said that the all-pole signal model is not well suited to modelling all time series. A more general model includes nonrecursive paths in the model generating filter, which leads to a polynomial in z^{-1} in the numerator of (2.2); this is the pole-zero or autoregressive moving average model [1,5,28]. Whilst any signal can be modelled by an all-pole model of high enough order [1], this may not be feasible or desirable in practice.

To find the optimum LPCs for a stationary, infinite duration time series, the prediction error of (2.1) may be minimised in a mean square sense. If the partial derivative of $E\{e_t^2\}$ is taken with respect to each of the a_i, for $i = 1,..p$ and set equal

to zero, we have:

$$E\left\{\frac{\partial e_t^2}{\partial a_i}\right\} = E\left\{2\frac{\partial e_t}{\partial a_i}e_t\right\} = 0, \qquad i = 1,\ldots p \tag{2.3}$$

thus from (2.1)

$$E\{y_{t-i}e_t\} = 0, \qquad i = 1,\ldots p \tag{2.4}$$

The last equation is a statement of the orthogonality principal of linear mean square estimation, which states that the error must be orthogonal to the available data. Substituting (2.1) into (2.4) leads to a set of p equations, called the Yule-Walker or normal equations. These may be written in matrix form as follows.

$$\begin{pmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} = - \begin{pmatrix} r_1 \\ \vdots \\ r_p \end{pmatrix} \tag{2.5a}$$

where $r_i = E\{y_t y_{t-i}\}$ for $i = 1,\ldots p$ is the autocorrelation function of the signal y_t at the $i-$th lag. An additional equation is obtained by multiplying (2.1) by y_t and taking expected values:

$$r_0 = -\sum_{i=1}^{p} a_i r_i + \sigma^2 \tag{2.5b}$$

where $\sigma^2 = E\{e_t^2\}$ is the error (e_t) variance.

One method of linear prediction analysis involves first estimating the autocorrelation function, and then solving (2.5a). The direct solution $\underline{a} = -R^{-1}\underline{r}$, where the variables may be interpreted from (2.5a), is a matrix form of the Wiener Hopf equation [84]. Instead of calculating the R matrix inverse directly to find the LPCs, the Toeplitz nature of R (same elements on any leading diagonal), may be exploited by using a set of recursive equations known as Levinson-Durbin recursion. (For derivation see, for example, [24] or [22] section 2.4.)

$$\sigma_0^2 = r_0 \tag{2.6a}$$

$$K_i = \frac{-\left(r_i + \sum_{j=1}^{i-1} a_j^{i-1} r_{i-j}\right)}{\sigma_{i-1}^2} \tag{2.6b}$$

$$a_i^i = K_i \tag{2.6c}$$

$$a_j^i = a_j^{i-1} + K_i a_{i-j}^{i-1}, \quad 1 \le j \le i-1 \tag{2.6d}$$

$$\sigma_i^2 = (1 - K_i^2)\sigma_{i-1}^2 \tag{2.6e}$$

The equations are solved for increasing predictor orders by repeating (2.6b) to (2.6e) for $i = 1, \ldots p$, where a_i^n is the $i-$th LPC for the $n-$th order predictor. (In future when the order superscript is omitted for notational simplicity, the assumed order will be p; the maximum order of the all-pole model.) An additional set of variables occurs in these equations: K_i, $i = 1, \ldots p$ are called reflection coefficients or partial correlation (PARCOR) coefficients and are directly related to the lattice structure (see Chapter 3). σ_i^2 is the error power output for the $i-$th order predictor. Approximately $p^2 + O(p)$ multiplications are required for these recursions, compared to $O(p^3)$ for direct matrix inverse solutions.

We may note that the above solution was for a stationary input signal available for all time. This formulation of linear prediction analysis is called the autocorrelation method, and generates a Toeplitz R matrix which enables the basic Levinson recursion to be used. With a finite number of data points used to generate the autocorrelation function estimates, this method applies an effective pre and post window to the data. Alternative methods exist (e.g. "covariance method") which do not impose this restriction, but take more operations to solve. In Chapter 4 some of these methods are examined in relation to the exact least squares lattice derivation.

2.2 Prediction Error Filters and the LMS Algorithm

We will now consider how the Yule-Walker equations may be solved iteratively, using an adaptive transversal filter and a simple algorithm to update the coefficients. The structure of Figure 1.1 is described by the all-pole model (2.1). As its output is a prediction error, it is also known as a (forward) Prediction Error Filter (PEF). The variable coefficients are frequently called weights. The structure has feedforward paths only and therefore is an all zero filter. It is, in fact, an inverse filter to the all-pole

generating filter of Figure 2.1; its transfer function is the reciprocal of that of the signal synthesis filter. Its zeros should cancel the poles added by the generating filter, and in this case its output will be an uncorrelated sequence, of power equal to that of the generating filter input.

A simple gradient algorithm may be derived by considering the partial derivative of the current error squared with respect to the weights. In (2.3) the derivative of the expected error squared was used; now if the expectation is dropped, the quantity $y_{t-i}e_t$ represents the "instantaneous" rate of change with respect to a_i. Following the "method of steepest descent" [21], the weight a_i is updated by subtracting a small number multiplied by the current error gradient. This has the effect of moving the weight in a direction opposite that of greatest error increase. Thus the mean square error should decrease with each iteration during convergence. This is called the Widrow-Hoff LMS algorithm :

$$e_t = y_t + \sum_{i=1}^{p} a_{t-1,i} y_{t-i} \tag{2.7}$$

$$a_{t,i} = a_{t-1,i} - 2\mu e_t y_{t-i}, \qquad i = 1, \ldots p \tag{2.8}$$

where the first equation is simply (2.1) with added time indexes and $a_{t,i}$ is the $i-$th adaptive coefficient at time t. The parameter μ is called the adaption constant and controls both the stability and convergence rate. The great advantage of the LMS algorithm is its simplicity; only $\approx 2p$ multiplications are required per new time sample to evaluate (2.7) and (2.8).

Performance characteristics of the LMS algorithm such as stability, convergence rate, excess mse and weight vector noise, have been widely studied and many analytical results are available. [2,3,4,20] Some of these may be found in Appendix A, section 1. The terminology used there is slightly different as it aligns closely with [2,3], however no difficulty should be found in relating it to the more general nomenclature of this chapter.

An important conclusion drawn from studies of the LMS algorithm is that the statistics of the input signal have a large effect on performance; in particular the eigenvalues of the signal autocorrelation matrix determine stability and convergence speed. A large spread in eigenvalues implies that a lower adaption constant is required to maintain stability, long convergence times for the mean square error (MSE) to reach its optimum value and possibly, bias in the converged weights. As the ratio of maximum to minimum eigenvalues is approximately equal to the spectral dynamic range [1], these problems become more apparent for highly coherent input signals. An interesting geometric interpretation of the eigenvalue spread is possible. In the p dimension space formed by the weight errors, i.e. estimated a_i − ideal a_i, the surface of constant mean square error forms a (hyper)ellipsoid which becomes more elongated (less spherical) as the eigenvalues diverge, because each ellipsoid axis length is related to one eigenvalue [21]. The method of steepest descent, which moves the weights in the direction of maximum error decrease (i.e. perpendicular to the ellipsoid surface), is thus less likely to point directly towards the origin as the ellipsoid becomes more elongated, and so more steps will be needed to converge to minimum MSE.

The LMS algorithm is not the only possibility for the adaptive structure of Figure 1.1, although it is the most commonly used. A number of variations on the basic LMS algorithm exist to further simplify its implementation, for example use of only the error sign to effect a small constant-size weight change [18], or updating only one weight instead of p weights in each sampling interval [77]. Alternatively, the technique of gradient averaging mentioned in Appendix A was found to decrease weight bias and numerical accuracy problems. Adaptive algorithms with better performance than the LMS (for the transversal structure) may be obtained by multiplying the second term in (2.8) by an estimate of the signal autocorrelation matrix inverse; this becomes the realm of recursive least square estimation and Fast Kalman algorithms, and is mentioned further in Chapters 4 and 8. Other adaptive algorithms for determining the optimum weights of a linear combiner have performance inferior to the LMS algorithm, but may

be used on a wider class of problems (such as when the error signal is not available, [83]).

It will be seen in Chapter 3 that many of the difficulties associated with transversal filters using the LMS algorithm do not occur with the lattice structures, and this gives them a significant advantage. The penalty, as usual, is increased algorithm complexity and number of computations. A simulation of the MSE convergence (learning curve) is included in Figure 3.3 for comparison with lattice results. The mean error powers from transversal and lattice filters are plotted from startup, and an abrupt change in the input signal is included at time t= 50 (see section 3.2). The superiority of the lattice filters, both from convergence rate and excess MSE criteria, is evident.

2.3 All-pole Spectral Estimates

The all-pole or autoregressive spectral estimate may be derived from (2.2). Taking the complex magnitude squared, and substituting $z = e^{j\omega}$ to evaluate on the unit circle in the z plane, gives the all-pole spectral estimate:

$$P(\omega) = |Y(e^{j\omega})|^2 = \frac{|E(e^{j\omega})|^2}{|1 + \sum_{k=1}^{p} a_k e^{-jk\omega}|^2} \tag{2.9}$$

The numerator of (2.9) is simply the error power spectrum, which will be a constant because the output error is white. Since we are usually only interested in relative power spectra rather than absolute, a relative all-pole estimate (following Griffiths [4]), may be defined:

$$Q(\omega) = \frac{1}{|1 + \sum_{k=1}^{p} a_k e^{-jk\omega}|^2} \tag{2.10}$$

The spectral estimate Q is a function, at any frequency, only of the estimated LPCs. Bias and variance of the estimate are thus determined by the statistics of the adaptive filter coefficients. In the next section the quality of Q estimates is examined for the adaptive transversal filter case.

Some general comments about the all-pole spectral estimate are useful at this stage. It has been shown [25] that the all-pole spectral estimate is equivalent to the max-

imum entropy method (MEM) of spectral estimation for Gaussian random processes. The latter method was derived by Burg [26] in the late 1960s as an alternative to conventional (e.g. periodogram) methods. In particular it achieved greater resolution than existing methods. The MEM seeks to extrapolate the known autocorrelation function of the signal so that the entropy of the probability density function is maximised at each step. This has been interpreted as imposing the fewest constraints whilst maintaining consistency with the available data [27,28,23]. As indicated by the comments in section 2.1, this class of spectral estimate is not suited to signals poorly represented by low-order all-pole models (such as sinusoids in appreciable noise). The selection of model order may be an important consideration, and has been addressed by Akaike and others [1,23]. Another potential difficulty is that the method does not estimate relative powers between different components easily; Lacoss [82] has suggested that for narrowband components the area under spectral peaks is proportional to power, rather than peak amplitude.

2.4 Spectral Estimate Statistics for the Adaptive Transversal Filter

The bias and variance of the (relative) all-pole spectral estimate $Q(\omega)$ are analysed in Appendix A. Expressions are derived for these quantities when the adaption constant is sufficiently small, and predicted results compared to simulated results with reasonable agreement. The analysis assumes that under the LMS algorithm the weight noise components are unbiased, mutually uncorrelated and of equal variance. These assumptions have been shown to be reasonable [3 (Appendix D), 20,30] if the input signal dynamic range is not too great and the adaption constant is small. A summary of the results follows:

$$\text{if} \quad A(\omega) = 1 + \sum_{k=1}^{p} a_k e^{-jk\omega} \tag{2.11}$$

and $A'(\omega)$ is similarly defined, and produces $Q'(\omega)$, using the noisy versions of ideal coefficients a_k,

$$\text{and} \quad \delta(\omega) = E\left\{ \frac{|A'(\omega) - A(\omega)|^2}{|A(\omega)|^2} \right\} \tag{2.12}$$

(i.e. δ is the normalised variance of the adaptive predictor transfer function), also if σ_w^2 is equal to the weight variance, then

$$1) \quad \delta(\omega) = p\sigma_w^2 Q(\omega) \tag{2.13}$$

$$2) \text{ if } \delta(\omega) << 1 \qquad E\{Q'\} - Q \simeq (2\alpha - 1)\delta Q \tag{2.14}$$

$$E\{(Q' - Q)^2\} \simeq 2\alpha\delta Q^2 \tag{2.15}$$

where

$$\alpha(\omega) = 1 + \frac{\sin p\omega}{p \sin \omega} \cos\left((p+1)\omega + 2\theta_A\right) \tag{2.16}$$

$$A = |A|e^{j\theta_A} \tag{2.17}$$

a function whose range is 0 to 2.

The maximum bias and variance occur at maximum Q, as δ is proportional to Q. However, provided $\delta(\omega) << 1$ for all ω, the bias is usually negligible and the normalised variance is small (i.e. $\approx 2\delta$). As δ is a function of the adaption constant (Appendix A, (7)), the analysis enables a suitable value of μ to be selected so that spectral estimates of adequately low bias and variance are obtained.

Situations for which this spectral estimation technique and results might be useful are those requiring very frequent on-line spectral estimates, using simple dedicated hardware with high data rate signals suitable for all-pole modelling. A VLSI signal processor such as that of Adelaide University's TFB (currently being designed, see Appendix E), could perform 30 pole modelling, using the LMS algorithm, at over 30 kHz sampling rates, and periodically provide sets of LPCs to a general purpose microprocessor for spectral estimate calculation and subsequent analysis.

The type of application just outlined would make fairly inefficient use of the data and thus not be suitable for all situations (e.g. when the data was only locally stationary

or limited in duration). This is due to the inherent limitation of the LMS algorithm requiring a low adaption constant (to maintain stability and low weight noise) and thus long convergence times. The lattice filter can achieve much better performance and would thus be preferred. The statistics of the all-pole spectral estimates from adaptive lattices are examined in Chapter 5.

Chapter 3 Adaptive Gradient Lattices

This chapter examines the lattice structure and its relation to the Prediction Error Filters from Chapter 2. Several adaptive gradient algorithms are introduced, their relative merits discussed and performance examined using simulation results. Existing studies of convergence are surveyed, and a new method of analysing lattice convergence is presented. It is shown how this method may be used to predict coefficients and MSE convergence for a range of input signals, including certain non-stationary signals.

3.1 Lattice Structure

It is useful to review briefly how the lattice structure, shown in Figure 1.2, is related to the Prediction Error Filter (PEF). The adaptive transversal filter of Figure 1.1 has already been identified as a forward PEF described by (2.7). A backwards PEF may be similarly constructed; its task is to predict y_{t-p} from $y_t,\ y_{t-1}, \ldots y_{t-p+1}$

$$r_{p,t} = y_{t-p} + \sum_{i=0}^{p-1} b_i^p y_{t-i} \tag{3.1}$$

where b_i^p, for $i = 0, \ldots p-1$, are the pth order backward prediction coefficients. The 1 step delayed backward error is actually more useful than $r_{p,t}$, i.e.

$$r_{p,t-1} = \sum_{i=0}^{p} b_i^p y_{t-i-1} \tag{3.2}$$

where b_p^p is equal to one. The equivalent expression for the forward error is

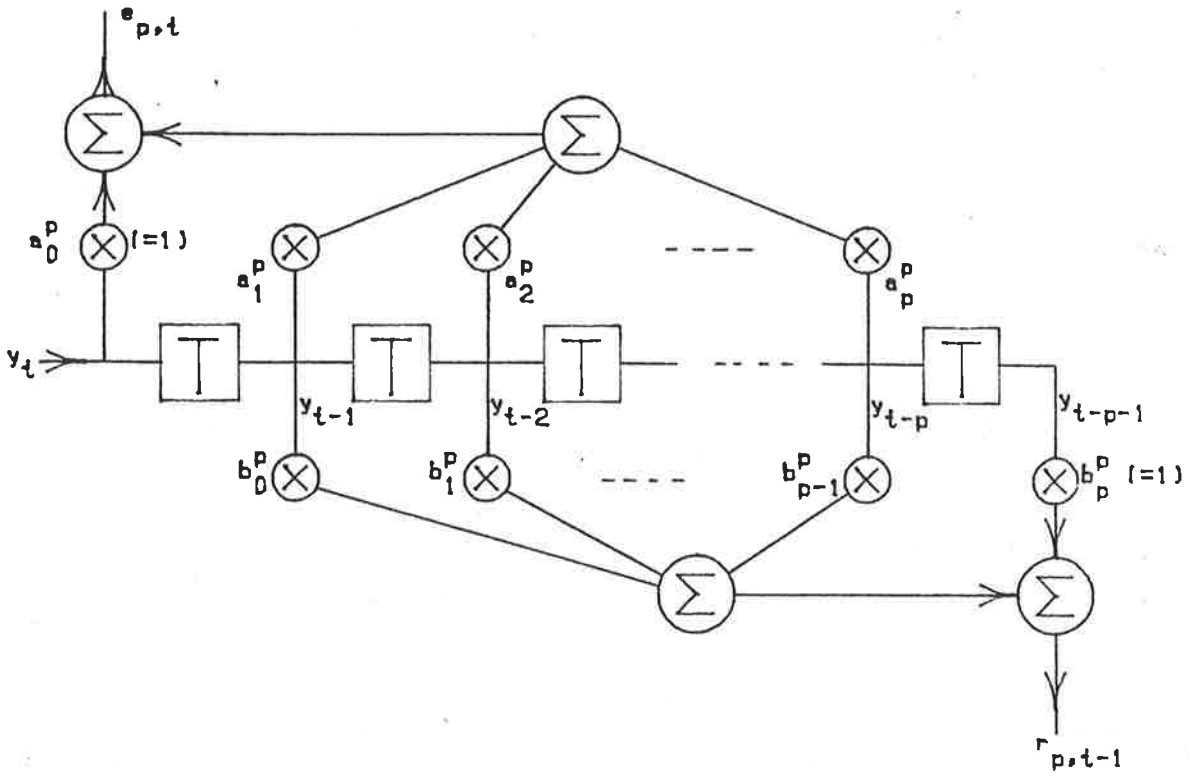
$$e_{p,t} = \sum_{i=0}^{p} a_i^p y_{t-i} \tag{3.3}$$

where a_0^p equals one. The forward and delayed backwards errors are generated by linear combinations of the same data, $y_{t-1}, \ldots y_{t-p}$. This is depicted in Figure 3.1 .

For stationary signals the optimum backward prediction coefficients are equal to the optimum forward predictor coefficients in reversed order.

$$b_i^p = a_{p-i}^p, \qquad i = 0, \ldots p \tag{3.4}$$

Figure 3.1 Forward and backward prediction error filters



This may be shown by using (3.1) to derive the normal equations following the method used in Chapter 2. If the resulting set of equations is compared to (2.5a) the above equality is apparent.

The basic lattice structure may now be generated by considering how the $n-$th stage forward error may be produced from the preceding stage. Substituting n for p in (3.3) and using (2.6d) from the Levinson recursion, which relates LPCs of adjacent orders, we have

$$e_{n,t} = \sum_{i=0}^{n} a_i^{n-1} y_{t-i} + K_n \sum_{i=0}^{n} a_{n-i}^{n-1} y_{t-i} \tag{3.5}$$

As a_n^{n-1} is zero, the first term in (3.5) is $e_{n-1,t}$. Using (3.4) and (3.2) the second term may be rearranged and found equal to $K_n r_{n-1,t-1}$. Thus,

$$e_{n,t} = e_{n-1,t} + K_n r_{n-1,t-1} \tag{3.6}$$

In a similar way, starting from (3.1), the following relation concerning backward errors

of consecutive orders may be established:

$$r_{n,t} = r_{n-1,t-1} + K_n e_{n-1,t} \tag{3.7}$$

These two equations define the $n-$th stage of a lattice structure shown in Figure 1.2. The p reflection coefficients $K_1, \ldots K_p$ now form the parameters which must be selected for optimum performance in an adaptive lattice filter, i.e. optimum error minimisation. From the derivation given above the forward and backward errors propagating along the lattice are the same as would be found from corresponding forward and backward PEFs, assuming optimum coefficients in all cases. The error powers must therefore decrease, or remain constant, from left to right as the prediction order increases. This is shown by (2.6e). Note that the above approach to the lattice derivation assumed stationary signals of unlimited duration, with the minimisation being in terms of expected values.

Several other characteristics of lattice filters may be briefly noted. Whereas the PEF performs a single or global error reduction, the lattice achieves the same result stage by stage, for successively higher prediction orders, by adjusting each reflection coefficient for optimum error power decrease from its stage. Successive stages in the lattice do not effect preceding stages, and so the $p-$th order lattice contains the optimum predictors for all orders 1 to p. The independence of successive lattice stages may be considered in the following way [33]. Suppose $e_{n,t}$ is available and is y_t minus the best linear combination of data y_{t-1} to y_{t-n}, (denoted by best $LC\{y_{t-1}, \ldots y_{t-n}\}$). Similarly $r_{n,t-1}$ is available and is y_{t-n-1} minus the best $LC\{y_{t-1}, \ldots y_{t-n}\}$. $r_{n,t-1}$ must therefore represent y_{t-n-1} without those components which may be predicted from y_{t-1} to y_{t-n}. Thus to form $e_{n+1,t}$ (y_t − best$LC\{y_{t-1}, \ldots y_{t-n-1}\}$), it is sufficient to take $e_{n,t}$ plus a constant times $r_{n,t-1}$; this will add the appropriate component due to y_{t-n-1} and not affect the linear combination of the remaining past data samples. By similar arguments the delayed backward errors are mutually uncorrelated.

The all-pole model filter will be stable if all its poles lie inside the unit circle. This corresponds to the zeros of the lattice filter, or associated PEFs, lying inside the

unit circle. A useful condition for stability of the all-pole model filter follows from this [1,35]

$$|K_i| < 1 \qquad i = 1, \ldots p \tag{3.8}$$

Several alternative lattice forms may be derived with only one multiplier per stage [31]. These do not seem as useful in continuously updated lattice filters due to the extra difficulty in updating the modified reflection coefficients. Finally the orthogonality of the lattice permits a useful factorisation of the inverse autocorrelation matrix, in terms of upper and lower triangular matrices composed of LPCs of predictor orders 1 to p [31,52]. This is used in Chapter 5.

3.2 Gradient Lattice Algorithms

Having defined the lattice structure, it remains to determine suitable algorithms for updating the reflection coefficients after each new data sample. Several solutions to this problem exist [31,32,33,34] and a number of these are now considered in a unified framework.

The most obvious technique is to minimise the sum of the forward plus backward error powers for each stage. If the expected value of $e_{n,t}^2 + r_{n,t}^2$ is differentiated with respect to K_n, using (3.6) and (3.7), and the result set equal to zero, we have

$$K_n = \frac{-2E\{e_{n-1,t} r_{n-1,t-1}\}}{E\{e_{n-1,t}^2 + r_{n-1,t-1}^2\}}, \quad n = 1, \ldots p \tag{3.9}$$

The optimum reflection coefficient is seen to be a normalized cross-correlation between the forward and delayed backward errors from the previous stage. Burg's method of maximum entropy analysis (see section 2.3) uses this approach of backward plus forward error minimisation in a block method; the expectations of (3.9) are replaced by summations over the available data and the reflection coefficients are generated one stage at a time.

In a continuously updated lattice a widely used method of estimating $K_{n,t}$, the current $n-$th stage reflection coefficient, is to estimate the denominator and numerator

of (3.9) separately. The following equations (following [7]), are evaluated for each new data sample.

$$v_{n,t} = \lambda v_{n,t-1} - 2e_{n-1,t}r_{n-1,t-1} \tag{3.10a}$$

$$w_{n,t} = \lambda w_{n,t-1} + e^2_{n-1,t} + r^2_{n-1,t-1} \tag{3.10b}$$

$$K_{n,t} = \frac{v_{n,t}}{w_{n,t}} \tag{3.10c}$$

λ is a weighting factor between zero and one whose effect is to forget the influence of old data and thereby enable adaptive response under non-stationary input signals. The $K_{n,t}$ update may be rewritten from (3.10):

$$K_{n,t} = \frac{-2\sum_{j=1}^{t}(\lambda^{t-j}e_{n-1,j}r_{n-1,j-1}) + \lambda^t v_{n,0}}{\sum_{j=1}^{t}\lambda^{t-j}(e^2_{n-1,j} + r^2_{n-1,j-1}) + \lambda^t w_{n,0}} \tag{3.11}$$

where $v_{n,0}$ and $w_{n,0}$ are the initial values used in (3.10). This scheme provides one pole low pass filtering of the denominator and numerator estimates of (3.9). (Other weighting functions have been used, for example in speech analysis [8].) The value of λ determines the cutoff of this filter; λ closer to 1 indicates more stable, less noisy, estimates of the expectations because of the large number of samples contributing to the estimate, however, this is offset by the larger convergence time. Small λ implies fast but relatively noisy response.

Taking expected values of (3.10a), and assuming stationarity of the predictor errors, shows the mean value of v_n equals the numerator of (3.9) multiplied by N_{eff}, where

$$N_{eff} = \frac{1}{1-\lambda} \tag{3.12}$$

Similarly for (3.10b), N_{eff} is the number of data samples in a rectangular window which would give the same mean v_n and w_n, assuming stationarity. In addition it is easily shown that for λ close to 1, N_{eff} is the time constant associated with the exponential convergence of v_n and w_n. Note, however, that N_{eff} is not the time constant for K_n, whose convergence is more complicated and is examined in the next section.

The exponential weighting defined by (3.10) leads to the same expression for the reflection coefficient (3.11), as a block minimisation of the exponentially weighted error squared sum, that is if

$$E_n(t) = \sum_{j=1}^{t} (e_{n,j}^2 + r_{n,j}^2)\lambda^{t-j} \tag{3.13}$$

is minimised with respect to $K_{n,t}$ the result is (3.11), assuming zero initial conditions. Gradient algorithms do not achieve the minimum value of $E_n(t)$ however, except for the first stage, because all reflection coefficients are being continuously adjusted (and in turn affecting higher order stages) with each new data sample, rather than by stage by stage block minimisation. Hence the prediction errors at any given time for the block method to which (3.13) applies, will not equal those obtained using (3.10). It will be shown that certain gradient algorithms nevertheless achieve close to the optimum performance in many situations. It will also be seen that the exact least squares algorithms, whilst still continuously updating the lattice coefficients, achieve their optimum performance by adjusting the weighting given to each new prediction error, unlike the uniform weighting used by the gradient lattices.

The algorithm of (3.10) appears to bear little resemblance to the LMS gradient algorithm of Chapter 2, but the similarity can be shown as follows. Using a similar "gradient" approach as previously:

$$K_{n,t} = K_{n,t-1} - \beta \frac{\partial(e_{n,t}^2 + r_{n,t}^2)}{\partial K_{n,t}} \tag{3.14}$$

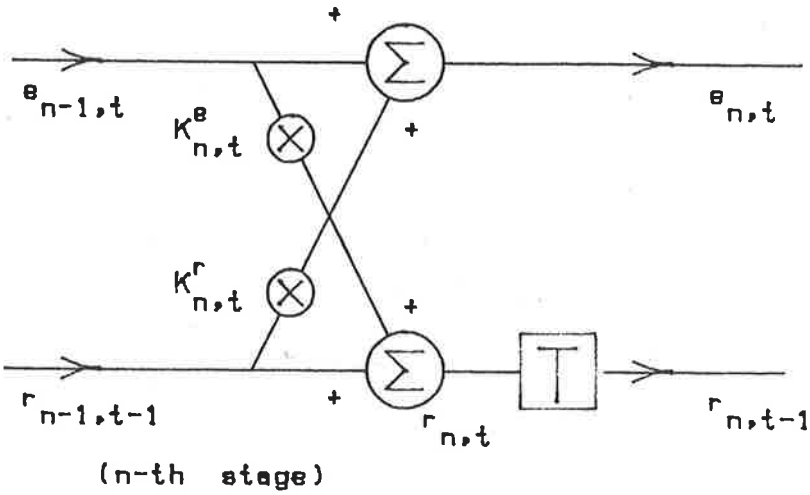
where β is a small number corresponding to μ in (2.8). Evaluating this equation using (3.6) and (3.7) and neglecting terms in β^2 leads to

$$K_{n,t} = K_{n,t-1}\left(1 - 2\beta(e_{n-1,t}^2 + r_{n-1,t-1}^2)\right) - 4\beta e_{n-1,t} r_{n-1,t-1} \tag{3.15}$$

Now if (3.10) is rearranged the reflections coefficient update may be written as

$$K_{n,t} = K_{n,t-1}\left(1 - \frac{(e_{n-1,t}^2 + r_{n-1,t-1}^2)}{w_{n,t}}\right) - \frac{2e_{n-1,t} r_{n-1,t-1}}{w_{n,t}} \tag{3.16}$$

Figure 3.2 Lattice filter with two reflection coefficients per stage



Thus, if the adaption constant β is made equal to the reciprocal of twice $w_{n,t}$ the algorithms will be the equivalent except at startup when β will not be very small. The advantage of the original algorithm is that the single parameter λ may be chosen independently of signal power, and allows a clearer interpretation of the exponential error weighting. This class of adaptive lattice algorithms is sometimes called "normalized algorithms". They are generally preferred over those algorithms related to (3.15) [7,36,34], although in some cases, those in the latter class may have some computational advantages. (For example if β is fixed at some small number in (3.15) no divisions are required, however a suitable value of β must still be determined for each stage and performance would generally be degraded.)

We will now consider adaptive algorithms which minimise the forward and backward error powers separately, rather than as a single sum. This approach leads to two reflection coefficients per stage as shown in Figure 3.2. Minimising $E\{e^2_{n,t}\}$ with respect to K^r_n gives

$$K^r_n = \frac{-E\{e_{n-1,t}r_{n-1,t-1}\}}{E\{r^2_{n-1,t-1}\}} \tag{3.17}$$

Similarly, minimising the backward error power gives

$$K_n^e = \frac{-E\{e_{n-1,t} r_{n-1,t-1}\}}{E\{e_{n-1,t}^2\}} \tag{3.18}$$

So far the denominators of (3.17) and (3.18) have been assumed to be of equal magnitude, so the reflection coefficients will be equal. This may not be the case for non-stationary signals, or during convergence for a stationary signal. Algorithms employing separate forward and backward reflection coefficients might therefore be expected to perform better under these conditions.

A potential difficulty with (3.17) and (3.18) is that the reflection coefficients may be greater than one in magnitude. However, their product is always less than or equal to one (by considering their joint second order moments, [38] p.210) so one of them, at least, must be less than one in magnitude. This leads to the so called "minimum method", in which the smaller is always used. Other possibilities involve using a generalised mean value of K_n^e and K_n^r, whose mean magnitude is always less than 1 [32]. One of these is the harmonic mean:

$$K = \frac{2K^e K^r}{K^e + K^r} \tag{3.19}$$

which turns out to be equal to the criterion first considered in (3.9).

In order to test the performance of some of these gradient lattice algorithms, computer simulations of their operation were conducted using a number of synthetically generated test signals. The simulation software is described in Appendix B. The gradient lattice algorithms tested were all of the "normalised" type. They were: the harmonic mean algorithm (GRAD-HM) defined by (3.6), (3.7) (with $K_{n,t}$ used in these equations) and (3.10); the forward and backward algorithm (GRAD-F+B), using separate reflection coefficients for each stage based on estimating (3.17) and (3.18) as shown below; and the minimum algorithm (GRAD-MIN), mentioned above.

additional correlation updates (c.f. (3.10))

$$w^e_{n,t} = \lambda w^e_{n,t-1} + e^2_{n-1,t} \tag{3.20a}$$

$$w^r_{n,t} = \lambda w^r_{n,t-1} + r^2_{n-1,t-1} \tag{3.20b}$$

$$v'_{n,t} = \lambda v'_{n,t-1} - e_{n-1,t} r_{n-1,t-1} \tag{3.20c}$$

for GRAD-F+B

$$K^r_{n,t} = \frac{v'_{n,t}}{w^r_{n,t}} \tag{3.21a}$$

$$K^e_{n,t} = \frac{v'_{n,t}}{w^e_{n,t}} \tag{3.21b}$$

(and with $K^r_{n,t}$ in (3.6) and $K^e_{n,t}$ in (3.7))

for GRAD-MIN

$$K_{n,t} = \text{minimum magnitude}\{K^e_{n,t},\ K^r_{n,t}\} \tag{3.22}$$

As would be expected from the preceding discussion, the differences in performance between these algorithms were small after convergence for stationary signals. For non-stationary signals some variations were apparent. With respect to mean square error minimisation, the GRAD-F+B performed best and the GRAD-MIN worst. An example is shown in Figure 3.3, in which a transversal filter LMS algorithm plot is included for comparison (see Chapter 2). In this test an input signal was generated by filtering white Gaussian noise in a 2 pole filter whose transfer function changed abruptly after 50 samples. In this "parameter jump" test the a_1 coefficient (see (2.1) and Figure 2.1 changed from -0.5 to +0.5 at time = 50, while a_2 remained fixed at 0.95. Four stage adaptive filters were used, with $\lambda = 0.95$ for the lattices and $\mu = 0.05$ for the LMS algorithm. The plot shows final stage error power averaged in time over 5 samples and over 25 realisations of the signal. The transversal filter using the LMS algorithm had slower convergence and greater excess mean square error. The GRAD-F+B reflec-

tion coefficients occasionally exceeded one in magnitude, but this occurred only during startup and did not seem to cause any stability problems.

Other performance comparisons for these gradient algorithms based on different criteria are made in the next section and in later chapters. Section 4.3 discusses determination of the predictor coefficients from the reflection coefficients for gradient and exact least square lattices.

3.3 Lattice Convergence Models

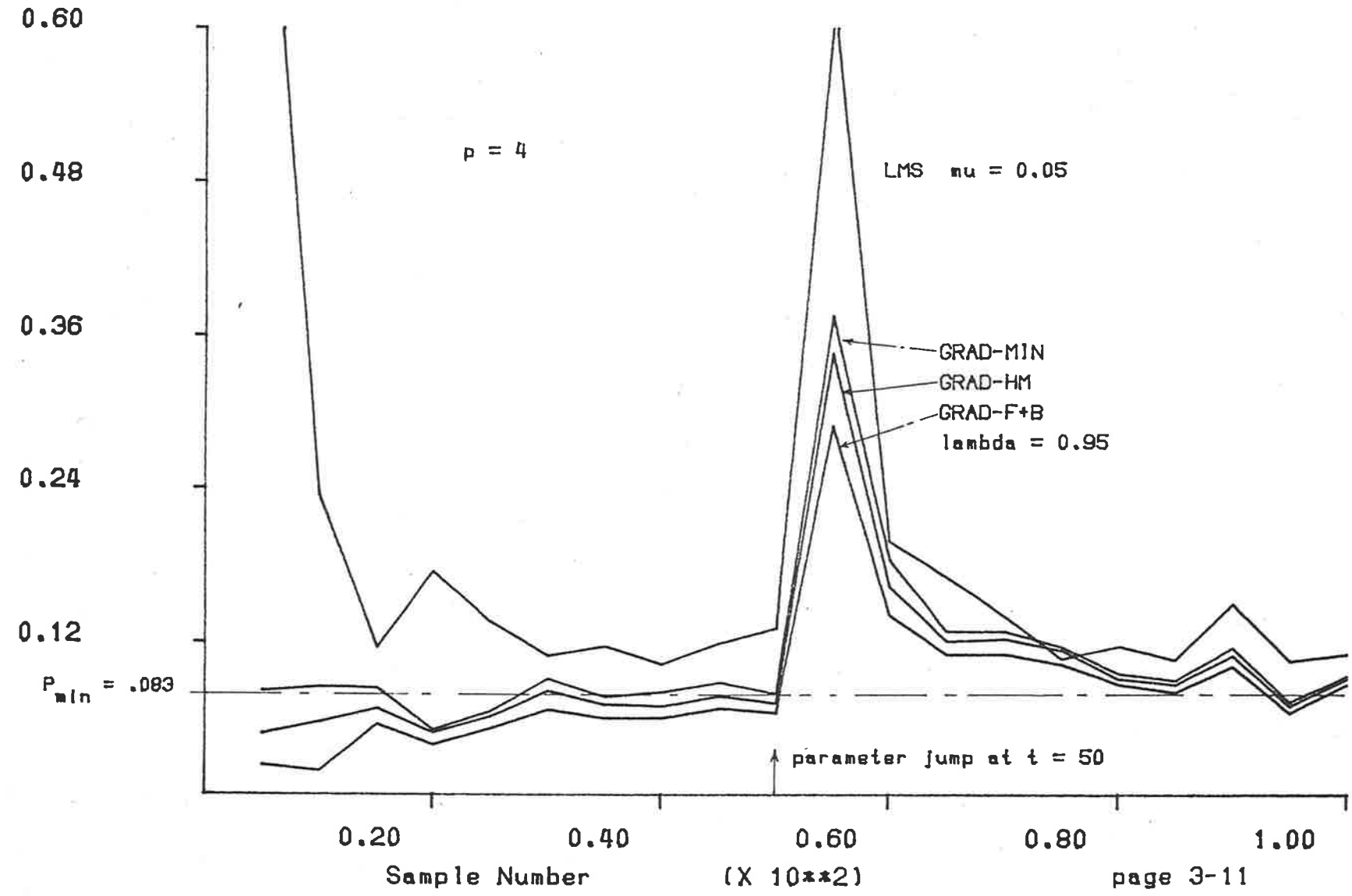
This section first examines existing studies of lattice convergence and shows the difficulties associated with considering multistage convergence. A new method of convergence analysis, based on exponentially weighted error minimisation as would occur in a block solution, is applied to the lattice, and its predictions compared to simulation results for several test signals.

Approximate analytical solutions of single stage lattice convergence with stationary input signals are possible and several gradient algorithms have been studied [36,7]. By applying a similar technique to that used in [36] to the harmonic mean algorithm of the previous section, an instructive result for the first stage reflection coefficient convergence is obtained. After taking expected values of (3.11) the right hand side expectation may be simplified if certain assumptions are made. Although the numerator and denominator are both random variables, if λ is close to 1 and t large, the "noise" on each term becomes small compared to its mean value; furthermore, assuming stationary inputs $e_{n-1,t}$ and $r_{n-1,t-1}$ we have

$$E\{K_{n,t}\} \approx \frac{-2E\{e_{n-1,t}r_{n-1,t-1}\}(\sum_{j=1}^{t}\lambda^{t-j}) + \lambda^t v_{n,0}}{E\{e_{n-1,t}^2 + r_{n-1,t-1}^2\}(\sum_{j=1}^{t}\lambda^{t-j}) + \lambda^t w_{n,0}} \tag{3.23}$$

The mean track of the reflections coefficient is given by the ratio of two exponential decays. Defining the "time constant" τ as the number of samples to move from

Figure 3.3 Adaptive filter MSE test



its initial value, $K_{n,0}$, to $(1-e^{-1})$ of the distance towards its final value $K_{n,\infty}$, i.e.

$$K_{n,0} = \frac{v_{n,0}}{w_{n,0}} \tag{3.24a}$$

$$K_{n,\infty} = \frac{-2E\{e_{n-1,t}r_{n-1,t-1}\}}{E\{e^2_{n-1,t} + r^2_{n-1,t-1}\}} \tag{3.24b}$$

$$K_{n,\tau} = K_{n,0} + (1-e^{-1})(K_{n,\infty} - K_{n,0}) \tag{3.24c}$$

leads to the result

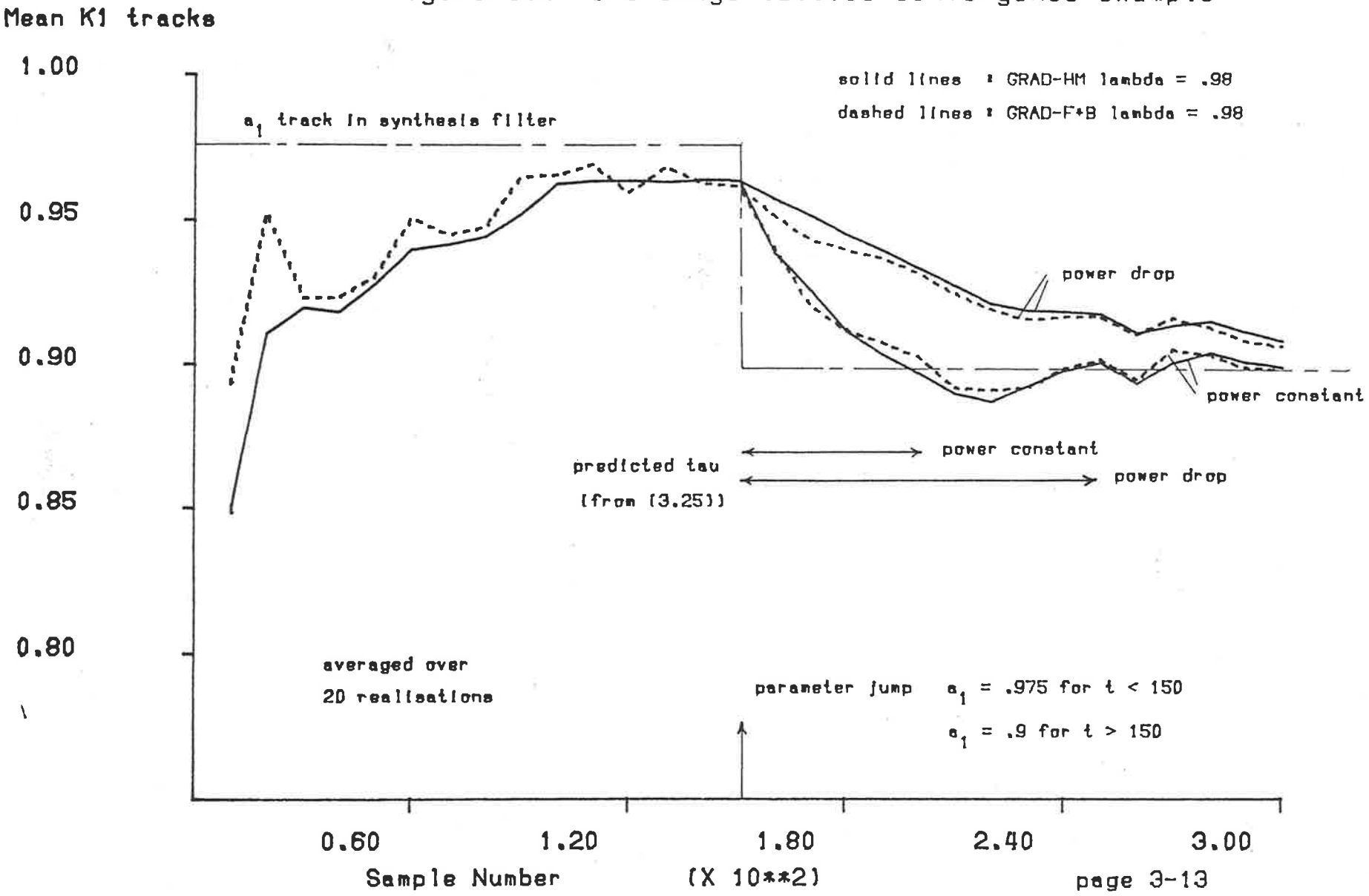
$$\tau \approx \frac{1}{1-\lambda} \ln \left[(e-1) \frac{(1-\lambda)w_{n,0}}{E\{e^2_{n-1,t} + r^2_{n-1,t-1}\}} + 1 \right] \tag{3.25}$$

If the initial value $w_{n,0}$ is zero the predicted time constant is zero. This obviously can not happen in practice, and is due to a breakdown in the assumptions leading to (3.23), however it does indicate that the single stage startup convergence from zero initial conditions will be fast.

Now consider the case if the lattice stage had converged by $t < 0$, and at $t = 0$ the crosscorrelation $E\{e_{n-1,t}r_{n-1,t-1}\}$ changed but the error power remained constant. From (3.25) and (3.12) the time constant would be N_{eff}. Thus for this situation the convergence time is easily predicted, however, with non-constant error powers, the convergence may be severely effected. This is illustrated in Figure 3.4. The mean $K_{1,t}$ track is shown for two signals, both generated from a one pole filter using the "parameter jump" method again. At $t = 150$ the a_1 coefficients in both cases changed from 0.975 to .9; normally this would reduce the power of the output signal but in the second case the signal was multiplied by a gain factor to keep its power constant. The effect on K_1 convergence is clearly seen and agrees reasonably with the predicted τ from (3.25).

We may note in passing that if the convergence derivation used above is applied to the GRAD-F+B algorithm, the result is essentially the same. GRAD-F+B could be expected to show similar performance to GRAD-HM when the assumptions used above are valid i.e. λ close to 1, large t. However, during startup these arguments can not be applied; if expressions for K_1^r and K_1^e are written out for the first few samples, it

Figure 3.4 One stage lattice convergence example



is seen that they can be quite different. Thus the differences between GRAD-HM and GRAD-F+B should be most noticeable at startup or low λ . This is substantiated by Figure 3.4.

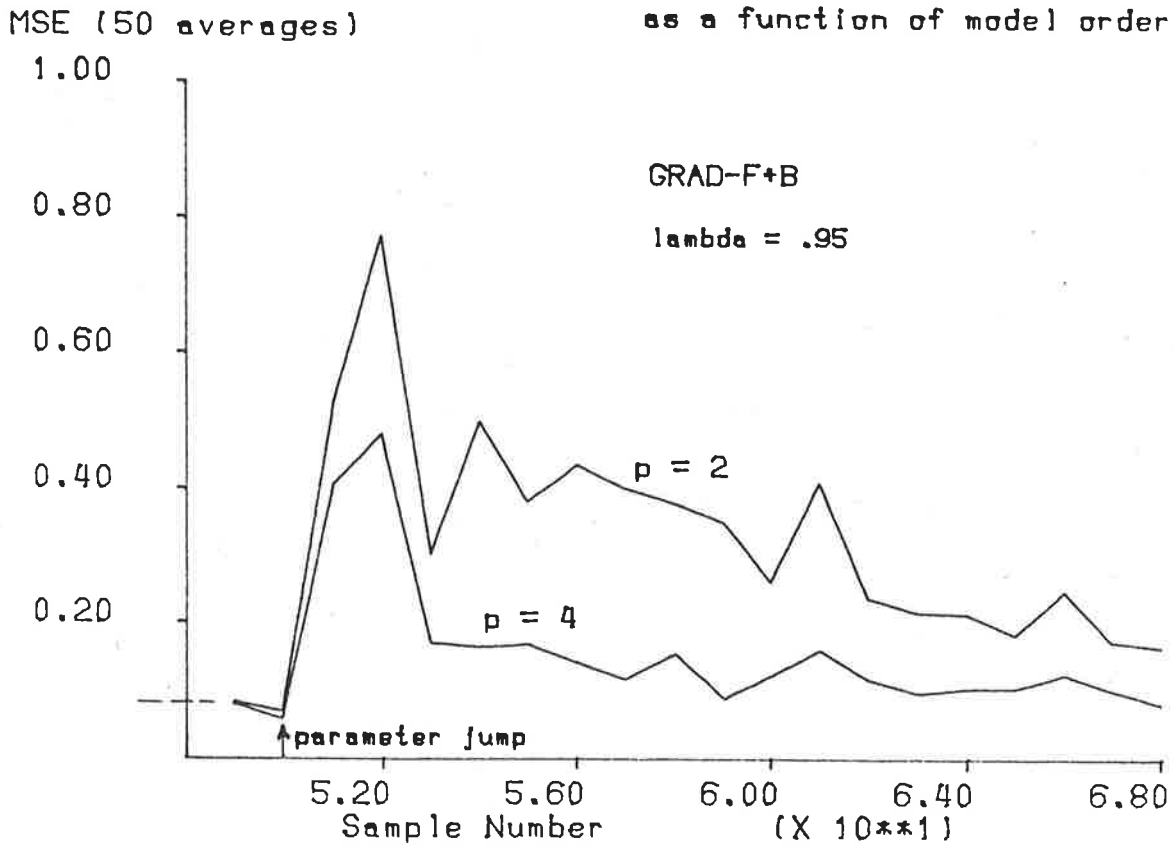
Multistage Convergence

The previous analysis of single stage lattice convergence cannot be easily extended to the multistage lattice due to the non-stationarity of the backward and forward error signals into each stage. During convergence all stages adjust simultaneously and independently without waiting for preceding stages outputs to stabilize. Several stages might adapt initially to the dominant spectral component of a signal, then as the cross-correlations between backward and forward errors change, so high order stages will readjust their reflection coefficients. The reflection coefficient tracks may therefore exhibit interesting changes of directions and reversals during convergence. The action of successive stages reaching their optimum states has been called "decoupling" in [7].

The output error power variations during convergence of a multistage lattice appear to be even more difficult to analyse than reflection coefficient tracks. Since all stages of the lattice will influence the final stage error and will attempt to minimise their outputs in the best way at any particular time, the MSE time constant would be expected to be shorter than that of individual reflection coefficients. However, each stage is constrained by its "memory" of past inputs (as determined by v_n and w_n for GRAD-HM) so there is much scope for variation. For example, the MSE plots of Figure 3.3 were produced for a 2 pole signal into 4 stage lattice filters. i.e. the last two stages would normally be unnecessary. However, after the step change in statistics of the input signal, the last two stages, with effectively zero initial conditions, responded very fast. Figure 3.5 shows an expanded plot of the MSE response for 4 stage compared to 2 stage lattice (GRAD-F+B) for the same signal: the former removed most of the excess rise in only 3 samples, although $N_{eff} = 20$.

Honig and Messerschmitt [36,37] have developed a recursive method of multistage

Figure 3.5 MSE during input transient as a function of model order



lattice convergence analysis which, while not producing analytical solutions, does allow prediction of mean reflection coefficients and error power as a function of time. They assume in part, that the "noise" on preceding reflection coefficients has no effect on the mean track of $K_{n,t}$. Their method involves calculating the error cross-correlation and power for each stage as a function of time, from which the evolution of the next reflection coefficient may be calculated. Then, by considering a set of time varying forward and backward predictors, the output error statistics may be deduced.

A new method is now presented which appears to achieve comparable performance, while being considerably simpler and less computationally expensive. The evolution of adaptive lattice coefficients is difficult to analyse because of the inherent adaptivity, and hence nonstationarity, of the lattice. However the optimum lattices (Exact Least Square forms, see next chapter) give solutions equal to those of the block methods

if they operate over the same data and use the same error weighting function. Gradient lattices achieve a somewhat suboptimum solution, as already noted. Assuming optimum performance, at any time the lattice coefficients may be predicted by considering the appropriate block solution. It will be seen that this may also be achieved if the data is non-stationary. This process may be repeated to obtain solutions at times of interest.

Assume we wish to find the coefficients $a_1, \ldots a_p$ at time t for a block solution, to minimise

$$e(t) = \sum_{k=0}^{t} e_{p,k}^2 \lambda^{t-k} \tag{3.26a}$$

using (3.3) and rearranging leads to

$$e(t) = \sum_{i=0}^{p} \sum_{j=0}^{p} a_i^p a_j^p \left[\sum_{k=0}^{t} y_{k-i} y_{k-j} \lambda^{t-k} \right] \tag{3.26b}$$

and taking expected values

$$\text{minimise} \quad E\{e(t)\} = \sum_{i=0}^{p} \sum_{j=0}^{p} a_i^p a_j^p R_t^c(i,j) \tag{3.27a}$$

where

$$R_t^c(i,j) = \sum_{k=0}^{t} E\{y_{k-i} y_{k-j}\} \lambda^{t-k}, \quad 0 \le i,j \le p \tag{3.27b}$$

If an expression can be found for the "composite" autocorrelation $R_t^c(i,j)$, which is a function at any time t of the (non-stationary) autocorrelation $R(k-i, k-j) = E\{y_{k-i} y_{k-j}\}$, then (3.27a) can be evaluated using any of the standard block methods. When this expression is, to a good approximation, a function of $|i-j|$ then (3.27a) can be solved efficiently by the Levinson recursion, (2.6). Two such cases are now considered.

Case 1

Assume that the statistics of the input signal change suddenly at time s, $0 < s < t$ and the autocorrelation is known before and after s. Then

$$R_t^c(i,j) = \sum_{k=0}^{s} r'_{i-j} \lambda^{t-k} + \sum_{k=s+1}^{s+p} E\{y_{k-i} y_{k-j}\} \lambda^{t-k} + \sum_{k=s+p+1}^{t} r''_{i-j} \lambda^{t-k} \tag{3.28}$$

where

$$r'_{i-j} = E\{y_{k-i}y_{k-j}\} \quad k \le s, \quad 0 \le i,j \le p \tag{3.29a}$$

$$r''_{i-j} = E\{y_{k-i}y_{k-j}\} \quad k > s + \max[i,j], \quad 0 \le i,j \le p \tag{3.29b}$$

r' and r'' are the stationary autocorrelation functions for y_t before and after time s, and both are functions only of lag $|i-j|$. The middle term in (3.27) occurs in the "overlap" region where the statistics of y_t are changing. Assuming that the samples on either side of the step change are uncorrelated, the overlap contribution may be approximated as follows. Those terms straddling the step change have no contribution, i.e. $E\{y_m y_n\} = 0$ when $m > s$, $n \le s$ or $m \le s$, $n > s$. This leads to the lower order lags from the overlap term having more effect as they straddle the step change less frequently, e.g. considering the overlap term contributions in a second order case,

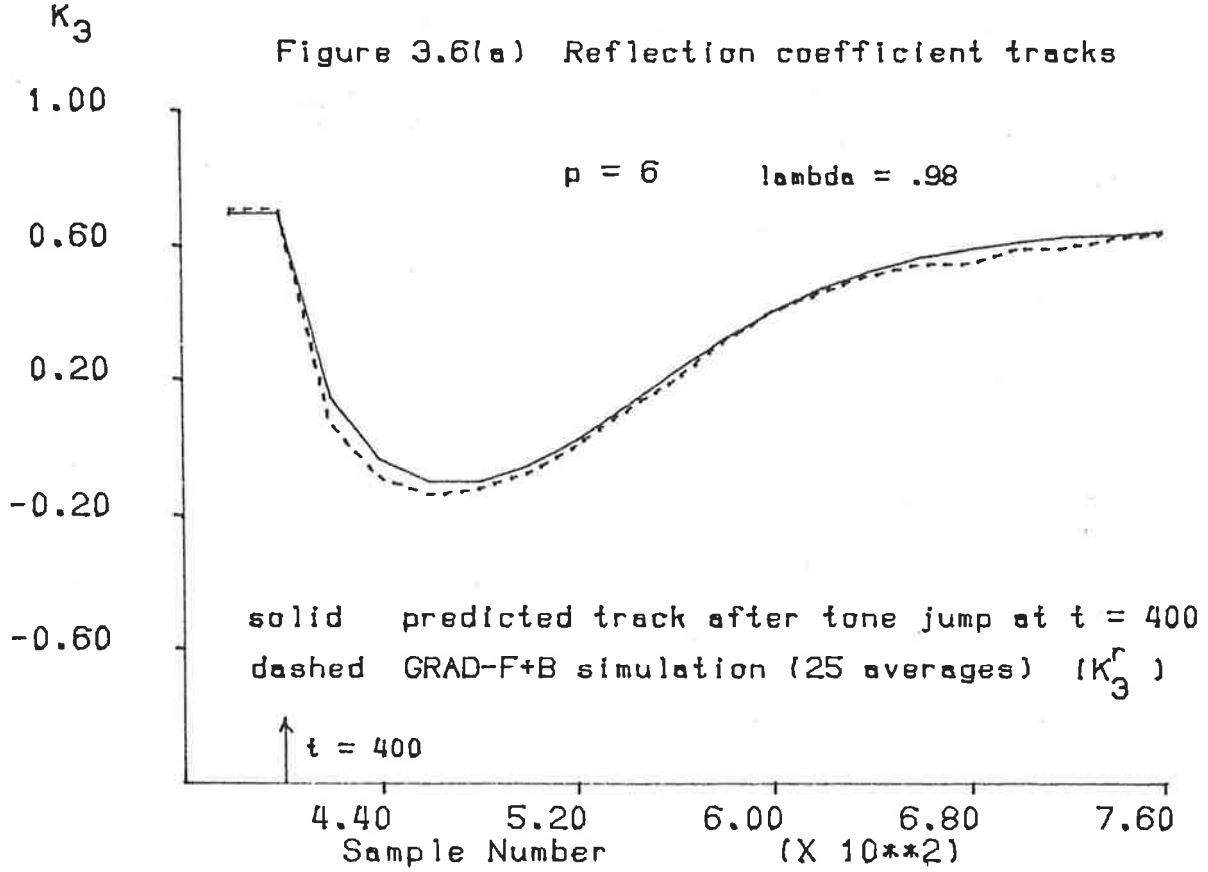
$$\sum_{k=s+1}^{s+2} E\{y_{k-i}y_{k-j}\} = \begin{array}{c} \\ i=0 \\ i=1 \\ i=2 \end{array} \begin{array}{c} \begin{array}{ccc} j=0 & j=1 & j=2 \end{array} \\ \begin{pmatrix} 2r''_0 & r''_1 & 0 \\ r''_1 & r''_0 + r'_0 & r'_1 \\ 0 & r'_1 & 2r'_0 \end{pmatrix} \end{array}$$

A triangular weighting as a function of lag $|i-j|$ may be observed in this example, plus the non-Toeplitz structure of the overlap autocorrelation contribution. However, the overlap contribution may be approximated by a Toeplitz autocorrelation if we assume

$$\sum_{k=s+1}^{s+p} E\{y_{k-i}y_{k-j}\} \approx \left(\frac{r'_{i-j} + r''_{i-j}}{2}\right)(p - |i-j|) \tag{3.30}$$

Furthermore, assuming that the exponential weighting in the overlap term is approximately constant, (3.28) becomes

$$R_t^c(i,j) = \lambda^{t-s}\left(\frac{1-\lambda^{s+1}}{1-\lambda}\right) r'_{i-j} + \frac{r'_{i-j} + r''_{i-j}}{2}(p - |i-j|)\lambda^{t-s-\frac{p}{2}} + \left(\frac{1-\lambda^{t-s}}{1-\lambda}\right) r''_{i-j} \tag{3.31}$$

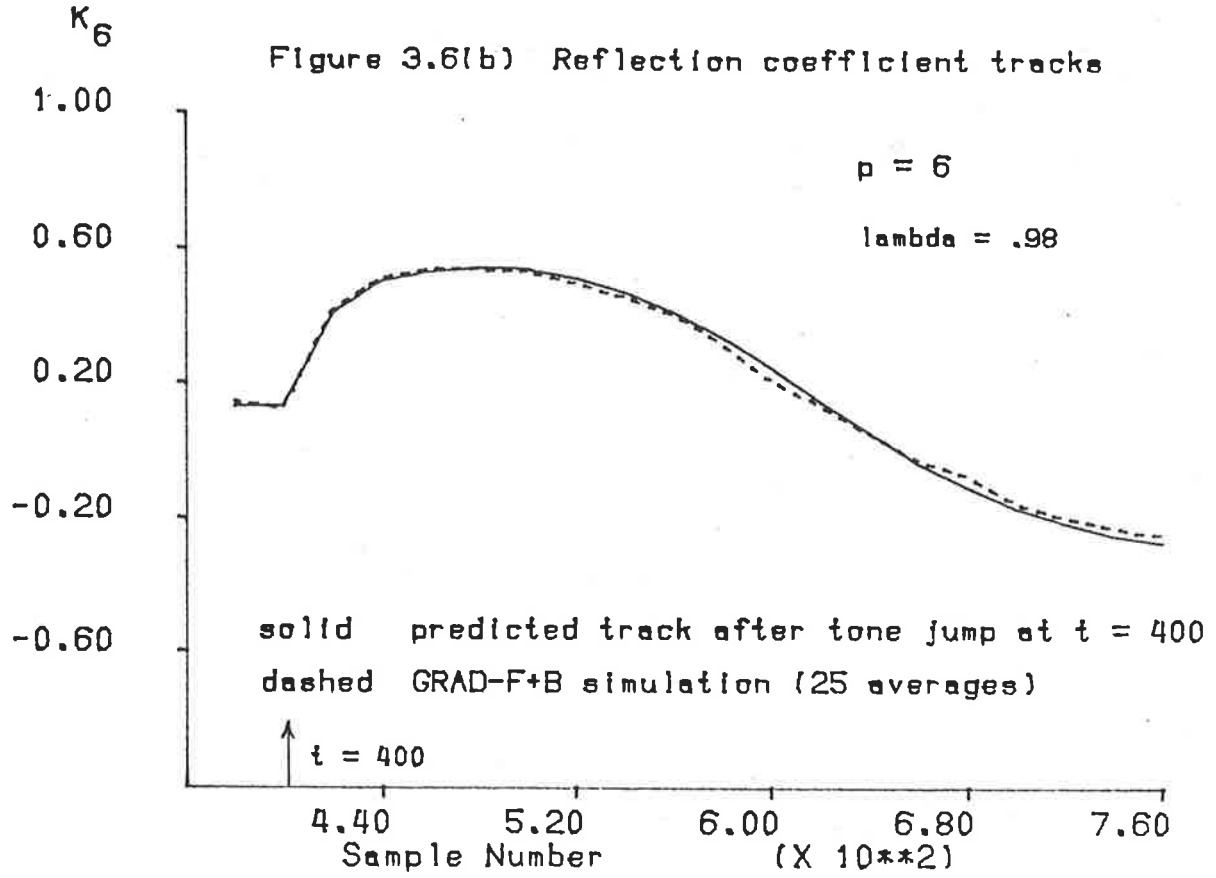


(3.31) is a function of lag only and so allows the Levinson recursion to solve (3.27a) efficiently. Linear prediction coefficients and reflection coefficients may therefore be predicted as a function of time, knowing the autocorrelation functions r' and r''. Although the method uses a number of assumptions, most of these involve the overlap term which will be relatively small if p is small compared to N_{eff}.

Good agreement between simulations and predicted results have been found for a range of signal types. An example is shown in Figure 3.6. In this case, a sinusoid was added to white noise and at sample 400 its frequency and amplitude were suddenly changed (including random phase jump). The autocorrelations r' and r'' were calculated by

$$r_i = \sigma_e^2 \delta(i) + \frac{a^2}{2} \cos(2\pi i f_n) \quad 0 \leq i \leq p \tag{3.32}$$

where σ_e^2 is the white noise power, a the sinusoid amplitude and f_n is the sinusoid normalised frequency ($0 \leq f_n \leq 0.5$). The mean reflection coefficient tracks K_3 and



K_6 from a simulation using 25 realisations of the signal are shown, plus the predicted tracks using the method above. See section 4.4 for details of exact least square lattice performance for this test.

Case 2

An alternative class of signals may be considered if we assume that the (non-stationary) autocorrelation function is a continuous function of time which may be approximated by an m-th degree polynomial for each lag

$$E\{y_{k-i}y_{k-j}\} = c_{0,|i-j|} + c_{1,|i-j|}k + \cdots c_{m,|i-j|}k^m \tag{3.33}$$

This method obviously requires the autocorrelation to be a relatively "smooth" function

of time; an example is considered shortly. When (3.33) is substituted into (3.27):

$$R_t^c(i,j) = \sum_{n=0}^{m} c_{n,|i-j|} S_{n,t} \tag{3.34}$$

$$\text{where } S_{n,t} = \sum_{k=0}^{t} k^n \lambda^{t-k} \tag{3.35}$$

A recursive expression for $S_{n,t}$ was derived by an extension of the usual method of summing geometric series. The result is

$$S_{n,t} = \frac{t^n - \lambda^t - \sum_{i=1}^{n-1} \binom{n}{i} (S_{n-i,t} - t^{n-i}) - S_{0,t-1} + 1}{1-\lambda} \tag{3.36}$$

$$\text{where} \qquad S_{0,t} = \frac{1-\lambda^{t+1}}{1-\lambda}$$

Using these results and knowing the polynomial coefficients $c_{n,i}$, the "composite" autocorrelation $R_t^c(i-j)$ may be determined at different times and the Levinson recursion employed to produce parameter estimates as the lattice tracks a non-stationary signal.

An example of the case 2 situation is the study of lattice response to a "random chirp" signal. A two pole filter was used to filter white Gaussian noise as previously described, except that in this case the a_1 coefficient was made a linear function of time, and a_2 remained constant.

$$a_1(t) = b + ct \tag{3.38}$$

The values of constants b and c were chosen so that the power of the chirp remained approximately constant over the analysis interval, for reasons explained later. To determine the autocorrelation function of the signal, consider the generating filter of Figure 2.1.

$$e_t = y_t + a_1 y_{t-1} + a_2 y_{t-2} \tag{3.39}$$

Now, e_t is uncorrelated with y_{t-1} and y_{t-2}, so multiplying by y_{t-1}, taking expected values and assuming, as before, the autocorrelation function is a smooth, slowly changing function of time, leads to

$$\frac{r_1(t)}{r_0} = \frac{-a_1(t)}{1+a_2} \tag{3.40}$$

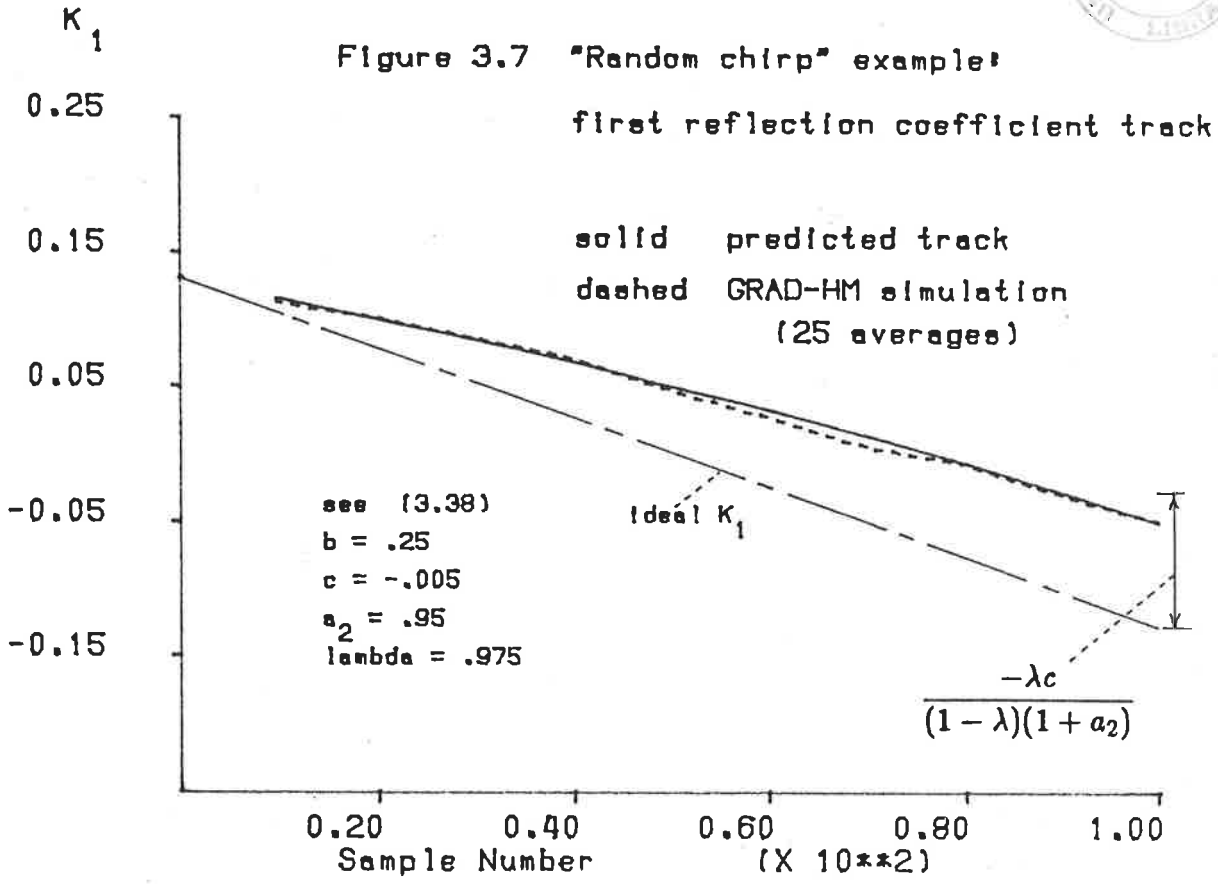


Figure 3.7 "Random chirp" example: first reflection coefficient track

so

$$r_1(t) = \frac{-br_0}{1+a_2} + \frac{-cr_0}{1+a_2}t \tag{3.41}$$

The polynomial coefficients $c_{0,0}$, $c_{0,1}$, and $c_{1,1}$ of (3.33) are thus identified and for this simple case the $K_{1,t}$ solution to (3.26) is easily found to be:

$$K_{1,t} \approx \frac{b - \frac{c\lambda}{1-\lambda}}{1+a_2} + \frac{tc}{(1+a_2)(1-\lambda^{t+1})} \tag{3.42}$$

(This same result was first derived by considering the behaviour of K_1 for the GRAD-HM gradient algorithm for the specified signal using (3.23), with constant power assumption for simplicity. With this direct approach, however, the analysis soon becomes difficult, even for K_2 in this example.) In more complex examples this simplification is not possible, but prediction of LPCs or reflection coefficients is possible via a computer program evaluation of (3.34) - (3.37) followed by efficient solution of (3.27a). These computations would be made for each time the predictions are required.

Figure 3.7 shows the simulated and predicted mean K_1 tracks for the random chirp signal. The "ideal" $K_{1,t}$, using only the current autocorrelations $-r_1(t)/r_0$ from (3.40), and not a composite of past estimates, is also shown. After startup transients the lag between this ideal $K_{1,t}$ and the mean lattice's $K_{1,t}$ tends to a constant, as shown by comparing the first terms of (3.41) and (3.42).

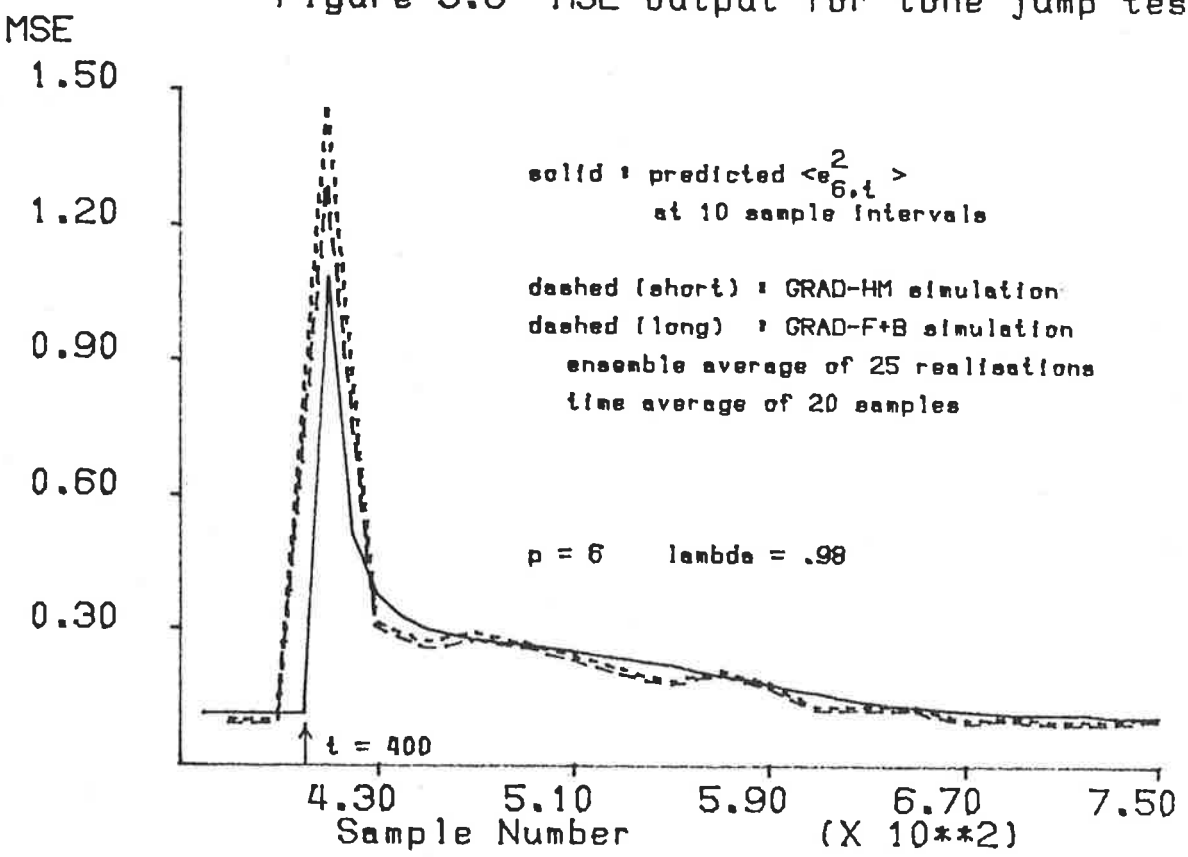
For both case 1 and case 2, the MSE output may also be predicted as a function of time. Although the Levinson recursion is used to solve (3.26), the error power σ_i^2 from (2.6e) can not be used in the present case, because the "composite" autocorrelation R_i^c is not equal to the signal autocorrelation at time t. Put another way, the lattice error, while achieving the specification of minimising the exponentially weighted sum, is not necessarily orthogonal to the data at any given time, so the usual error formulas can not be used. However, from (3.3):

$$E\{e_{p,t}^2\} = \sum_{i=0}^{p}\sum_{j=0}^{p} a_i^p a_j^p R_t(i,j) \tag{3.43}$$

Thus, at each prediction time, t, after the LPCs have been determined by solving (3.27a), the mean error power may be found from (3.43) using the current signal autocorrelation $R_t(i,j)$. Figure 3.8 shows the 6th stage mean forward error power for the sinusoid jump in noise example (Figure 3.6). The chief differences between the predicted and simulation results are due to the predicted values applying to discrete times (i.e. $t = 360, 370, \ldots 750$), but the simulation software again giving an ensemble and time average, the latter for final stage error squared from $t = 361$ to $t = 380$, $381 - 400, \ldots 741 - 760$.

This section has demonstrated that when the adaptive lattice gives close to optimum least square results, its mean coefficient response to the (well-behaved) transient and nonstationary signals considered here, may be readily predicted by using the appropriately weighted input signal statistics. Other characteristics of the coefficient response, such as variance, would be much harder to analyse under these nonstationary

Figure 3.8 MSE output for tone jump test



conditions. The convergence model presented above is implemented, for the two cases considered, in program TRACKS (see Appendix B). Chapter 8 summarises some of the results from this chapter.

Chapter 4 Exact Least Square Lattices

This chapter examines time recursive, optimum least square error solutions to the all-pole modelling problem for time series. In particular, the recently derived "Exact" Least Square (ELS) lattice algorithms are considered, and compared to the gradient lattices analysed in Chapter 3, from several points of view.

4.1 Recursive Least Square Estimation Methods

The field of linear least squares estimation is very broad; some indication of this is that although the technique was first used by Gauss to predict the motions of heavenly bodies, important advances are still being made. In order to put the ELS lattices into perspective, this section surveys a subfield relevant to discrete-time recursive solutions for time series all-pole models. Reviews of recent advances in this and related areas are given in [43,59 appendix 2].

The aim, again, is to minimise the prediction error squared from an all-pole model of the time series. Whereas this was done in Chapter 3 by a stochastic approach (i.e. taking expected values and solving a set of linear equations involving second order statistics of the signal), the present technique minimises the square error directly using the given data. The one dimensional case is considered here, but may be extended to handle a vector of data at each sampling instant , instead of a scalar observation (e.g. see [10,6]). If data samples y_s to y_t are available, a suitable objective is :

$$\text{minimise} \qquad V_{s,t} = \sum_{k=s+p}^{t} e_k^2 \tag{4.1a}$$

$$\text{where} \qquad e_k = y_k + \sum_{i=1}^{p} a_i y_{k-i} \tag{4.1b}$$

This formulation makes no assumptions about the data outside the given region, and is

usually called the covariance method. In matrix notation (4.1) becomes

$$\begin{pmatrix} e_{s+p} \\ e_{s+p+1} \\ \vdots \\ e_t \end{pmatrix} = \begin{pmatrix} y_{s+p} \\ y_{s+p+1} \\ \vdots \\ y_t \end{pmatrix} + \begin{pmatrix} y_{s+p-1} & \cdots & y_s \\ y_{s+p} & \cdots & y_{s+1} \\ \vdots & \ddots & \vdots \\ y_{t-1} & \cdots & y_{t-p} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \tag{4.2a}$$

or

$$E_{s,t} = Y_{s+p,t} + Y_{p,s,t-1} A_{s,t} \tag{4.2b}$$

$$V_{s,t} = E_{s,t}^T E_{s,t} \tag{4.2c}$$

(using notation similar to [10] where the first of the 3 subscripts, if present, indicates the number of columns, and the last two the range of data on which the quantity depends.) Solving (4.1) by differentiation leads to a set of normal equations :

$$R_{p,s,t-1} A_{s,t} = -Y_{p,s,t-1}^T Y_{s+p,t} \tag{4.3a}$$

$$\text{where} \qquad R_{p,s,t-1} = Y_{p,s,t-1}^T Y_{p,s,t-1} \tag{4.3b}$$

$$\text{so} \qquad A_{s,t} = -R_{p,s,t-1}^{-1} Y_{p,s,t-1}^T Y_{s+p,t} \tag{4.3c}$$

where R^{-1} may be a generalised matrix inverse if $R_{p,s,t-1}$ is singular [10]. Obviously these equations parallel those of Chapter 2, and specifically (4.3a) reduces to (2.5a), using estimated autocorrelations in the latter, under the "autocorrelation" or pre and post windowed case. Here it is assumed the data is zero outside the interval $y_{s+p-1}, \cdots y_{t-p}$, and the error squared sum to be minimised is the same as above. In this situation the data matrix $Y_{p,s,t-1}$ has triangular regions of zeros in the upper right and lower left corners, resulting in the sample covariance matrix having a Toeplitz structure. The prewindowed case is also of interest; usually it is assumed that the data samples y_0 to y_t are available, and that $y_k = 0$ for $k < 0$ (thus $s = -p$). This, like the covariance (no windows) situation leads to non-Toeplitz covariance matrices, which nevertheless have special properties.

Block methods solve (4.3c) by exploiting the Toeplitz or allied properties of the sample covariance matrix, e.g. Levinson recursion (see Chapter 2), fast Cholesky

factorisation [61,60,12]. Our interest is with time recursive methods, which update the estimate of $A_{s,t}$, or some other parameterisation of the optimum predictor, as each new data point becomes available. The conventional way of doing this uses a matrix inversion result, which avoids recalculating the new inverse at each time step by using a sequence of multiplications and scalar divisions in $O(p^2)$ operations. (See Chapter 7 of [40], or [45] for a good description of this method and its extensions).

Two "fast" methods of recursive LS estimation have recently been derived by Morf and others, which use only $O(p)$ operations per time step [41,62]. Because of the similarity of the first of these methods to Kalman filtering [63,59], this class of algorithms has been called "fast Kalman" algorithms. They implement the recursive LS solution in a tapped delay line form, whereas those in the second group use the lattice structure (Exact Least Square (ELS), or simply Least Square (LS), lattice algorithms). An important difference between these two is that the former methods provide the solution at each time step for a given order predictor, whereas the latter solve for increasing predictor orders, from 1 to p at each time. The lattice methods therefore include update equations for both time *and* order. They appear to have some advantages over fast Kalman algorithms (e.g. normalised algorithm forms, solution for all predictor orders) and will now be considered in more detail.

4.2 Exact Least Square Lattice Algorithms

The ELS lattice algorithms were first derived by partitioning the covariance matrix (e.g. see [43 appendix B]). A linear algebra approach using projection operators was then suggested by Shensa [47], and has been useful for insight into how the algorithms operate and the introduction of normalised versions. The projection, or geometric, derivation of ELS algorithms is lengthy and may be found in [6,10,47,51]. An overview of the method follows. Substituting (4.3c) into (4.2b) gives

$$E_{s,t} = (I - Y_{p,s,t-1} R_{p,s,t-1}^{-1} Y_{p,s,t-1}^T) Y_{s+p,t} \tag{4.4a}$$

$$= (I - P_{p,s,t-1}) Y_{s+p,t} \tag{4.4b}$$

where the matrix $P_{p,s,t-1}$ is called a projection operator.

$$P_{p,s,t-1} = Y_{p,s,t-1}(Y^T_{p,s,t-1}Y_{p,s,t-1})^{-1}Y_{p,s,t-1} \tag{4.5}$$

If $Y_{s+p,t}$ is considered to be a vector of current data samples, then the columns of $Y_{p,s,t-1}$ are delayed (shifted) vectors of data, whose delays range from 1 to p sampling times. They span a vector subspace called the "subspace of past observations". The elements of the sample covariance matrix in (4.3b) correspond to various inner products between these delayed vectors. The function of $P_{p,s,t-1}$, which from (4.5) is composed only of delayed data vectors, is to project the current data vector $Y_{s+p,t}$ onto the subspace of past observations. The projection in (4.4b), $(I - P_{p,s,t-1})$, maps current data vectors onto a subspace orthogonal to that spanned by past data vectors. $E_{s,t}$, the forward error vector, therefore represents that component of $Y_{s+p,t}$ which can not be derived from linear combinations of past data vectors.

Time and order update expressions for the projection operators may be derived in the prewindowed [6] and covariance [10] cases. The prewindowed order updates, for example, result from splitting subspaces into non-intersecting components (orthogonal decomposition). Thus as the forward error vector is orthogonal to the subspace of past observations, the subspace spanned by current *and* delayed data vectors is equal to the direct sum of the subspace spanned by past observations and that spanned by the error vector. This gives an order update for the projection operator, which may be applied to the current data vector resulting in (after taking most recent components) :

$$e_{n,t} = e_{n-1,t} + \frac{\Delta_{n,t}}{R^r_{n-1,t-1}} r_{n-1,t-1} \tag{4.6}$$

where $\Delta_{n,t}$ is the partial autocorrelation between the current data vector projected onto the $(n-1)$th order subspace of past observations and the $n-$th delayed data vector, and $R^r_{n-1,t-1}$ is the scaled $(n-1)$th order backward error covariance at time $t-1$. Comparing this to (3.6) indicates part of the lattice structure, and identifies the quantity corresponding to the reflection coefficient $K_{n,t}$. A similar ELS relation exists for $r_{n,t}$.

Time update formulas give updates for the partial autocorrelations $\Delta_{n,t}$. It transpires that the time updates contain gain terms which represent the amount of new information in the current data sample (the "angle" between the subspace of past observations and that spanned by past *and* current data [6]). These gains are not present in the gradient algorithms, and account for the principal difference between the two (see section 4.3).

The time updates in the prewindowed case may be easily modified for the exponential weighting previously encountered. This allows adaptive response to time varying data, and achieves, (c.f. (4.1a))

$$\text{minimum} \qquad \sum_{k=0}^{t} e_k^2 \lambda^{t-k}$$

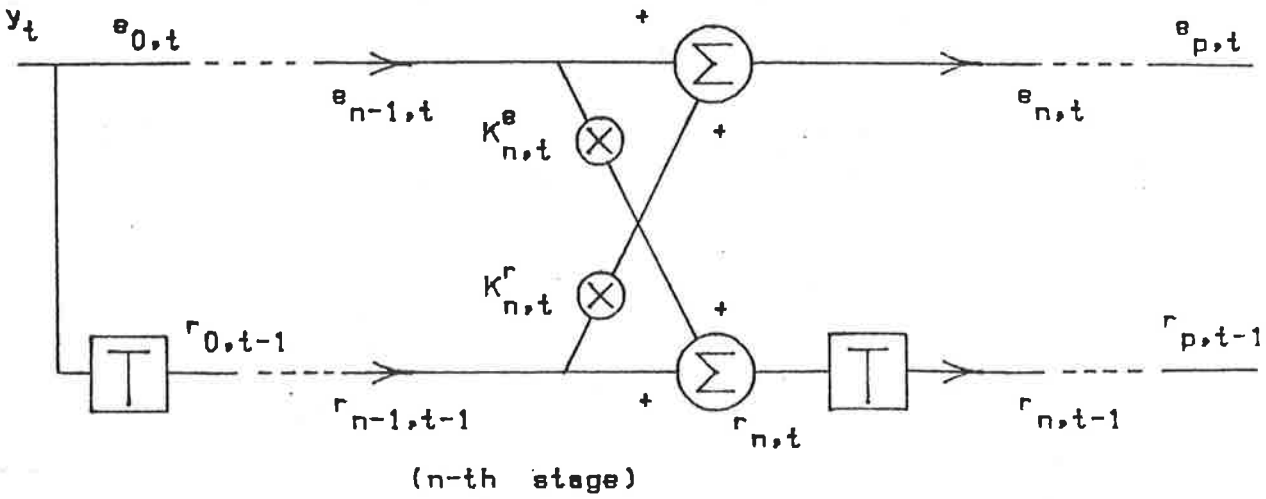
The λ weighting corresponds to a change in the inner product definition on the vector space [6,47], i.e. if X and Y are two $(t+1)$ dimensional vectors, with components x_i and y_i, then the inner product is

$$< XY >_t = \sum_{k=0}^{t} x_i y_i \lambda^{t-k}, \qquad 0 < \lambda \leq 1 \tag{4.7}$$

In the covariance case a rectangular weighting "sliding memory" scheme is more appropriate for removing the effect of old data. As each new data sample becomes available both t and s in (4.1a) are incremented by 1 and the optimum LS solution updated.

Normalised versions of the ELS lattice algorithms are available for the prewindowed and covariance cases. Surprisingly, the normalised forms are often simpler in terms of their number of operations than their unnormalised counterparts. The normalised prewindowed ELS algorithm uses only three update equations per time step (compared to about 3 times this many in the unnormalised form). This simplification is due to the normalisation being more than just magnitude adjustments; an "angle" normalisation is included which eliminates several variables altogether and leads to a remarkably simple algorithm [6]. The normalised algorithms are expected to have better

Figure 4.1 ELS unnormalised prewindowed lattice filter



performance for finite word length arithmetic implementations due to the error magnitudes remaining approximately constant from input to output. Two potential difficulties exist for the normalised forms. In some applications (e.g. adaptive prewhitening) the unnormalised error output is required, and secondly, square root operations are required in all update formulas. The latter aspect can be overcome with a class of bit recursive algorithms (CORDIC [64,65]), which enable the evaluation of certain transcendental functions, including square roots, as efficiently as multiplications. (Another advantage of the unnormalised ELS forms is the presence of the extra gain terms, which will be discussed shortly. These have been used directly in some applications, e.g. [16].)

The unnormalised prewindowed ELS algorithm is now given. It will be denoted ELS-UP. Its normalised version (ELS-NP) is given in Appendix C, plus the normalised covariance ELS algorithm (ELS-NC) and some examples of their performance. Other implementation details may be found in Appendix B. Figure 4.1 shows the lattice structure of ELS-UP, which also applies to GRAD-F+B.

Initialisation (for each new sample y_t)

$$e_{0,t} = r_{0,t} = y_t \tag{4.8a}$$

$$R^e_{0,t} = R^r_{0,t} = \lambda R^e_{0,t-1} + y_t^2 \tag{4.8b}$$

$$\gamma^c_{0,t} = 1 \tag{4.8c}$$

then for $n = 1$ to p

$$\Delta_{n,t} = \lambda \Delta_{n,t-1} - \frac{e_{n-1,t} r_{n-1,t-1}}{\gamma^c_{n-1,t-1}} \tag{4.9a}$$

$$K^r_{n,t} = \frac{\Delta_{n,t}}{R^r_{n-1,t-1}} \tag{4.9b}$$

$$K^e_{n,t} = \frac{\Delta_{n,t}}{R^e_{n-1,t}} \tag{4.9c}$$

$$e_{n,t} = e_{n-1,t} + K^r_{n,t} r_{n-1,t-1} \tag{4.9d}$$

$$r_{n,t} = r_{n-1,t-1} + K^e_{n,t} e_{n-1,t} \tag{4.9e}$$

$$R^e_{n,t} = R^e_{n-1,t} - K^r_{n,t} \Delta_{n,t} \tag{4.9f}$$

$$R^r_{n,t} = R^r_{n-1,t} - K^e_{n,t} \Delta_{n,t} \tag{4.9g}$$

$$\gamma^c_{n,t-1} = \gamma^c_{n-1,t-1} - \frac{r^2_{n-1,t-1}}{R^r_{n-1,t-1}} \tag{4.9h}$$

R^e and R^r are the forward and backward error covariances respectively. They are scaled, e.g. R^e_i is approximately $E\{e_i^2\}$ times $(1-\lambda)^{-1}$ during steady state operation, the latter factor being the effective number of data samples involved. The γ^c terms are the additional gain factors previously alluded to and can be shown to lie between 0 and 1. Morf and Lee have emphasised their connection to the log likelihood function [51]. They show that if the data samples y_t to y_{t-p} (assumed Gaussian) are representative of the sample covariance matrix R (which is implicit in the reflection coefficient values), then the joint probability distribution of y_t to y_{t-p} is maximised by the γ^c terms being close to unity. If this is not the case the small values of these gain terms cause large updates in (4.9a), and thus rapid adaption to the new data.

A simple non-probabilistic interpretation of these terms is also possible from

(4.9h). Under steady state, converged conditions the second term in (4.9h) will be approximately $1-\lambda$. The backward error covariance, from a time update perspective, contains the past history of many errors and will usually be slow to change. A sudden change in the input statistics however may cause backward errors to increase greatly within a few samples, thereby making γ^c terms much smaller and so rapidly updating the partial autocorrelations.

Finally note that the ELS-UP algorithm as displayed in (4.9) has been written in a form consistent with preceding chapters and most suitable for comparison with gradient algorithms. Some subscripts, time indices and signs differ from existing references [51,6,12], which also have differences between themselves.

4.3 Determination of the LPCs

In some adaptive lattice applications (e.g. all-pole spectral estimation) it is necessary to determine the optimum Linear Prediction Coefficients from the lattice. Typically this might be required at some submultiple of the sampling rate. For the standard gradient lattices the procedure is straightforward. To obtain the forward LPCs (i.e. the coefficients in (3.3) resulting in minimum MSE) the impulse response of the lattice, from input to forward error output, is required. (Recall that the final stage forward error $e_{p,t}$ from the lattice is the same as that from a $p-$th order forward PEF, therefore their transfer functions and impulse responses are the same.) If the impulse is applied at $t=0$, then the final stage error output at $t=1$ ($e_{p,1}$) will be a_1^p, similarly $e_{p,2}=a_2^p$ etc. The LPCs for any order may be obtained, of course, by using the appropriate number of lattice stages. The b_i^p, for $i=0,\ldots p$ may be similarly obtained from the backward error output impulse response (using the *current* last stage backward error i.e. $r_{p,t}$), and will equal the a_i^p in reverse order (see 3.4). This procedure is equivalent to using two equations of the Levinson recursion, (2.6c) and (2.6d), to obtain the LPCs for successively higher orders.

The two reflection coefficients per stage gradient lattice, GRAD-F+B, was pro-

duced by augmenting the standard lattice structure to minimise the backward and forward error powers separately, and in so doing the nonstationary response was improved (section 3.2). In this case the upper/ lower transfer function represents the best (estimated) whitening filter for the forward/ backward predictor. The a_i and b_i so obtained will not, in general, be equal, although they will asymptotically converge for stationary signals and $\lambda = 1$. This process for finding the predictor coefficients is described by a generalised version of (2.6c) and (2.6d) :

repeat for m = 1 to p

$$a_m^m = K_m^r \tag{4.10a}$$

$$b_0^m = K_m^e \tag{4.10b}$$

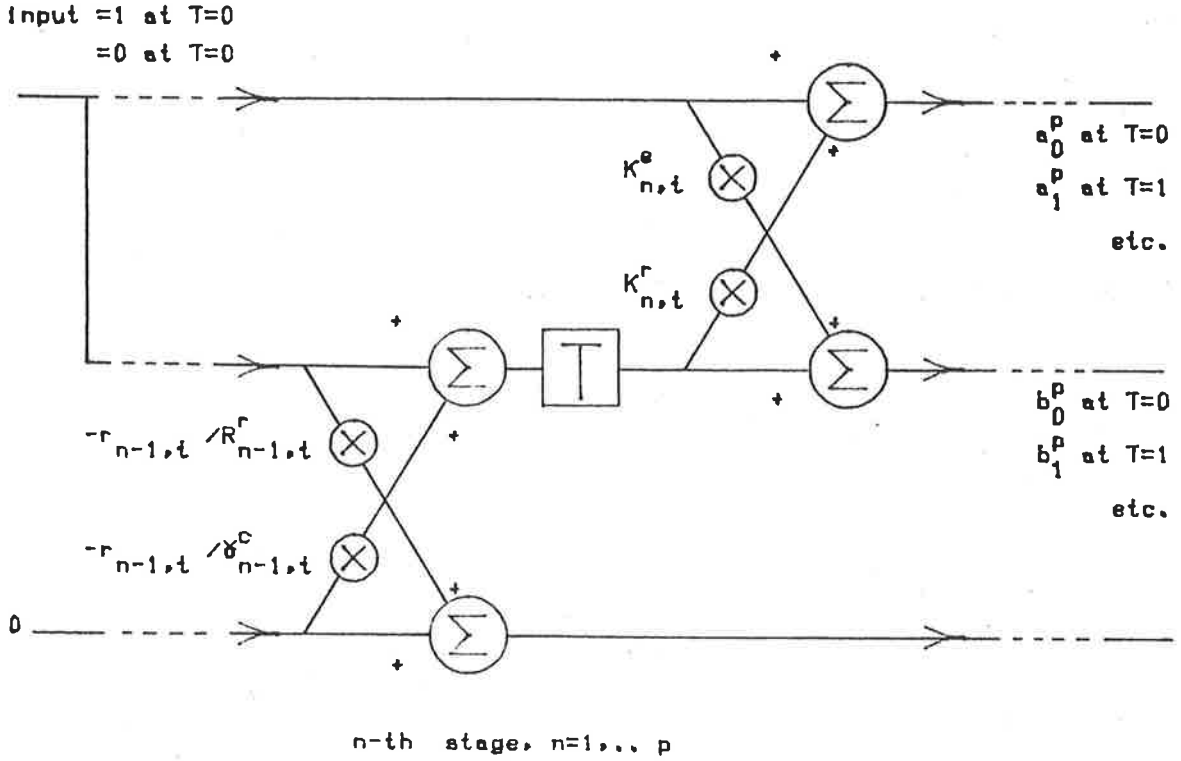
for $n = 1, \ldots m-1$

$$b_n^m = b_{n-1}^{m-1} + K_m^e a_n^{m-1} \tag{4.10c}$$

$$a_n^m = a_n^{m-1} + K_m^r b_{n-1}^{m-1} \tag{4.10d}$$

For the ELS lattices the solution for the optimum LPCs is obtained via the mathematics of the projection operator derivation (e.g. [12,10]) and is not as straightforward as the gradient cases. The result for ELS-UP lattice is shown in Figure 4.2. The impulse response from the top output gives the a_i^p, for $i = 0, \ldots p$, while that from the middle output gives the b_i^p, for $i = 0, \ldots p$. This filter may be viewed as an augmented lattice, with the extra "correction" terms (from the bottom line) a function, in part, of the backward errors in the lattice at the specified time. This filter has been called the "exact whitening filter" because it allows determination of the exact (optimum) predictor coefficients at time t. Assuming lattice startup at $t = 0$, as data passes through the lattice the correction terms in the exact whitening filter become progressively smaller (at least in the $\lambda = 1$ situation) as the error covariance increases. Asymptotically therefore these terms have no effect on the optimum LPCs. Other forms of the exact whitening

Figure 4.2 Exact whitening filter for ELS unnormalised prewindowed lattice (ELS-UP)



filters exist (e.g. see [9]), but that shown in Figure 4.2 is convenient as all quantities required may be extracted from the adaptive filter at one time.

Friedlander [12] has suggested an alternative method of finding the approximate LPCs for ELS lattices. This amounts to not using the bottom line correction terms in the whitening filter. As this so called "frozen" method would probably perform somewhere between ELS and gradient algorithms, with the degradation from ELS performance difficult to predict, it is not used in the examples which follow, although the facility is available in the simulation software.

4.4 Comparison of Gradient and ELS lattices

This section will first examine the gradient forward and backward (GRAD-F+B) and Exact Least Squares unnormalised prewindowed (ELS-UP) algorithms, and show

they are the same but for the γ^c terms. This agrees with the conclusions of Medaugh and Griffiths [49], where a different version of the gradient algorithm was used and additional assumptions were required.

Rewriting (3.20) and (3.21)

$$K^r_{n,t} = \frac{v'_{n,t}}{w^r_{n,t}} = \frac{\lambda v'_{n,t-1} - e_{n-1,t} r_{n-1,t-1}}{\lambda w^r_{n,t-1} + r^2_{n-1,t-1}} \tag{4.11}$$

this may be arranged to give

$$K^r_{n,t} = K^r_{n,t-1} \lambda \frac{w^r_{n,t-1}}{w^r_{n,t}} - \frac{e_{n-1,t} r_{n-1,t-1}}{w^r_{n,t}} \tag{4.12}$$

$$\text{so} \quad \overline{K}_{n,t} = \lambda \overline{K}_{n,t-1} - e_{n-1,t} r_{n-1,t-1} \tag{4.13a}$$

$$\text{where} \quad \overline{K}_{n,t} = K^r_{n,t} w^r_{n,t} \tag{4.13b}$$

By similar operations on the other gradient reflection coefficient, the same result (4.13a) is achieved with

$$\overline{K}_{n,t} = K^e_{n,t} w^e_{n,t} \tag{4.13c}$$

Rewriting the error updates:

$$e_{n,t} = e_{n-1,t} + K^r_{n,t} r_{n-1,t-1} \tag{4.13d}$$

$$r_{n,t} = r_{n-1,t-1} + K^e_{n,t} e_{n-1,t} \tag{4.13e}$$

That part of the gradient algorithm represented by (4.13a) to (4.13e) is therefore the same as the ELS counterpart, (4.9a) to (4.9e) if the ELS γ^c terms are unity. The variables which are not common correspond as follows:

variable	GRAD	ELS
partial autocorrelation	$\overline{K}_{n,t}$	$\Delta_{n,t}$
forward error covariance	$w^e_{n,t}$	$R^e_{n-1,t}$
backward error covariance	$w^r_{n,t}$	$R^r_{n-1,t-1}$

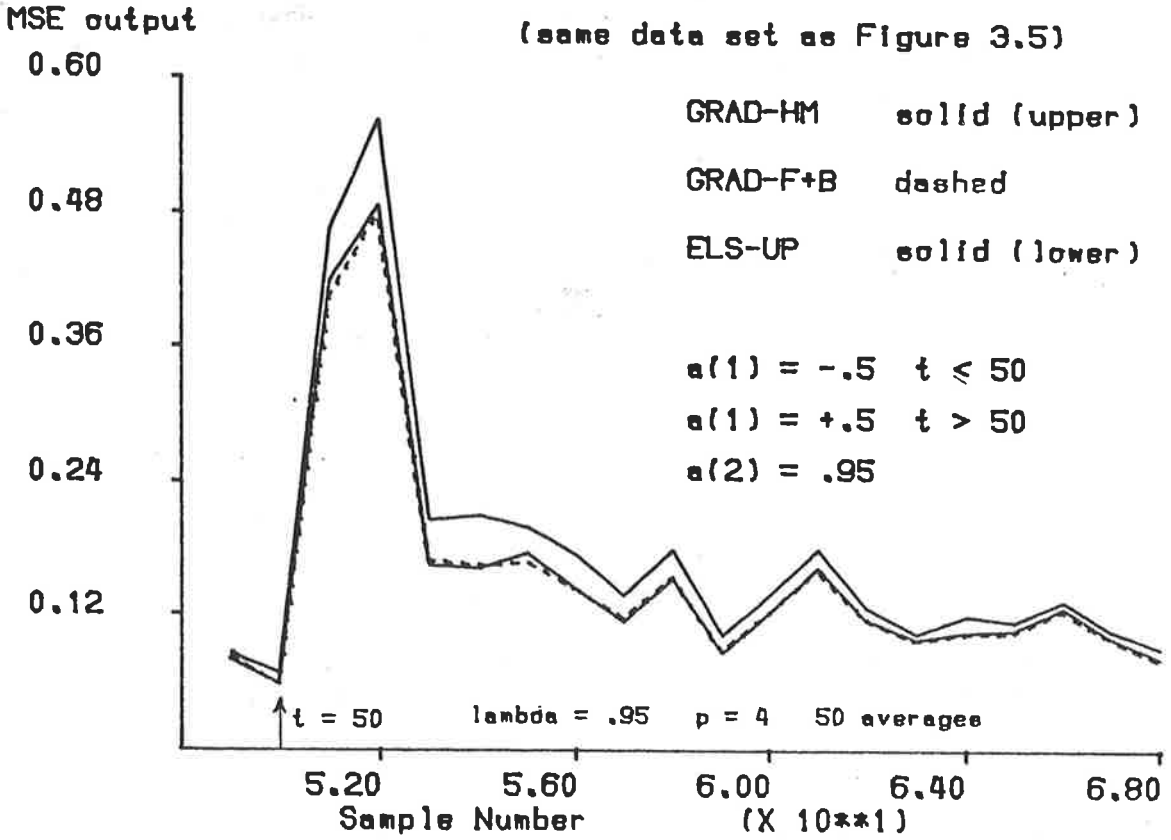
(The difference in subscripts here is due to slightly different definitions of the quantities concerned, arising from the separate derivations of the two algorithms.) Now considering the error covariance updates, the ELS equations are order updates ((4.9f) and (4.9g)), whereas the gradient solution uses time updates ((3.20a) and (3.20b)). Time updates do exist in the former case but are not usually employed as additional computations would be involved. When time updates for R^c and R^r are examined ([6], equations (54a) and (54b)) and translated via the above table, they are the same as the gradient updates in (3.20) (again assuming γ^c terms are unity).

Some performance and implementation comparisons between previously considered exponentially weighted adaptive lattice algorithms are now presented. Simulation results for mean square error output and estimated filter parameters are given for a range of input signals, and compared to predicted results from the model presented in Chapter 3 where appropriate. Other statistics of the parameter estimates, such as their variance, are also of interest and are considered with respect to all-pole spectral estimation in Chapter 5.

Consider first the tone jump test from Chapter 3, with results for some mean reflection coefficients plotted in Figures 3.6a and 3.6b. These plots were for the GRAD-F+B algorithm. The simulation results for both GRAD-F+B and ELS-UP algorithms, using the same conditions and realizations of the data, were so close for the mean 3rd reflection coefficient (K_3^r and K_6^r were recorded in fact) that they were indistinguishable from that shown in Figure 3.6a. For the mean 6th reflection coefficient the differences in simulation results could just be seen; at the turning point of $< K_6 >$ in Figure 3.6b the GRAD-F+B result was about 3% greater than ELS-UP (GRAD-HM was about 5% greater than ELS-UP). Figure 3.8 shows the mean error output for this test. Compared to GRAD-F+B the ELS-UP error was actually 2% higher! GRAD-HM was 14% larger than GRAD-F+B.

Figure 4.3 shows the MSE output comparison with finer resolution in time than

Figure 4.3 MSE comparison for "parameter jump" test
(same data set as Figure 3.5)

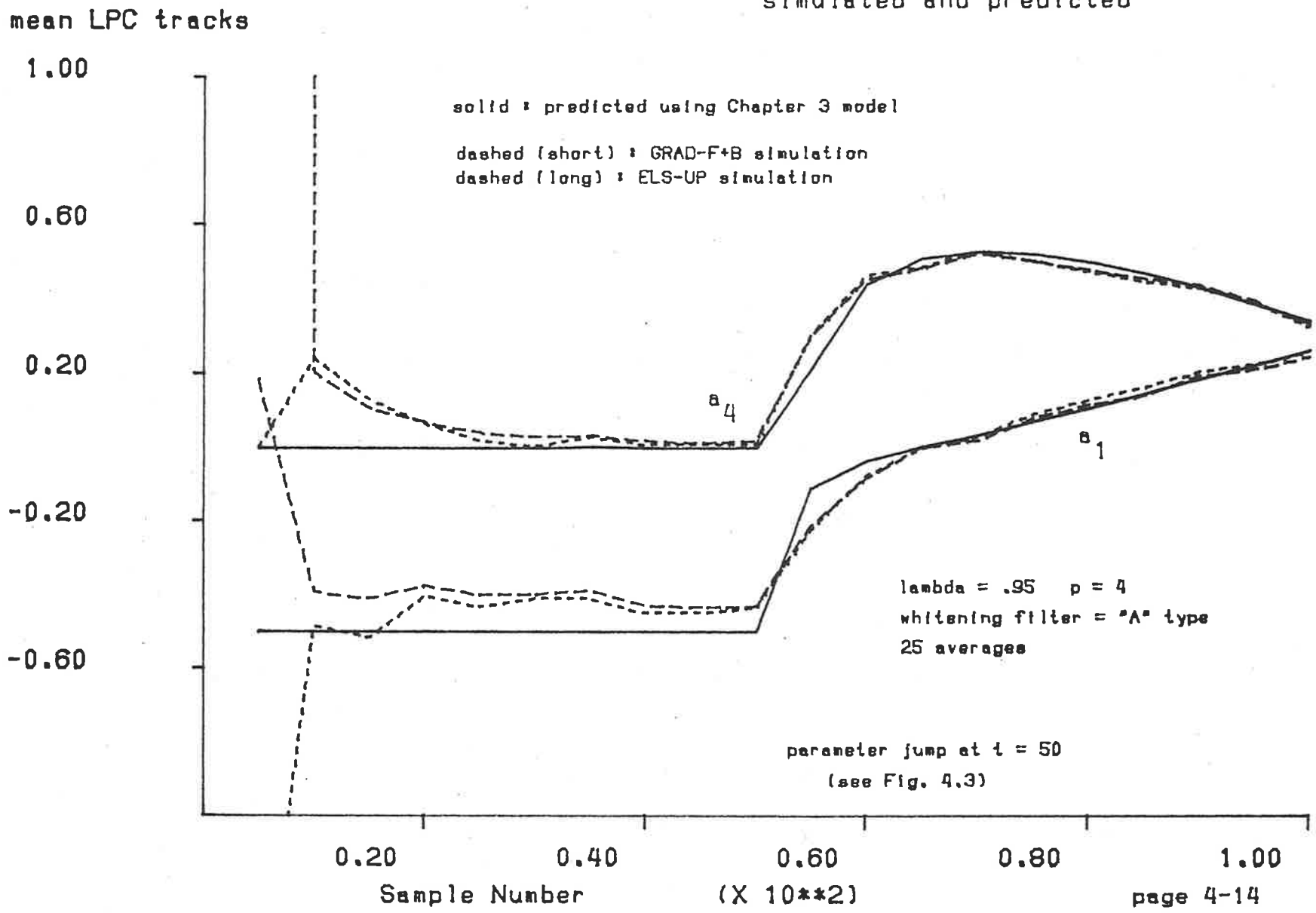


the preceding example. It used the same conditions as in Figure 3.5, namely a two pole signal with "parameter jump" at time $t = 50$ (see section 3.3). Four stage lattices were used in this test. Once again the ELS-UP and GRAD-F+B results are almost identical, with the latter being slightly superior. The GRAD-HM has a significantly higher forward error power transient following the step change.

The mean tracks of two LPCs are shown in Figure 4.4 for the same simulation conditions. The LPC estimates were derived using the whitening filters from section 4.3, at intervals of 5 samples. The predicted tracks, according to the model from Chapter 3, are also shown, and agree well with the simulated results. The GRAD-F+B and ELS-UP exhibit very similar behaviour, except within the first few samples, where they both give initially erratic estimates, as expected.

The examples presented so far have indicated little difference in MSE output or

Figure 4.4 Mean LPC tracks : simulated and predicted



parameter estimates for the ELS algorithm and its closest gradient counterpart. This coincides with the analysis of the previous section, considering that the γ^c terms will be almost unity. Some similar results have been reported by Medaugh and Griffiths [49,50] with a slightly different gradient algorithm, under these "well behaved" conditions. The marginally higher ELS-UP transient error was not expected however.

Significant differences in the above two algorithms are most notable when considering the initial adaptive response to deterministic signals such as from impulse response tests. If the impulse response (with no added noise) of an all-pole filter is presented to the adaptive lattices, the ELS algorithms converge in p samples to the coefficients used in the generating filter [12]. This follows from the ELS lattice providing the optimum least square solution, which in the noiseless case requires as many samples as the unknown coefficients. The gradient algorithms show marked differences in performance under these conditions, with GRAD-HM far from optimum due to its restriction of a single reflection coefficient per stage. An example is shown in Figures 4.5 and 4.6 for the impulse response of a 4 pole synthesis filter with coefficients:

a_1	-2.7607
a_2	3.8106
a_3	-2.6535
a_4	.9238

This signal, denoted in simulations as "ub4p", has a power spectrum of high dynamic range ($\approx$64 dB), with two closely spaced peaks (hence the "beat" effect in the time series), and is used in spectral analysis tests in the next chapter. (It is also used in [23].) Figure 4.5 shows the impulse response time series and the final stage forward errors ($t = 1, \ldots 100$) for a 4 stage gradient lattice. The GRAD-F+B error deviations from zero may just be detected on this scale (ELS-UP errors were zero to within machine accuracy).

Figure 4.6 shows the estimated $a_{2,t}$ at $t = 5,\ 10, \ldots 100$ for the same test. The GRAD-F+B a_i estimates did not converge to the values given previously because by the

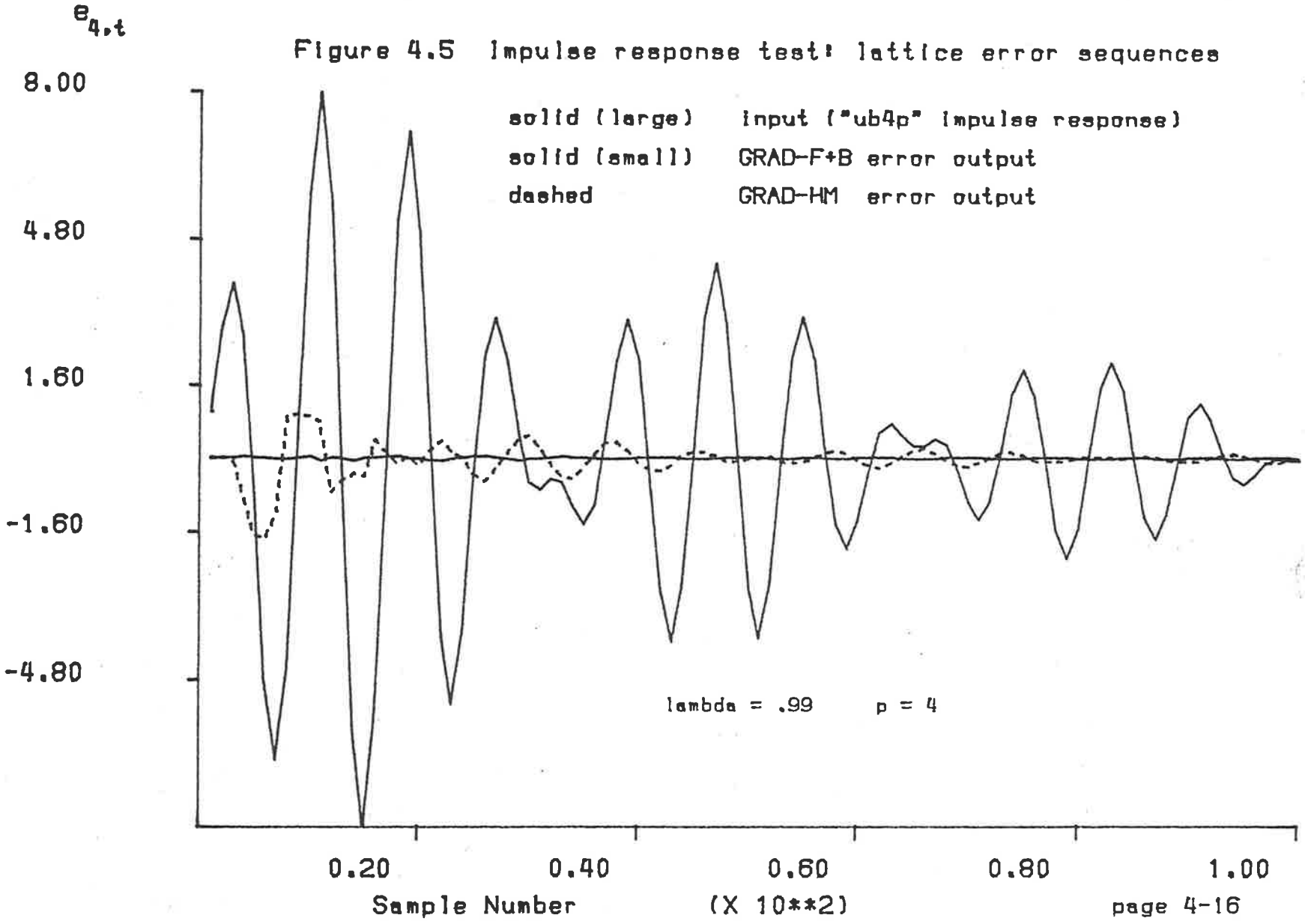


Figure 4.5 Impulse response test: lattice error sequences

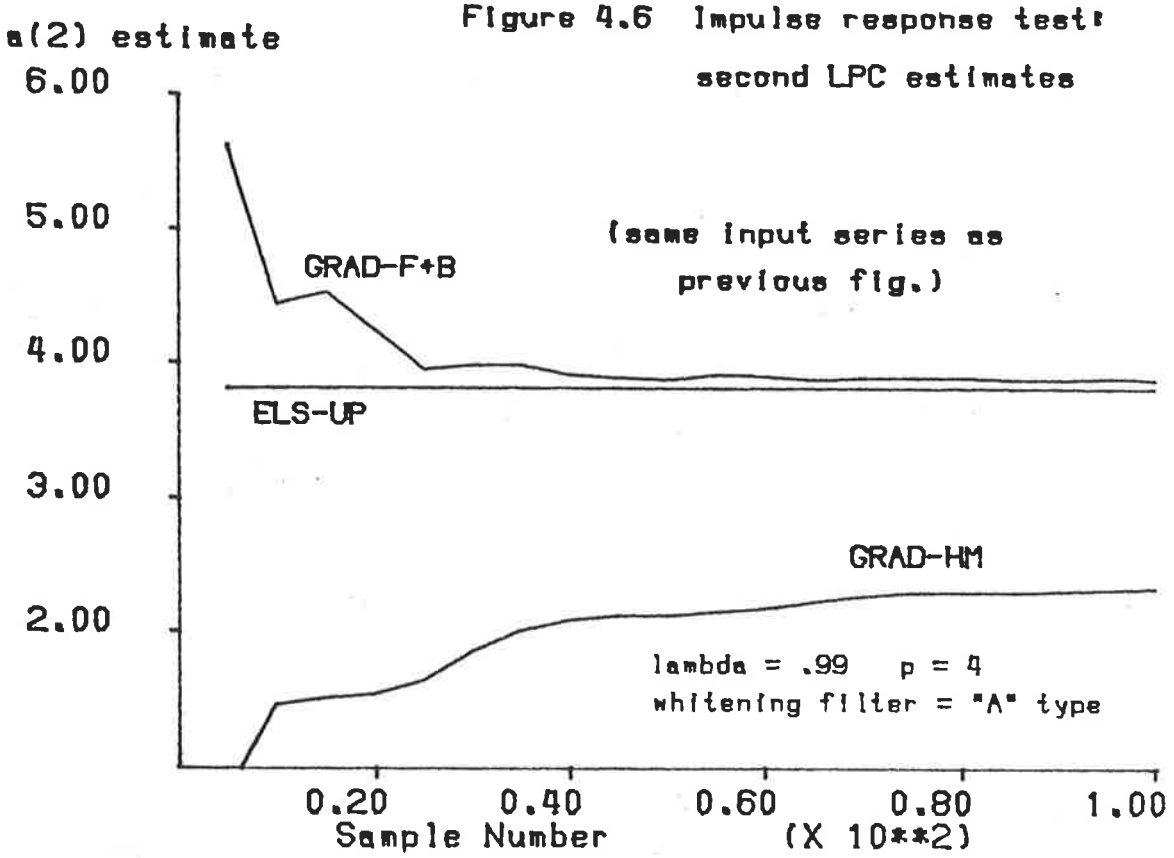


Figure 4.6 Impulse response test: second LPC estimates

time they were reasonably close the impulse response had diminished too far. (Indeed one of the reflection coefficients remained "stranded" just greater than unity but this would only have caused ill effects if the estimated coefficients had been used in a signal synthesis filter.) The corresponding all-pole spectral estimate resolved the peaks in about 15-20 samples. In contrast the GRAD-HM a_i estimates remained far from their ideal values, and the resulting spectral estimate never resolved the twin peaks.

In practical adaptive filter applications involving impulse-like inputs (e.g. speech analysis), the performance differences cited above may not be so apparent due to

(a) the addition of noise to the input signal

(b) imperfect impulse driving functions

(c) the use of a larger number of data samples

Figure 4.7 shows the estimated a_2 as a function of time for the same 4 pole signal as above, with an impulse forcing function at regular intervals of 100 samples, plus white

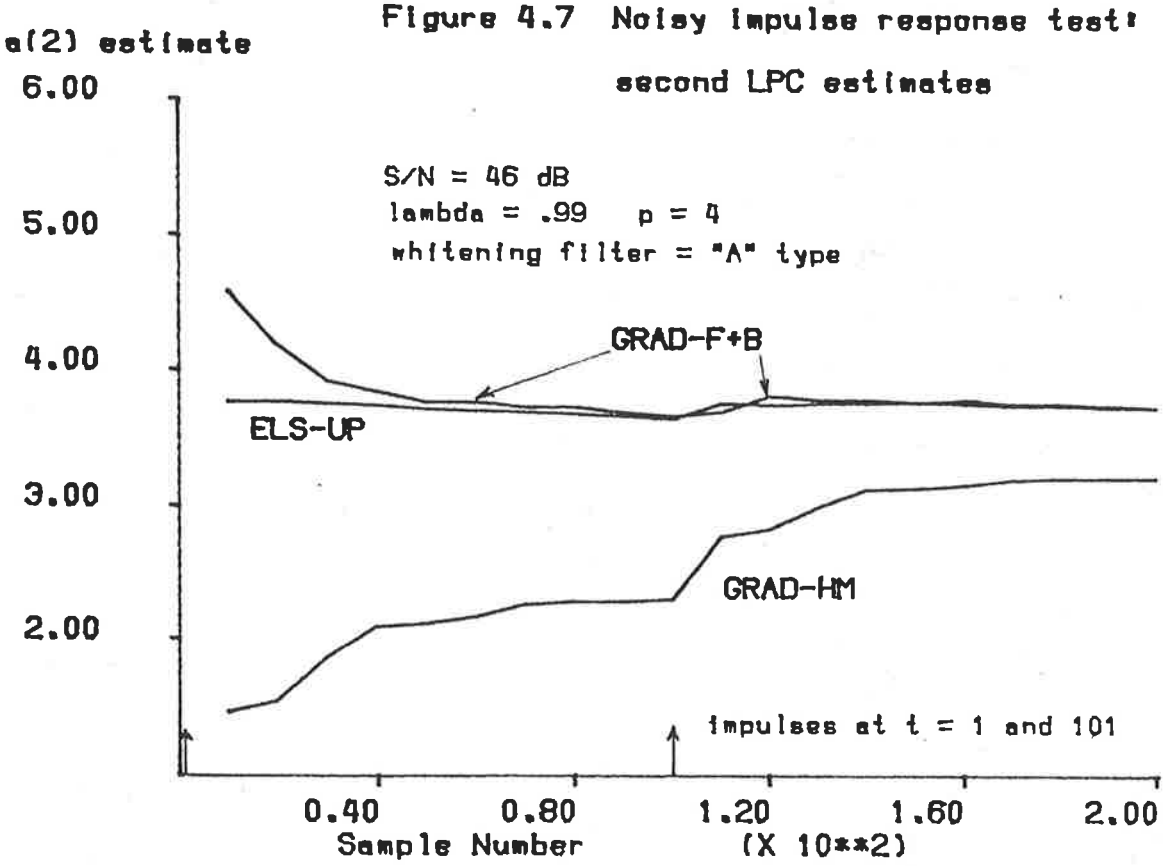


Figure 4.7 Noisy impulse response test: second LPC estimates

noise of small amplitude (mean signal to noise power ratio was 46 dB). The plot is for a single realisation of the impulse plus noise. Little difference is observed between the GRAD-F+B and the ELS-UP algorithms after the second impulse. The ELS Estimate may be seen to decrease slowly between impulses due to the progressive effect of noise being included in the LS solution. The faster response of ELS-UP at the second impulse is also of interest.

The following table shows the number of arithmetic operations used in the various exponentially weighted lattices. The second group of the table counts multiplication by the forgetting factor λ, as a shift and subtraction rather than a multiply operation. Each group then shows the number of multiplications (first column), divisions (second column) and shifts, additions and subtractions (third column). (The factor of 2 in (3.10)

is counted as a shift.)

lattice	defining equations	*	/	s+-	*	/	s+-
GRAD-HM	(3.6),(3.7),(3.10)	7	1	6	5	1	10
GRAD-F+B	(3.6),(3.7),(3.20),(3.21)	8	2	5	5	2	11
ELS-UP	(4.9a)-(4.9h)	7	4	6	6	4	8

The ELS-NP algorithm has not been included in the above list because a special architecture is appropriate, as mentioned above. Due to its attractive normalisation features a shorter wordlength would be required to achieve the same performance as unnormalised algorithms. The number of storage locations or memory required in each case might seem a useful specification, but depends rather on the algorithm coding. ELS-UP, for example, requires a minimum of about $3p$ locations (not counting temporary stores), but is very much easier to implement if about $8p$ locations are used. Memory requirements would not usually be a limitation.

Care must be exercised in drawing conclusions about the relative computational requirements from the above. Execution speeds of different instruction types, data input/output flow and algorithm control vary widely amongst possible implementations. To consider the TFB design again (Appendix E), since its inherent parallelism would generally enable shifts, additions, subtractions and data transfers all to occur during multiply and divide operations, and that divisions take twice as long as multiplications, then the relative minimum sampling time intervals would be approximately: GRAD-HM 7, GRAD-F+B 9, and ELS-UP 14.

This chapter has tried to present an overview of ELS lattice algorithms and show their relation to the previously discussed gradient algorithms. It is believed that additional insight into the operation of adaptive lattice filters may be gained by studying the similarities between these, largely independently derived, approaches.

Chapter 5 All-Pole Spectral Estimates

The all-pole spectral estimates were introduced in Chapter 2, where some new results relevant to adaptive transversal filters were summarised. This analysis is extended to lattice filters in the following section, with the aim of providing, where possible, expressions for spectral bias and variance. This enables the lattice parameter controlling the effective number of data samples used per estimate, to be tailored to generate estimates of suitable quality.

Section 5.2 considers those situations where the results of the first section can not be applied. The distribution function of the spectral estimate is considered, with a view to determining confidence levels. Computer simulations, using various lattice algorithms presented in Chapters 3 and 4, are compared with predicted results in several examples.

5.1 Bias and Variance of Lattice All-pole Spectral Estimates

As explained in section 2.3, the relative all-pole spectral estimate is equal to the reciprocal of the predictor transfer function magnitude squared.

$$Q(\omega) = \frac{1}{|1 + \sum_{k=1}^{p} a_k e^{-jk\omega}|^2}$$

Estimates of the LPCs (a_k, for $k = 1, \ldots p$) are derived from an adaptive lattice by using its reflection coefficients, as described in section 4.3. The (noisy) LPC estimates will again be denoted by dashes:

$$Q'(\omega) = \frac{1}{|A'(\omega)|^2} \tag{5.1a}$$

$$\text{where} \quad A'(\omega) = 1 + \sum_{k=1}^{p} a'_k e^{-jk\omega} \tag{5.1b}$$

$$a'_k = a_k + n_k, \quad k = 1, \ldots p \tag{5.1c}$$

where n_k is the noise on the $k-$th LPC estimate. Also define the vector

$$\underline{n} = (n_1,\ n_2, \ldots n_p)^T$$

The following analysis uses the result of Mann and Wald [53], that the n_k components are asymptotically unbiased and normally distributed, when a'_k, $k = 1, \ldots p$ are the least squares estimates of the LPCs (e.g. obtained by solving the normal equations (2.5a) using the estimated autocorrelation function). This is a good approximation in the stationary large sample case. The adaptive lattices provide LPC estimates which are either exact least square solutions, or slightly suboptimum solutions, in the case of the gradient algorithms. The former ELS algorithms give "prewindowed" or "covariance" estimates, as explained in Chapter 4, both of which tend to the same "autocorrelation" solution as N, the number of samples, becomes large. The gradient lattices might be expected to give larger biases and variances than their ELS (or block) counterparts because of their less than optimum error minimisation. Simulations using the best gradient algorithms indicate that often this degradation is small.

The covariance matrix of the LPC noise components is shown [53] to be proportional to the inverse of the signal covariance matrix, where the (zero-mean) signal y_t is assumed to be $p-$th order all-pole, i.e.

$$\underline{n} \sim N(\underline{0}, S) \tag{5.2a}$$

$$\text{where} \quad S = \frac{\sigma_p^2}{N} R^{-1} \tag{5.2b}$$

R is the signal covariance matrix, σ_p^2 is the $p-$th stage prediction error variance and N is the number of data samples used in the LPC estimation, assumed to be large, since (5.2a) is an asymptotic result. No assumptions are made regarding the distribution of the data sequence y_t. Now define

$$\begin{aligned} x(\omega) &= A'(\omega) - A(\omega) \\ &= \sum_{k=1}^{p} n_k e^{-jk\omega} \end{aligned} \tag{5.3}$$

Thus, from the above

$$E\{x(\omega)\} = 0 \tag{5.4a}$$

$$E\{A'(\omega)\} = A(\omega) \tag{5.4b}$$

$$\text{and let} \quad r(\omega) = E\{|x(\omega)|^2\} = Var(A'(\omega)) \tag{5.5}$$

$$\underline{e} = (e^{-j\omega},\ e^{-j2\omega}, \ldots e^{-jp\omega})^T$$

(Let superscript $*$ denote complex conjugate, and superscript $+$ denote complex conjugate transpose.) Then

$$\begin{aligned} r(\omega) &= E\{(\underline{e}^T\underline{n})^*(\underline{n}^T\underline{e})\} \\ &= \underline{e}^+ S\underline{e} && (5.6a) \\ &= \sum_{m=1}^{p}\sum_{n=1}^{p} s_{mn}\cos(m-n)\omega && (5.6b) \end{aligned}$$

where s_{mn} is the m,n-th element of S. Also let

$$\begin{aligned} s(\omega) &= E\{(x(\omega))^2\} = E\{(\underline{e}^T\underline{n})(\underline{n}^T\underline{e})\} \\ &= \underline{e}^T S\underline{e} && (5.7a) \\ &= \sum_{m=1}^{p}\sum_{n=1}^{p} s_{mn}e^{-j(m+n)\omega} && (5.7b) \end{aligned}$$

To derive the bias and variance of $Q'(\omega)$, given $|x(\omega)/A(\omega)| < 1$, (and dropping ω subscripts)

$$\begin{aligned} Q' &= \frac{1}{|A+x|^2} \\ &= \frac{1}{|A|^2(1+\frac{x}{A})(1+\frac{x}{A})^*} \\ &= \frac{1}{|A|^2}\left(1-\frac{x}{A}+\left(\frac{x}{A}\right)^2\ldots\right)\left(1-\left(\frac{x}{A}\right)^*+\left(\frac{x}{A}\right)^{*2}\ldots\right) \end{aligned}$$

so

$$\begin{aligned} E\{Q'-Q\} &= \frac{1}{|A|^2}E\left\{\left|\frac{x}{A}\right|^2+2Re\left[\frac{x^2}{A^2}\right]\ldots\right\} \\ &= \frac{r}{|A|^4}\left(1+2Re\left[\frac{s}{r}e^{-j2\theta_A}\right]\ldots\right) && (5.8) \end{aligned}$$

$$\text{where} \quad A = |A|e^{j\theta_A}$$

$$\text{and} \quad E\{(Q'-Q)^2\} = \frac{1}{|A|^4} E\left\{\left(-\frac{x}{A} - \left(\frac{x}{A}\right)^* \ldots\right)^2\right\}$$

$$= \frac{1}{|A|^4} E\left\{2\left|\frac{x}{A}\right|^2 + 2Re\left[\left(\frac{x}{A}\right)^2\right]\ldots\right\}$$

$$\approx 2\delta\alpha Q^2 \quad \text{if} \quad \delta << 1 \tag{5.9a}$$

$$\text{where} \quad \delta(\omega) = \frac{r(\omega)}{|A(\omega)|^2} \tag{5.9b}$$

$$\alpha(\omega) = 1 + Re\left[\frac{s(\omega)}{r(\omega)} e^{-j2\theta_A}\right] \tag{5.9c}$$

then from (5.8)

$$E\{Q'-Q\} \approx \delta Q(2\alpha - 1) \tag{5.9d}$$

$\delta(\omega)$ is the normalised transfer function variance and the condition $\delta(\omega) << 1$ (equivalent to $\text{var}(A'(\omega)) << |A(\omega)|^2$) must be true for the results in (5.9a) and (5.9d) to be valid. This might be arranged by ensuring N in (5.2b) is large enough, and will be considered in more detail shortly. Note that the results in (5.9) are the same as the transversal filter case, (Appendix A (17) and (18)) but with different definitions of $\delta(\omega)$ and $\alpha(\omega)$. In the earlier case the covariance matrix S was assumed to be proportional to the identity matrix, allowing easy evaluation of quantities $r(\omega)$ and $s(\omega)$.

The above results may be further simplified by considering the $\alpha(\omega)$ term. From (5.9c) and the definitions of $r(\omega)$ and $s(\omega)$ it follows:

$$0 \leq \alpha(\omega) \leq 2 \qquad \forall\omega$$

$$\alpha(0) = \alpha(\pi) = 2$$

In the case of S proportional to I, Appendix A gives an expression for $\alpha(\omega)$ involving $\sin(p\omega)/\sin(\omega)$ in (19), and shows it equals unity $p-1$ times between 0 and π. For intermediate frequencies the function is close to unity, especially if p is large. With S given by (5.2b) a stronger statement may be made about $\alpha(\omega)$ in the vicinity of strong peaks in $Q(\omega)$, at least for the 2 pole case. The peaks are of most interest as, from (5.9a), they exhibit the greatest spectral estimate variance. The location of the single

peak in the p=2 spectral estimate may be found by differentiating $|A(\omega)|^2$ to find its minimum value. This produces

$$\cos\omega_m = \frac{-a_1(1+a_2)}{4a_2} \tag{5.10a}$$

Now consider the situation when a_2 tends towards 1, yielding a high dynamic range, sinusoid-like input signal. Then

$$\cos\omega_m \approx \frac{-a_1}{2} \tag{5.10b}$$

The signal covariance matrix R is

$$R = \begin{pmatrix} r_0 & r_1 \\ r_1 & r_0 \end{pmatrix}$$

thus from (5.2b)

$$S = c\begin{pmatrix} 1 & \frac{-r_1}{r_0} \\ \frac{-r_1}{r_0} & 1 \end{pmatrix} \tag{5.11}$$

where c is a constant of no interest here. Now solving (say) (2.5a) for p=2 gives

$$\frac{r_1}{r_0} = \frac{-a_1}{1+a_2} \tag{5.12a}$$

so as a_2 tends towards 1, using (5.10b)

$$\frac{r_1}{r_0} \approx \cos\omega_m \tag{5.12b}$$

$s(\omega)$ may be evaluated using (5.7b),(5.11) and (5.12):

$$\begin{aligned} s(\omega) &= c(e^{-j2\omega} + e^{-j4\omega} - 2\cos\omega_m e^{-j3\omega}) \\ &= 2ce^{-j3\omega}(\cos\omega - \cos\omega_m) \\ \text{so} \qquad s(\omega_m) &\approx 0 \end{aligned} \tag{5.13}$$

To continue this enquiry $s(\omega)$ was determined in the following manner for several all pole signals (using 2 to 6 poles with dynamic ranges of <20 dB to >60dB). Starting

from the all-pole coefficients $a_1^p, \ldots a_p^p$ used in the signal synthesis filter, the LPCs for lower order stages were derived by using a backwards Levinson recursion [1]:

$$a_j^{i-1} = \frac{a_j^i - a_i^i a_{i-j}^i}{1 - (a_i^i)^2} \qquad 1 \leq j \leq i-1 \tag{5.14}$$

(This equation may be derived by rearranging (2.6c) and (2.6d)). The normalised autocorrelation function (assuming $r_0 = 1$) was then calculated (e.g. from (2.5a) assuming the $n-$th order)

$$r_n = -\sum_{k=1}^{n} a_k^n r_{n-k} \qquad 1 \leq n \leq p \tag{5.15}$$

following which the unnormalised autocorrelation function could be determined for specified σ_p^2 using (2.5b). The matrix R was then inverted, using Gauss-Jordan elimination, allowing calculation of $r(\omega)$, $s(\omega)$ and other quantities from (5.6b), (5.7b) etc. These evaluations showed that $s(\omega)$ was approximately zero at spectral peaks for all signals tested, and therefore the corresponding $\alpha(\omega)$ terms were approximately 1. Figure 5.1 shows a plot of $\alpha(\omega)$ for the 4 pole, high dynamic range signal "ub4p", used in Chapter 4. The locations of the twin spectral peaks are shown, and at these frequencies α is close to 1.

The results in (5.9) may thus be simplified: if ω_m is the frequency of a peak in $Q(\omega)$ and $\delta(\omega_m) << 1$, then

$$E\{Q'(\omega_m)\} \approx (1 + \delta(\omega_m))Q(\omega_m) \tag{5.16a}$$

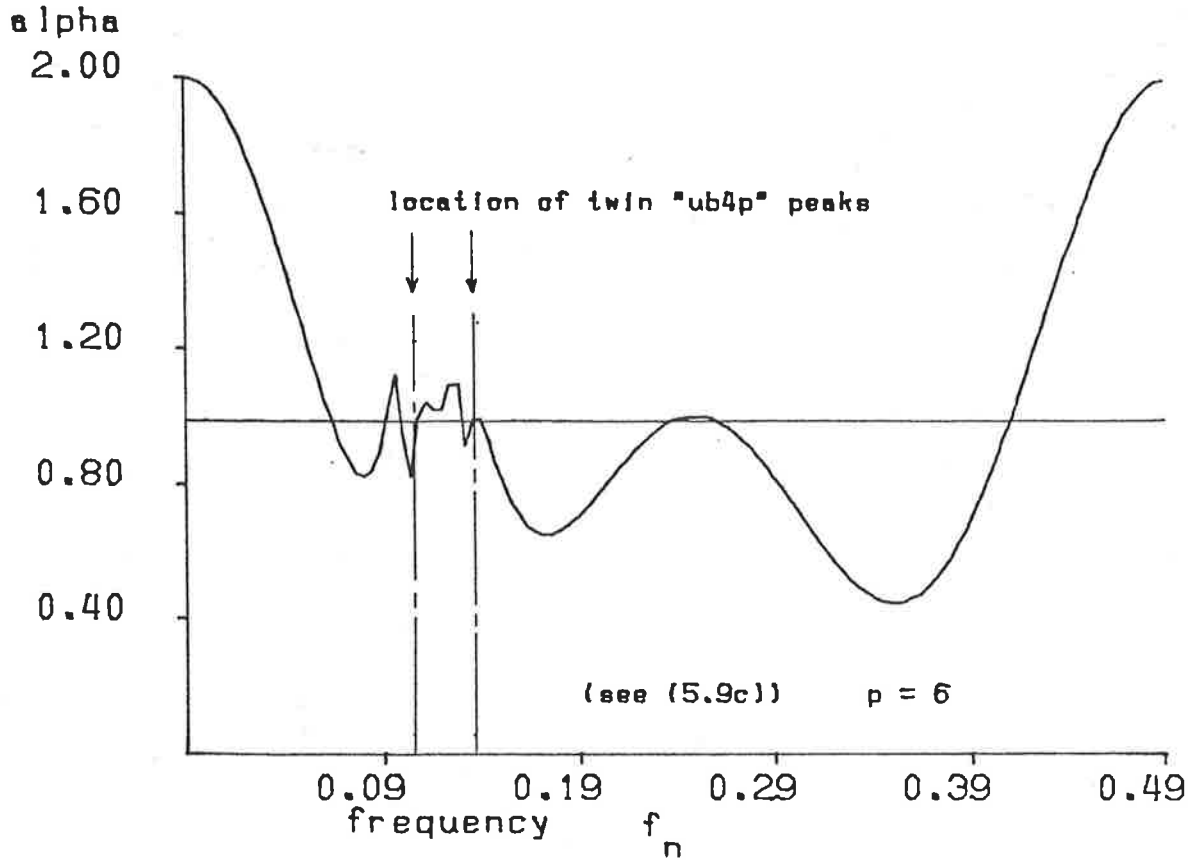
$$E\{(Q'(\omega_m) - Q(\omega_m))^2\} \approx 2\delta(\omega_m)Q^2(\omega_m) \tag{5.16b}$$

A consequence of these equations is that, as pointed out in Appendix A, the normalised bias in $Q'(\omega)$ will usually be small compared to the normalised variance, and thus may be neglected, e.g. if $\delta = .01$,

bias	$\approx$ 1 per cent
standard deviation	$\approx$ 14 per cent

and (5.16b) is thus a good estimate of Var(Q') if $\delta << 1$. Furthermore the number of "degrees of freedom", commonly used to measure power spectral estimate stability,

Figure 5.1 Plot of alpha function for "ub4p"



and defined by twice the average value of the spectral estimate squared divided by the variance [82], is approximately equal to the reciprocal of δ.

Having derived the preceding results concerning spectral estimate bias and variance, it was found that Akaike had in part examined similar areas [52], and a comparison with some of his results is now made. He first shows that the $p-$th order inverse covariance matrix R^{-1} may be usefully factorised into the product of upper and lower triangular matrices, which are composed of LPCs for predictions of order 1 to $p-1$, i.e.

$$R^{-1} = B^T D^2 B \tag{5.17}$$

where

$$B = \begin{pmatrix} 1 & 0 & \dots & 0 \\ a_1^1 & 1 & \dots & 0 \\ a_2^2 & a_1^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{p-1}^{p-1} & a_{p-2}^{p-1} & \dots & 1 \end{pmatrix} \qquad D^2 = \begin{pmatrix} \frac{1}{\sigma_0^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_1^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_{p-1}^2} \end{pmatrix}$$

This is a convenient result for the evaluation of quantities involving R^{-1}, as the matrix inversion previously described may be avoided; for example, knowing the $p-$th order predictors, use (5.14) and, rearranging (2.6e),

$$\sigma_{i-1}^2 = \frac{\sigma_i^2}{1-(a_i^i)^2} \qquad 1 \le i \le p \tag{5.18}$$

so R^{-1} may be determined. In practical lattice applications, however, the LPC estimates for all orders would be available via the "whitening filter" recursions using the reflection coefficients (section 4.3).

Substituting, (5.17) in (5.6a) and (5.7a) leads to

$$r(\omega) = \frac{\sigma_p^2}{N} \sum_{k=0}^{p-1} |A_k(\omega)|^2 \sigma_k^{-2} \tag{5.19a}$$

$$s(\omega) = \frac{\sigma_p^2}{N} \sum_{k=0}^{p-1} (A_k^*(\omega))^2 e^{-j2(k+1)\omega} \sigma_k^{-2} \tag{5.19b}$$

where $$A_k(\omega) = 1 + \sum_{l=1}^{k} a_l^k e^{-jl\omega} \tag{5.20}$$

In [52] Akaike goes on to derive a result for all-pole spectral estimate variance. After some manipulation this may be shown to be the same as (5.9a). The advantages in the current approach, with the use of the R^{-1} factorisation, are believed to be

(i) the introduction of $\delta(\omega)$ and the specification $\delta(\omega) << 1$ makes practical application of the results more useful (see also next section),

(ii) the simplification resulting from $\alpha(\omega_m) \approx 1$ makes (5.16b) considerably simpler to evaluate than Akaike's variance formula.

To make use of these results for the exponentially weighted lattices, the relationship between the effective number of samples N, in (5.2b), and the weighting factor λ is required. This has not been previously established, to the author's knowledge. It may be approached by considering the cross-correlation between the prediction error e_t and the past data sample y_{t-l}, $l = 1, \ldots p$. The distribution of this cross-correlation is used by Akaike as a starting point in the above-mentioned work. In the (usual) rectangular window case, following [52]

$$C_{ye}(l) = \frac{1}{N}\sum_{n=1}^{N} y_{n-l}e_n \qquad l = 1, \ldots p \tag{5.21a}$$

For the exponentially weighted window the appropriate cross-correlation is

$$C''_{ye}(l) = (1-\lambda)\sum_{n=-\infty}^{N} y_{n-l}e_n\lambda^{N-n} \qquad l = 1, \ldots p \tag{5.21b}$$

Now by equating the variances of these two normally distributed cross-correlations, it follows that the LPC estimates will have the same statistics (and so will the spectral estimates) i.e. let

$$E\{C_{ye}(l)C_{ye}(k)\} = E\{C''_{ye}(l)C''_{ye}(k)\} \qquad l,k = 1, \ldots p \tag{5.22}$$

the left hand side is

$$\frac{1}{N^2}E\left\{\sum_{n=1}^{N}\sum_{m=1}^{N} y_{n-l}y_{m-k}e_ne_m\right\}$$

Although e_n is only uncorrelated with y_{n-l} for $l > 0$, according to established statistical results [52,57], this variance equals

$$\frac{\sigma^2}{N}r_{lk}$$

where

$$\sigma^2 = E\{e_n^2\}$$

$$r_{lk} = E\{y_{n-l}y_{n-k}\}$$

Using a similar argument, the right hand side of (5.22) equals

$$(1-\lambda)^2\frac{\sigma^2}{1-\lambda^2}r_{lk}$$

Hence by equating these expressions

$$N = \frac{1+\lambda}{1-\lambda} \tag{5.23}$$

This may be compared to the "intuitive" result that might be expected by applying the exponential window only to the time series; the weighting of the error squared in (3.13) is equivalent to a $\lambda^{0.5}$ weighting applied to the data [47]. The equivalent number of samples, N', in a rectangular window might then be expected to be (c.f. (3.12))

$$N' \approx \frac{1}{1-\lambda^{0.5}} \tag{5.24}$$

Note that for λ close to 1, (5.23) and (5.24) tend to the same result.

Application of these results, and an example

$Q(\omega)$ statistics might be estimated for an exponentially weighted lattice as follows. Assume the frequency of the peak spectral estimate is known and the LPC estimates for all orders are available from the whitening filter. (5.23) determines the value of N to be used in (5.19a), which may be evaluated using (5.20), say at the peak frequency. ((5.18) may be used to evaluate the error power ratios.) Now from (5.9b) the maximum estimated δ may be determined. If δ is much less than one, (5.16) will estimate the predicted spectral variance; otherwise a value of N (and so λ) may be determined to ensure this condition is met.

An example is now presented comparing measured and predicted spectral estimate bias and variance, and illustrating some important practical considerations for lattice spectral estimation. A stationary 2 pole signal was generated using the coefficients a_1 = -.8 and a_2 = .95, and input to adaptive 4 stage lattice filters with both gradient and ELS algorithms. (See Appendix B for details of simulation software). The sliding rectangular window length for the covariance lattice (ELS-NC) was 500 samples. The other lattices all used exponential weighting, so from (5.23) λ = .996. Three

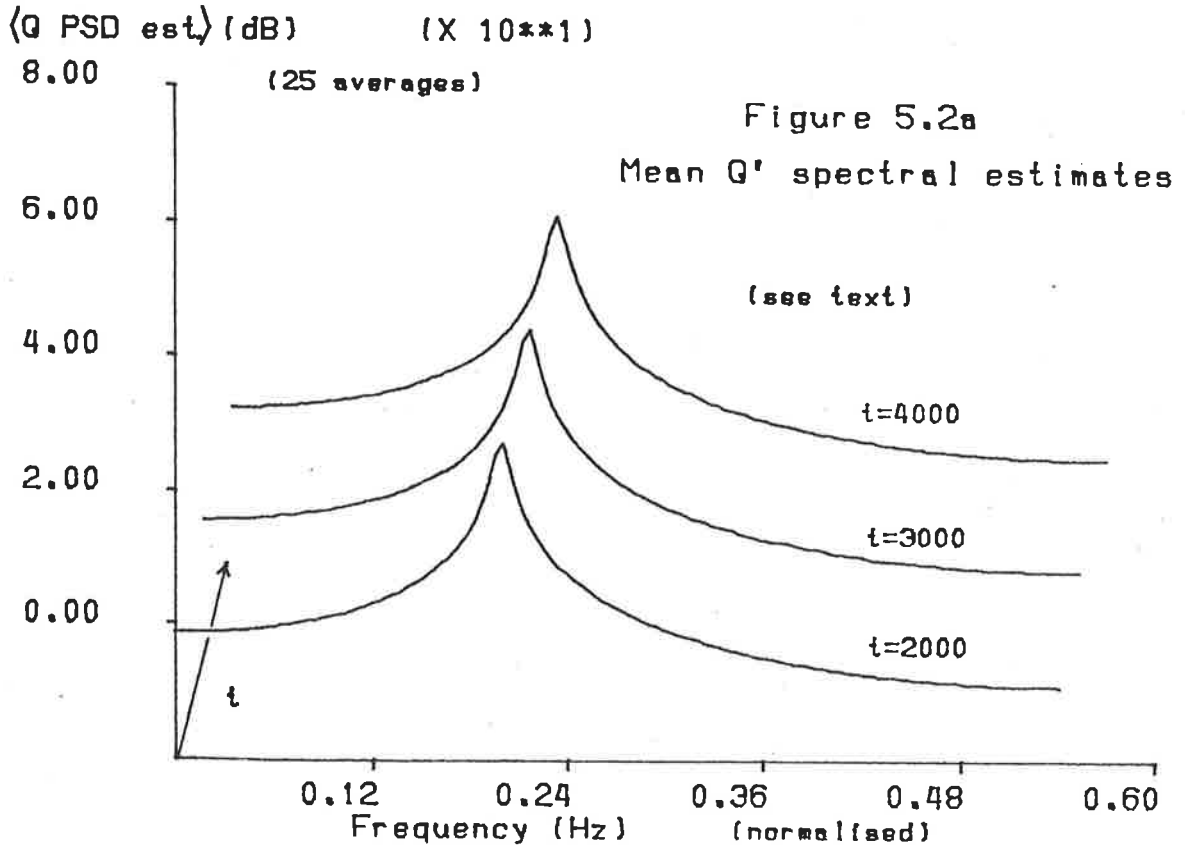
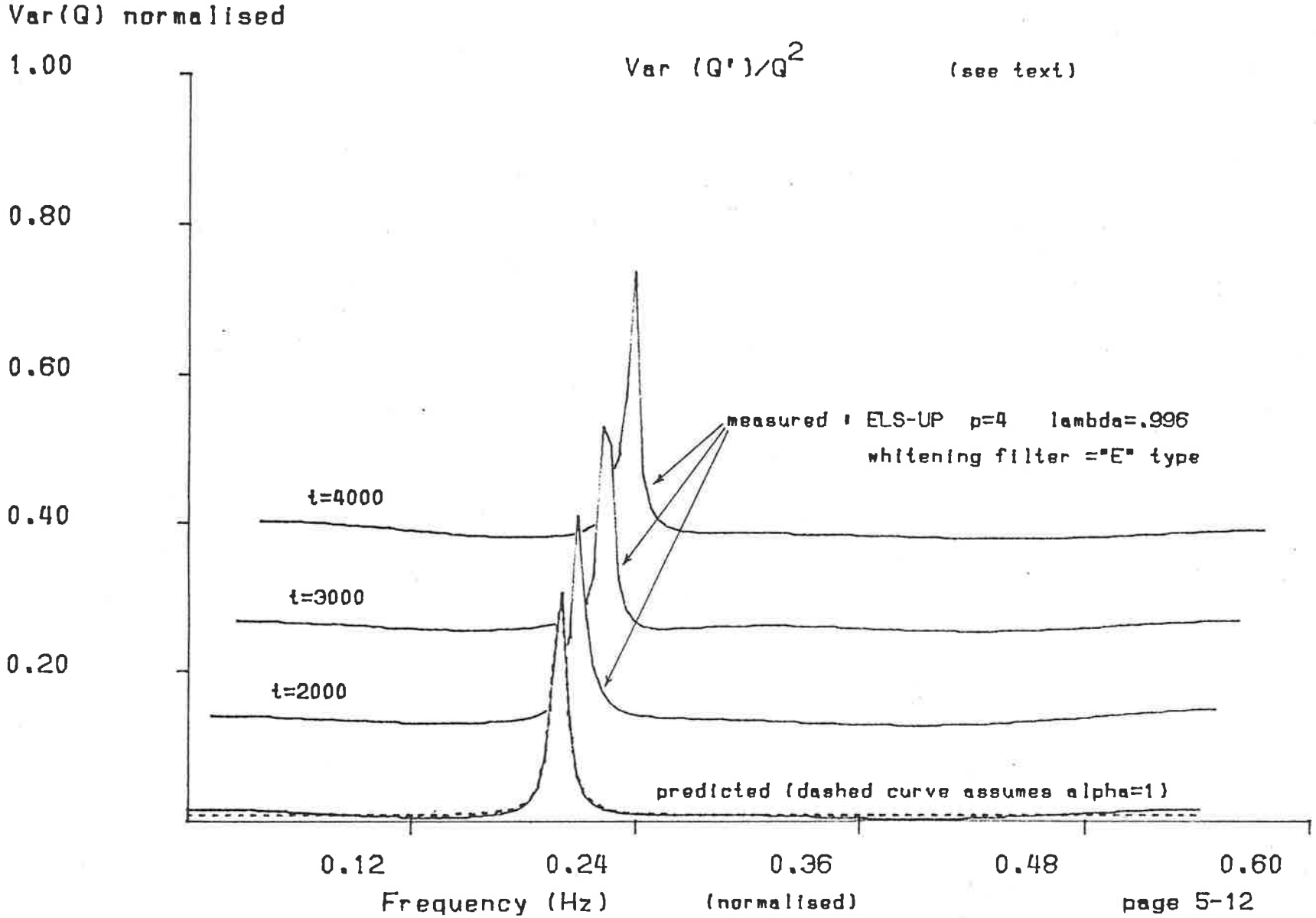


Figure 5.2a
Mean Q' spectral estimates
(see text)

spectral estimates were calculated for each of 25 realisations of the data, at $t = 2000$, 3000 and 4000. Quantities $\delta(\omega)$, $\alpha(\omega)$ etc. were predicted with the method described around (5.14) and (5.15). The calculated δ_{max} was .15 so no significant bias in $Q'(\omega)$ was expected (or found). Figure 5.2a shows the mean spectral estimates $< Q'(\omega) >$ from one simulation test. The plots are offset in the horizontal and vertical dimensions to give a three dimensional effect, with time as the third dimension. The results obtained by using different adaptive algorithms (listed shortly), or by calculating the all-pole model spectrum using the coefficients employed in the signal synthesis filter, were indistinguishable when plotted in this figure.

The simulated and predicted normalised variances show close agreement in Figure 5.2b. The latter variance is shown with and without the $\alpha(\omega) = 1$ assumptions, i.e. using (5.9a) and (5.16b), and the difference is small. Again the plots have been offset on the time axis for clarity, with the predicted variances plotted at t =1000 location

Figure 5.2b Predicted and measured Q' variance



for convenience. It may be observed that the variances are sharply peaked, and at other than spectral peaks are very small. Since the exact whitening filters give the optimum sets of forward and backward predictor coefficients (see section 4.3), either set or some combination may be used to generate the spectral estimates. Spectral estimates labelled "A" were derived from the optimum forward LPCs only (i.e. a_i, $i = 0, \ldots p$), while those labelled "E" made use of the forward *and* backward predictors, according to $(a_i + b_{p-i})/2$ for the $i-th$ component (see (3.4)). The simulated variances shown are for the ELS-UP lattice using "E" type spectral estimates. Other simulations had similarly shaped plots, with slightly different peaks (the following results are averages over all 75 estimates) :

result type	peak (var(Q')/Q^2)
predicted using (5.9a)	.30
predicted using (5.16a)	.31
simulated ELS-UP "E"	.31
simulated ELS-UP "A"	.38
simulated ELS-NC "E"	.34
simulated ELS-NC "A"	.38
simulated GRAD-F+B "E"	.37
simulated GRAD-HM	.37

Some comments regarding these results are in order. The sliding window and exponentially weighted ELS lattices gave similar performance due to the choice of λ. As might be expected the "E" type estimates exhibited only slightly lower variances than the "A" variety, as in this relatively large sample case the backward and forward predictors' estimates were almost identical. However it is worth noting that the lattice method does provide both predictors (in contrast to conventional LS estimation methods), and that better estimates may be made by making full use of the information. The gradient algorithms performed worse than the ELS "E" estimates but better than ELS "A" estimates. (The original motivation for considering "E" estimates was that the

gradient methods were performing better!) Some gradient methods (e.g. GRAD-HM) give only one set of predictors (by using one reflection coefficient per stage and implicitly assuming stationarity, see Chapter 3). Possibly the use of "A" spectral estimates might have advantages in non-stationary environments (e.g. better resolution, however see example at the end of this chapter).

Effective window length for small Q' variance when p=2

In the two pole case an easy method of ensuring that relatively unbiased and low variance spectral estimates are obtained, is possible. The following condition is required:

$$\text{var}(A'(\omega)) << |A(\omega)|^2 \quad \forall\omega \tag{5.25}$$

The minimum value of $|A(\omega)|^2$, using (5.10a) is,

$$|A(\omega_m)|^2 = (1-a_2)^2\left(1-\frac{(a_1)^2}{4a_2}\right) \tag{5.26}$$

$$\text{also} \quad S = \frac{\sigma_p^2}{N}\begin{pmatrix} r_0 & r_1 \\ r_1 & r_0 \end{pmatrix}^{-1} \tag{5.27a}$$

and from (2.5)

$$\frac{\sigma_p^2}{r_0} = \frac{(1-a_2)}{(1+a_2)}\left((1+a_2)^2-(a_1)^2\right) \tag{5.27b}$$

Using (5.6a), (5.27a), (5.12a), (5.27b) and (5.26), then substituting in (5.25) and simplifying leads to

$$N >> \frac{2(1+a_2)}{(1-a_2)} \tag{5.27c}$$

e.g. for the signal just considered $a_2 = .95$ so $N >> 78$ (in fact $N = 500$ was used, and δ was $\approx .15$). For exponentially weighted lattices, using (5.23) leads to

$$1-\lambda << \frac{1-a_2}{1+a_2} \tag{5.27d}$$

In practice this scheme would operate by allowing the adaptive lattice to converge, thus enabling estimation of a_2 (= K_2 for GRAD-HM), from which N or λ could

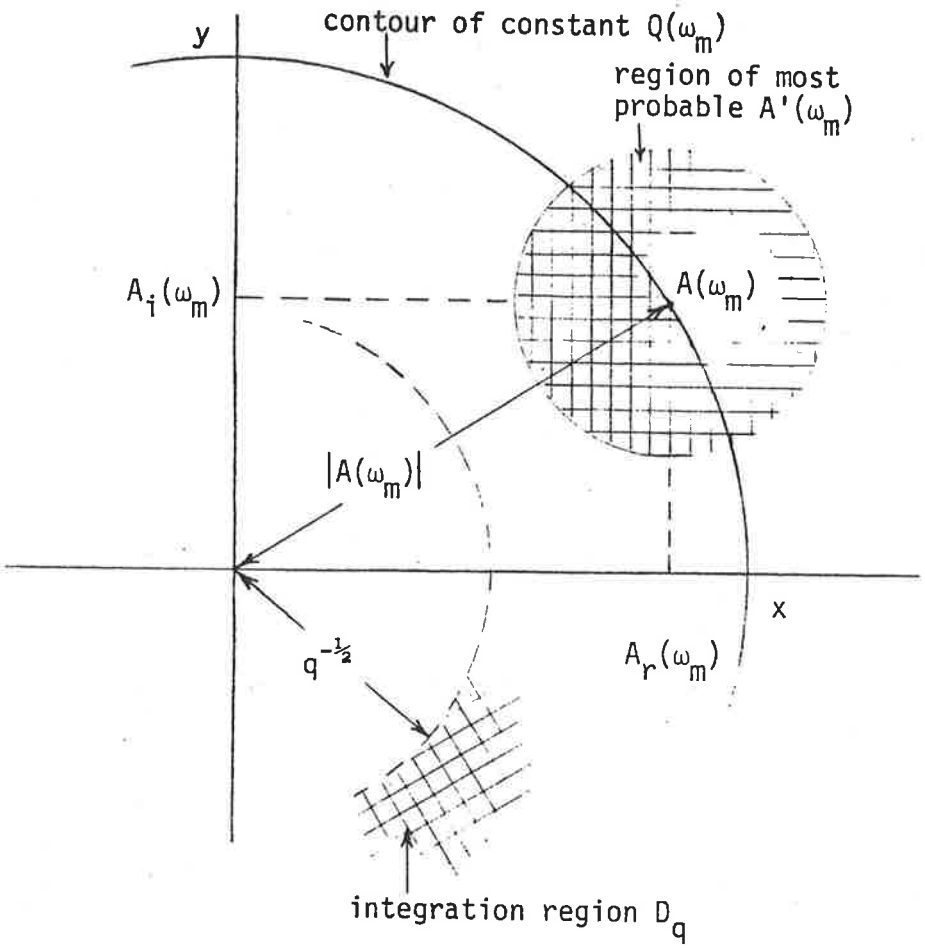
be deduced. Generalisation of this result for higher cases appears to be difficult. A crude approximation would be still to use a_2 as above, thereby, for the purpose of estimating N, fitting a 2nd order all-pole model to the signal. This is likely to give poor results when the input is not well modelled using two poles (perhaps when $< e_2^2 >$ is much larger than $< e_p^2 >$ a larger safety factor could be allowed in setting N). The alternative more computationally expensive technique, as already explained, would be to evaluate both sides of (5.25) at ω_m using (5.19) and the LPC estimates from the whitening filter.

5.2 Confidence Levels for All-pole Spectral Estimates

While the previous results on predicted $Q(\omega)$ variance are useful, they may only be employed if $\delta(\omega) << 1$ and they give no direct information about the distribution function of $Q(\omega)$. The latter will now be considered, under conditions for which $\delta(\omega) << 1$ may not be true, with the intention of providing confidence levels for $Q(\omega)$ estimates. Note that the LPC estimates are still assumed to have their asymptotic normal distribution, but as N decreases this assumption is likely to become invalid. However, as will be seen, good agreement between predicted and measured distributions has been found for high dynamic range all-pole signals (> 60 dB) with N as low as 50. Baggeroer [56] has considered non-asymptotic MEM spectral estimate confidence limits using a generalised student's t-distribution for the predictor coefficients. His work is applicable to block methods of estimating the covariance matrix in which the data series is partitioned into segments which are independent and jointly normal. He also shows that as the sample size becomes very large his results are consistent with those of Akaike previously mentioned [52].

It is useful to consider $A'(\omega)$ as depicted in Figure 5.3. $A(\omega)$ moves about the complex plane as a function of frequency and is closest to the origin at $A(\omega_m)$. The lattice estimates of LPCs generate $A'(\omega_m)$, which lie in an approximately bell shaped region around $A(\omega_m)$. If the "width" of this region, which is inversely proportional to

Figure 5.3 A' in the complex plane



the square root of the effective data length N, is much less than $|A(\omega_m)|$, (corresponding to $\delta(\omega_m) << 1$) the distribution of $Q'(\omega)$ tends towards normal. On the other hand if var(A') is large, some $A'(\omega_m)$ will fall close to the origin and give rise to very large spectral estimates.

The distribution function of Q' may be analysed by resolving A' into its real and imaginary components:

$$\text{let} \quad A'(\omega) = A'_r(\omega) + jA'_i(\omega) \tag{5.28a}$$

$$\text{and} \quad A(\omega) = A_r(\omega) + jA_i(\omega) \tag{5.28b}$$

from (5.1b)

$$A'_r(\omega) - A_r(\omega) = \sum_{k=1}^{p} n_k \cos k\omega \tag{5.28c}$$

$$A'_i(\omega) - A_i(\omega) = \sum_{k=1}^{p} n_k \sin k\omega \tag{5.28d}$$

By considering (5.2a), A'_r and A'_i are unbiased. Their variance and cross-correlation may be calculated as follows:

$$\begin{aligned} \text{let} \quad \sigma_{re}^2 &= E\{(A'_r(\omega) - A_r(\omega))^2\} \\ &= E\{\sum_{k=1}^{p}\sum_{l=1}^{p} n_k n_l \cos k\omega \cos l\omega\} \\ &= \sum_{k=1}^{p}\sum_{l=1}^{p} s_{kl} \cos k\omega \cos l\omega \end{aligned} \tag{5.30a}$$

$$\begin{aligned} \text{similarly} \quad \sigma_{im}^2 &= E\{(A'_i(\omega) - A_i(\omega))^2\} \\ &= \sum_{k=1}^{p}\sum_{l=1}^{p} s_{kl} \sin k\omega \sin l\omega \end{aligned} \tag{5.30b}$$

$$\begin{aligned} \text{and} \quad E\{(A'_r(\omega) - A_r(\omega))(A'_i(\omega) - A_i(\omega))\} &= \sum_{k=1}^{p}\sum_{l=1}^{p} s_{kl} \cos k\omega \sin l\omega \\ &= \rho\sigma_{re}\sigma_{im} \end{aligned} \tag{5.30c}$$

where ρ is the correlation coefficient between the two jointly normal random variables A'_r and A'_i. In practice it seems that these variables are almost uncorrelated at the frequency of peak spectral estimate ω_m, although the intuitive reason for this is not obvious. This has been shown by evaluation of (5.30) at the peak(s) for several all-pole test signals, and may be proved in the 2 pole case. Using the assumption concerning a_2 tending towards 1, leading to (5.10b), and employing previous formulas for the 2 pole situation to evaluate (5.30) gives, after simplification,

$$\text{at } \omega = \omega_m, \qquad \sigma_{re}^2 = \sigma_{im}^2 \ (= \sigma_A^2)$$

$$\rho = 0$$

(Indeed, this follows from showing, in section 5.1, that $s(\omega) \approx 0$ at spectral peaks, since

$$\text{if} \quad s(\omega_m) = 0$$

$$E\{(x(\omega_m))^2\} = 0$$

$$E\{(A'_r(\omega_m) - A_r(\omega_m))^2 - (A'_i(\omega_m) - A_i(\omega_m))^2 - 2j(A'_r(\omega_m) - A_r(\omega_m))(A'_i(\omega_m) - A_i(\omega_m))\} = 0$$

$$\text{so} \quad \sigma^2_{re} = \sigma^2_{im}$$

$$\text{and} \quad \rho = 0 \;)$$

To simplify the following analysis it will therefore be assumed that A'_r and A'_i are independent at ω_m. Then as

$$\text{var}(A') = \sigma^2_{re} + \sigma^2_{im}$$

$$\text{and} \quad \sigma^2_A = \frac{\text{var}(A')}{2} = \frac{r(\omega_m)}{2} \tag{5.31}$$

the joint probability density function is

$$f(A'_r, A'_i) = \frac{1}{2\pi\sigma^2_A} e^{-\left((A'_r - A_r)^2 + (A'_i - A_i)^2\right)/2\sigma^2_A} \tag{5.32}$$

Thus to examine the distribution function of the spectral estimate at the peak frequency, ω_m, let

$$\begin{aligned} F_Q(q) = &\quad \text{prob} \quad \{Q'(\omega_m) < q\} \\ = &\quad \text{prob} \quad \{|A'(\omega_m)|^2 > \frac{1}{q}\} \end{aligned} \tag{5.33}$$

Now assume, without loss of generality, that in Figure 5.3, $A(\omega_m)$ lies at the point (η,0)

$$\eta = |A(\omega_m)| = (A^2_r(\omega_m) + A^2_i(\omega_m))^{0.5} \tag{5.34}$$

$$\text{then} \quad F_Q(q) = \int\int_{D_q} \frac{1}{2\pi\sigma^2_A} e^{-\left((x-\eta)^2 + y^2\right)/2\sigma^2_A} dx\, dy \tag{5.35}$$

where D_q is the region outside a circle of radius $q^{-.5}$. The shape of the integration region indicates a change to polar coordinates is appropriate:

$$x = r\cos\theta$$

$$y = r \sin\theta$$

$$\text{then} \quad F_Q(q) = \int_{q^{-0.5}}^{\infty} \frac{1}{2\pi\sigma_A^2} e^{-(r^2+\eta^2)/2\sigma_A^2} \, r \left[\int_0^{2\pi} e^{r\eta\cos\theta/\sigma_A^2} d\theta \right] dr \tag{5.36}$$

A substitution of variables simplifies further:

$$\text{let} \quad \nu = \frac{\eta}{\sigma_A} \qquad \left(= \left(\frac{2}{\delta(\omega_m)} \right)^{0.5} \right) \tag{5.37}$$

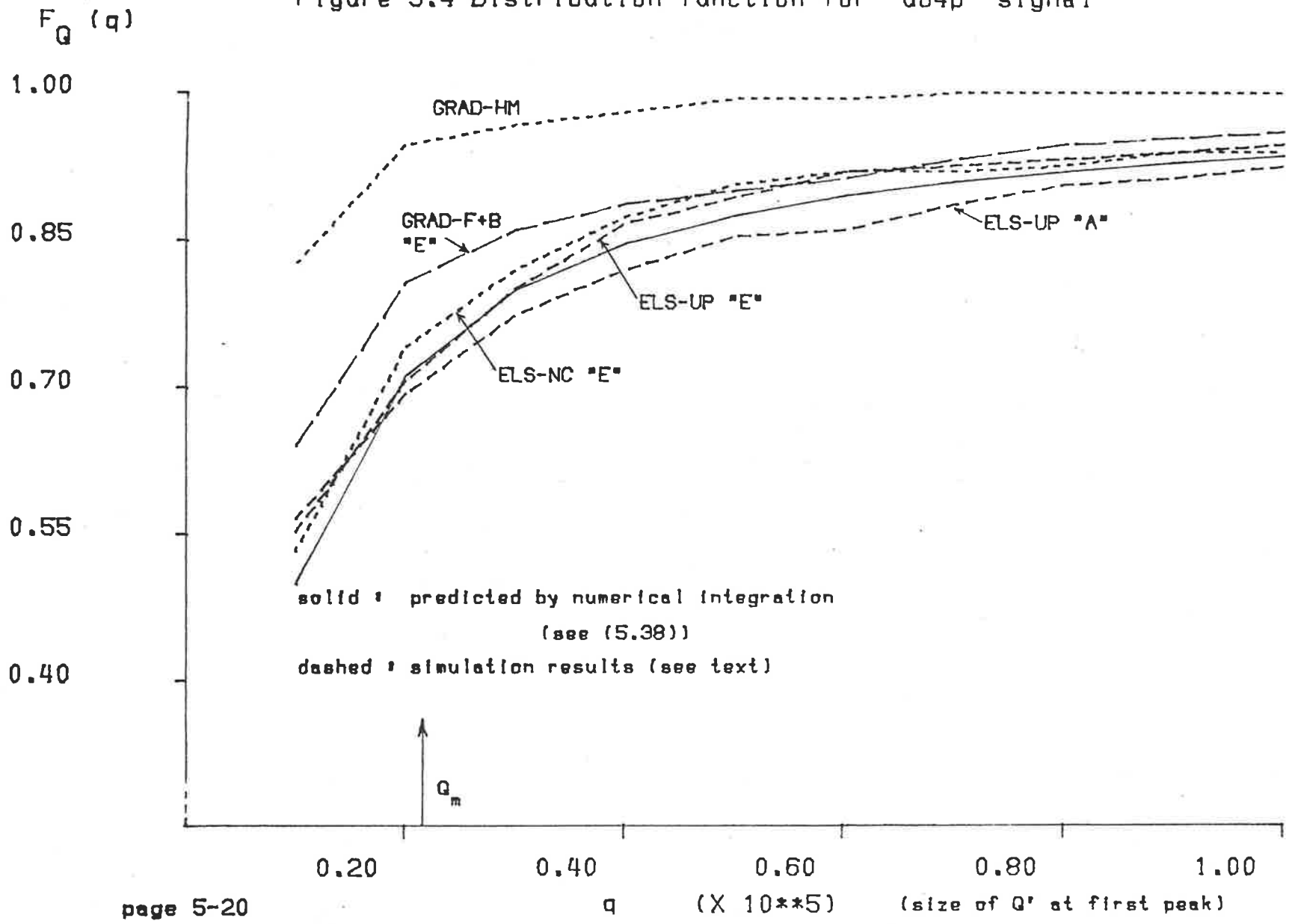
$$\text{and} \quad s = \frac{r}{\sigma_A}$$

$$\text{then} \quad F_Q(q) = 1 - \int_0^{\frac{q^{-0.5}}{\sigma_A}} s e^{-0.5(\nu^2+s^2)} \left[\frac{1}{2\pi} \int_0^{2\pi} e^{\nu s\cos\theta} d\theta \right] ds \tag{5.38}$$

It is interesting that this integral appears in the context of distribution functions of the envelope of a noisy sinusoid, studied by Rice [55]. He points out that the inner integral is a Bessel function. (5.38) has been numerically integrated, by Simpson's rule, for the 2 pole signal used in Figure 5.2, but with $N = 100$ samples instead of 500. The result agreed well with measured distribution functions from simulations of ELS and gradient lattices. To compare expected and actual performance on a more difficult example, the "ub4p" signal (Chapter 4, Figures 4.5 to 4.7) was tested with $N = 50$ samples ($\lambda = .96$). In this case δ_{max} was ≈ 1.5 so the formulas for normalised bias and variance were not applicable. Figure 5.4 shows the predicted and measured (by simulation over 50 realisations) distribution functions for the first peak of "ub4p". Simulation results included algorithms using forward predictor estimates only ("A" type), and forward and backward predictors ("E" type) to determine the spectral estimates. The lattices used 6 stages and spectral estimates were made at $t = 2000$, 3000, and 4000.

It can be seen that the distribution is highly non-symmetric about the model spectrum value Q_m, and exhibits a long tail at large q. The ELS "E" and "A" estimates were distributed fairly well as expected and in agreement with the numerical integration prediction. The results from gradient algorithms are perhaps of most interest in this example. The distribution of GRAD-F+B estimates agrees well at high q,

Figure 5.4 Distribution function for "ub4p" signal



but is significantly different elsewhere. GRAD-HM, which would have been expected to be comparable to the other gradient algorithm, has a quite different distribution function, from which large bias would be expected ($\approx$ 4 dB by comparing simulation mean spectral estimates to the all-pole model spectrum). The tests indicate that although there is little difference between adaptive algorithms in large sample "well-behaved" situations, this does not always hold. As the gradient algorithms become more like their ELS counterparts, their performance improves. This effect becomes more noticeable as the effective number of samples decreases and the signal dynamic range increases (thereby giving fewer independent samples). These conclusions are in accord with those of section 4.4.

Non-stationary data

Stationary data series have been assumed in this chapter until now. Analytical results for non-stationary data would be much harder to obtain, although this area may be investigated by simulations. Figures 5.5 (a) and (b) show some typical results. The familiar parameter jump test was used, with a_1 of the all-pole synthesis filter changing from -.5 to +.5 after $t = 50$, and a_2 staying fixed at .95. A four pole ELS-UP lattice was used with $\lambda = .98$ and "E" type prediction estimates. (For the corresponding stationary case, (5.27d) does not hold, so large normalised variances should be expected.) Figure 5.5(a) shows mean spectral estimates at intervals of five samples, while Figure 5.5(b) does the same for normalised $Q'(\omega)$ variance. In the latter case the plots for $t = 5$ and 10 have been omitted to avoid masking later results. It is of interest that the spectral estimate peak moves rapidly to the new position in a time much less than that expected from lattice coefficient convergence rates (i.e. a time much less than $(1-\lambda)^{-1}$). This is to be expected for the same reasons that rapid MSE convergence is often possible, as discussed in section 3.3 and shown in Figure 3.5. Another interesting feature is the drop in normalised variance during the period of most rapid adaption. This is probably due to an increase in the denominator of (5.9b) (due to a reduction in Q_{max}) rather

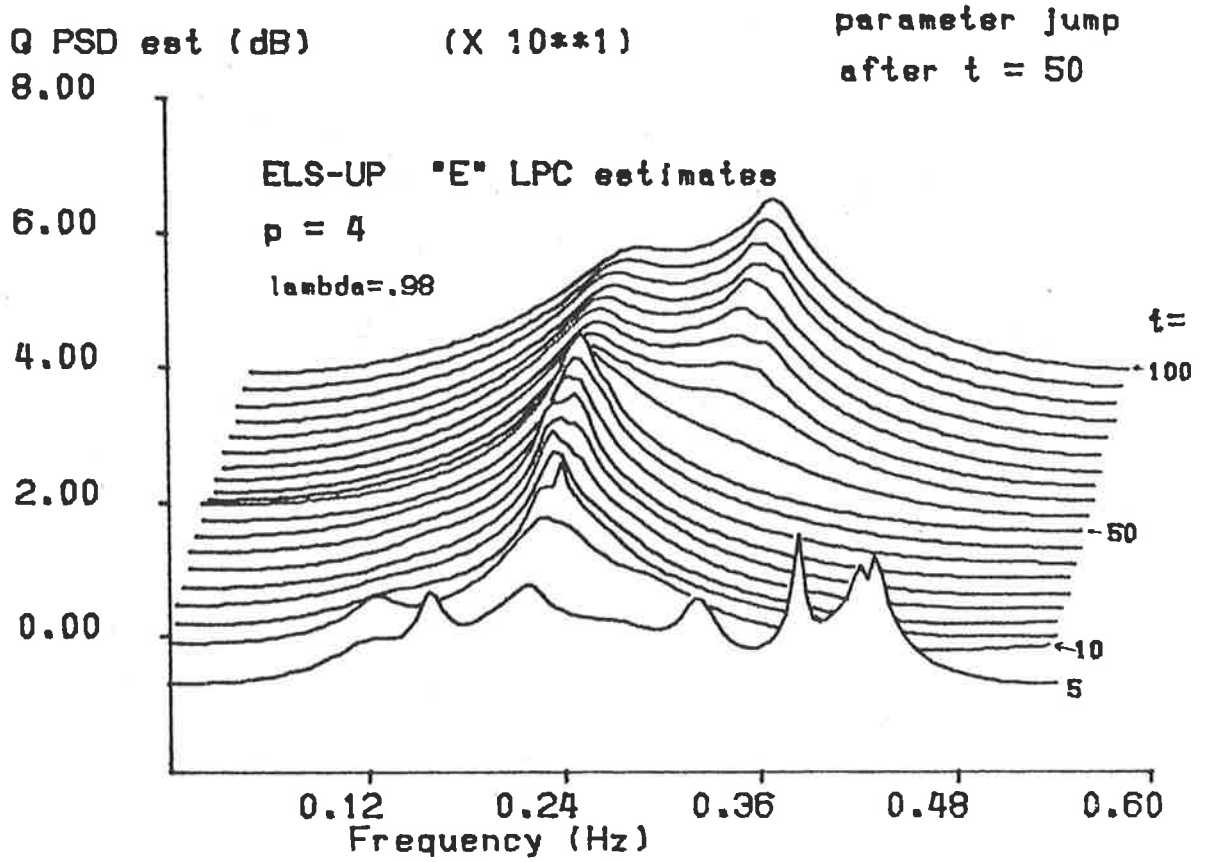


Figure 5.5(a) Mean spectral estimates for non-stationary test

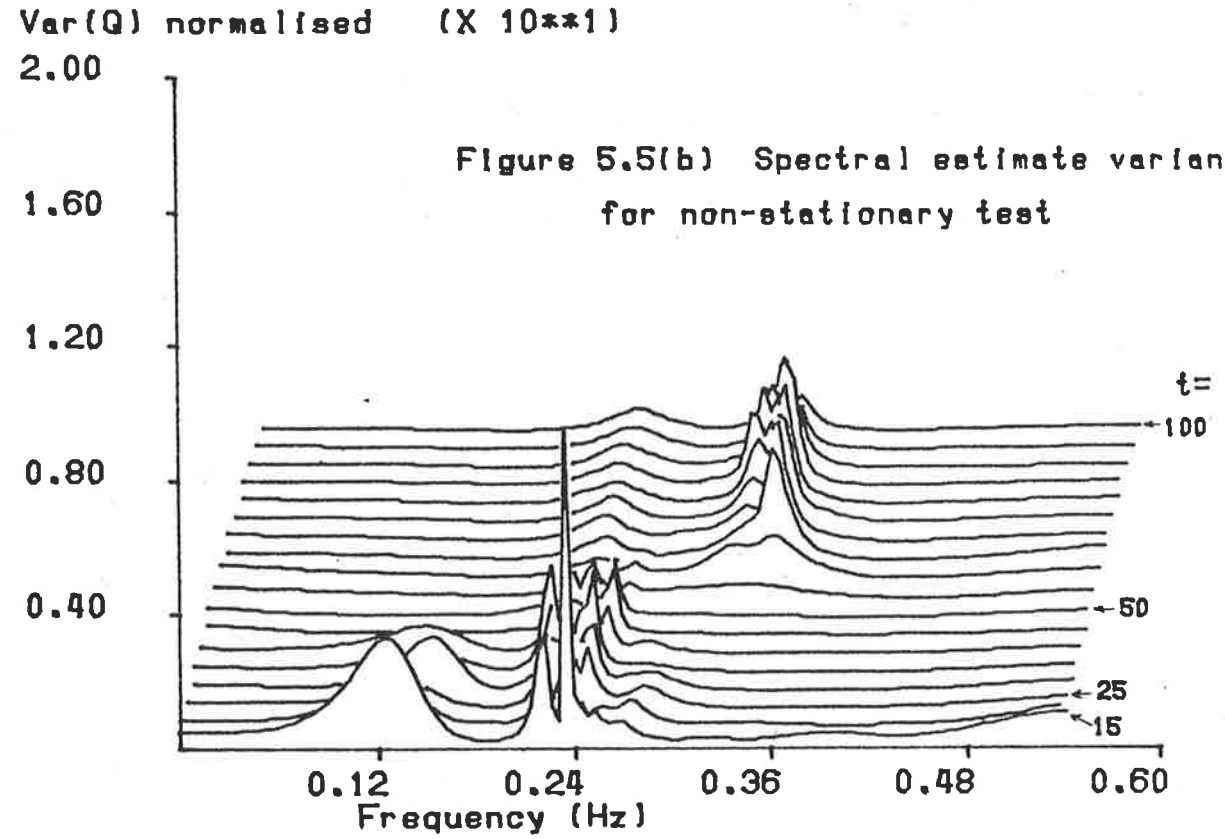


Figure 5.5(b) Spectral estimate variance for non-stationary test

than a change in the numerator. Finally, it may be stated that the same tests were repeated with ELS-UP using only the forward LPC estimates ("A" type), and although the stability was noticeably worse, the speed of convergence did not seem to be altered. GRAD-HM was also tested and gave very similar results to those shown.

Chapter 8 summarises some of the main results from this chapter and indicates how they may be applied.

Chapter 6 Implementation of LMS Algorithm

An implementation of the adaptive transversal filters discussed in section 2.2 is now described. This self-contained hardware uses the LMS algorithm and is capable of real time adaptive whitening, noise cancellation and all-pole spectral estimation at audio frequency sampling rates. Part of the motivation for the project was that the opportunity existed to realise a portion of the hardware in integrated circuit form. This was made possible by the first Australian Multi-Project Chip (MPC) exercise conducted by CSIRO, which enabled a number of VLSI circuits designed by different organisations to be fabricated on one wafer [29]. It was not possible to include all the adaptive filter components on the VLSI project; this portion comprised a recirculating memory for data samples and adaptive filter coefficients. The complete system, including VLSI component, fast arithmetic processor, controller and host microprocessor, is called the Adaptive Linear Predictor System (ALPS). A brief description of ALPS follows, with some addition information available in Appendix D. Section 6.2 explains how the LMS algorithm is realised in ALPS, and the last section mentions some possible extensions.

6.1 Brief Description of ALPS

Figure 6.1 shows a block diagram of ALPS. Components on the right hand side, including data store, arithmetic processor and finite state controller, perform the signal processing functions and may be called the "filter hardware". The remaining items, roughly on the left hand side, form a standard microprocessor system which carries out overall control and external communication. The filter hardware and host microprocessor operate largely independently (over separate buses) to achieve maximum processing speed.

Examining the VLSI data store first, the main components are two Recirculating Shift Register (RSR) blocks. Each of these is an array of shift registers, 8 bits wide by 12 words long, which (upon receipt of a shift command) moves data one place to

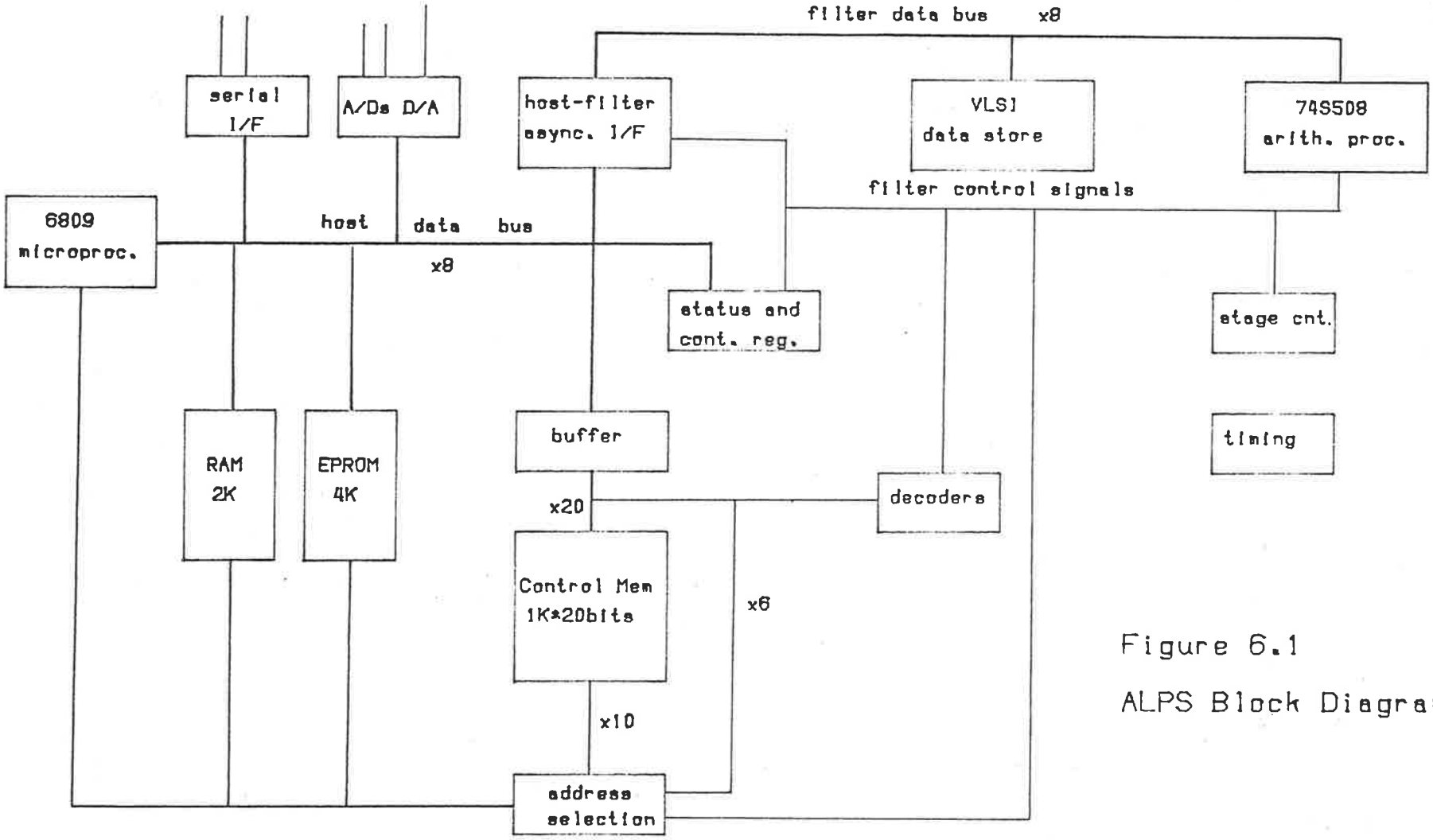


Figure 6.1

ALPS Block Diagram

the right and circulates the rightmost data to the leftmost storage element. Twelve successive shifts therefore circulate the RSR contents one complete revolution, leaving no net effect. Two other operations are possible on the RSRs: the rightmost word may be read to, or written from, the filter data bus. The VLSI data store also contains three temporary storage (scratch) registers with associated read and write operations to the external bus. Appendix D, section one, contains additional details of VLSI design, including a photograph of the fabricated circuit.

The fast arithmetic processor is a commercially available, bus oriented integrated circuit, capable of performing multiplication, division, sum of products and similar operations on 8 bit operands. Multiplications take approximately 0.5 microseconds at the maximum clock rate of 8 MHz. Other components of the filter hardware, such as the VLSI data store were designed to operate at this rate.

The filter hardware is controlled by a finite state machine along the lines described by Mead and Conway [71]. At any time the filter may exist in any one of 64 unique "states". Each of these states might control one part of the algorithm's execution; for example, add the current data sample y_t to the sum of products in (2.7). The progression from one state to another may only occur following a new clock cycle, and may be dependent on the status of various filter control signals. (To continue the previous example, do not proceed to the state which adds y_t until a signal from the host to filter interface indicates that this sample is available.)

The state transition information, in the case of ALPS, is stored in a random access memory (writable control store), which may be accessed by the host microprocessor during an initialisation period. This was done to achieve maximum flexibility. Apart from the possibility of multiple applications, each with its own algorithm and therefore unique control memory, the technique makes fault diagnosis much easier. To simplify the setup of control memory for different applications, a simple "microcode assembler" was written to translate from a conventional state transition table to filter control

memory contents. This runs on a UNIX computing system and allows "downloading" of the memory contents. An example of a finite state sequence is shown in appendix D, section 2.

The host microprocessor system performs overall control and supervision of the filter hardware and peripherals such as the A/D and D/A converters and serial interfaces. Raw data samples (or results) are presented to (or removed from) the filter hardware by means of an asychronous interface between the separate data buses, with a handshaking arrangement for communication. The host microprocessor software is largely interrupt driven. Of the two serial interfaces available, one is connected to a terminal for local control functions such as stop, start or change operating parameters. The other may be used to dump sets of filter parameters (weights) to a nearby minicomputer for subsequent analysis.

6.2 Algorithm realisation in ALPS

The adaptive transversal structure of Figure 1.1 and the LMS algorithm described in section 2.2 have been realised in the following manner. Assume that the current data sample y_t has just arrived via the host interface and has been stored in the VLSI data store. The upper RSR block now contains data samples y_t to y_{t-p} (again p is the filter order – 11 in this case), while the lower RSR block contains adaptive predictor coefficients a_0 to a_p as follows: $(a_0 = 1)$

$$y_{t-p} \quad \cdots \quad y_{t-1} \quad y_t$$

$$a_p \quad \cdots \quad a_1 \quad a_0$$

Evaluation of (2.7) proceeds by reading the rightmost element of each RSR, performing the multiplication required and then shifting both RSR blocks, thereby accumulating the sum of products which eventually represents the $p-$th order prediction error e_t. The error is sent back to the host and also a copy is stored in the first temporary storage

register (M1). The contents of the RSR blocks have now been returned to the state depicted above, having been circulated once completely.

The coefficient update, according to (2.8), can now be completed in similar fashion. The error is first scaled by the factor 2 μ and returned to M1. Then, after shifting both RSRs once, the rightmost data sample (top RSR block) is multiplied by the scaled error and added to the rightmost coefficient. The result is written back to the lower RSR as the new coefficient, and the process repeats p times. The RSR contents are then:

$$\begin{array}{ccccccc} y_{t-p+1} & & \cdots & & y_{t-1} & y_t & t_{t-p} \\ a_{p-1} & & \cdots & & a_1 & a_0 & a_p \end{array}$$

If the lower RSR is shifted once and the next data sample, when available in the following sampling interval, is written to the upper RSR, the data store will then have been returned to the original condition.

Not withstanding the limitation of 8 bit accuracy, the ALPS was successfully used to demonstrate adaptive whitening and all-pole spectral estimation. Test signals with various waveforms were generated by a function generator and pseudo random noise source. Adaptive whitening could readily be observed in the time domain by monitoring input signal and error output (the latter via the D/A converter). By periodically dumping filter coefficients to a minicomputer, on-line all-pole spectral estimates could be generated with (2.10). Figure 6.2 is an example of successive spectral estimates of a sinusoid in white noise, plotted versus frequency and time.

The filter simulation program FSIM (see Appendix B) can also simulate the fixed point arithmetic used in ALPS. Figure 6.3 shows the input and output signals when a sinusoid plus white noise was used. The signal to noise ratio was approximately 20dB, the input signal variance 12.8 and the effective adaption constant μ (after taking account of various data and coefficient scaling factors) was 0.0011. The effect of limited wordlength can be observed in the error series, although in this example the full precision

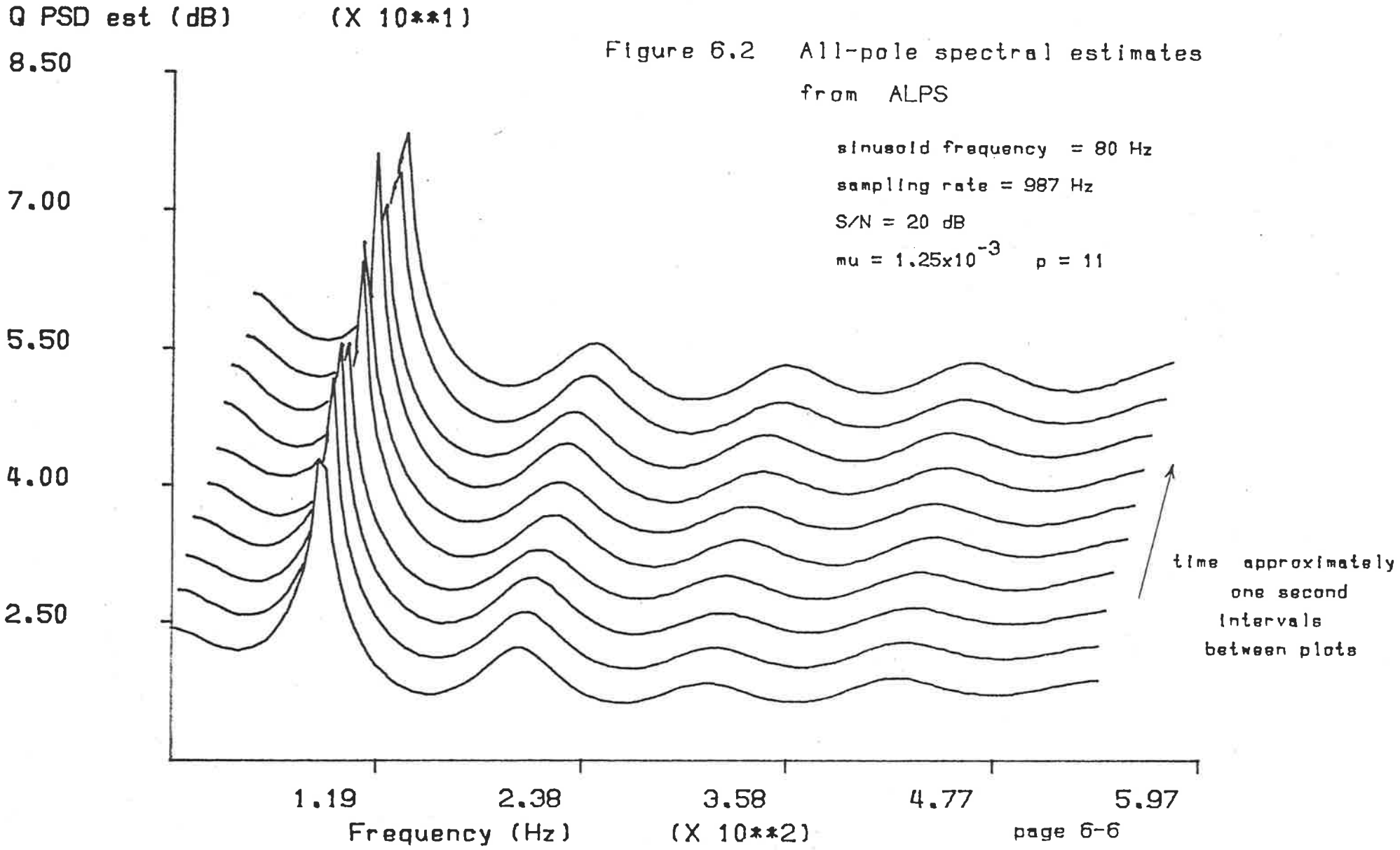
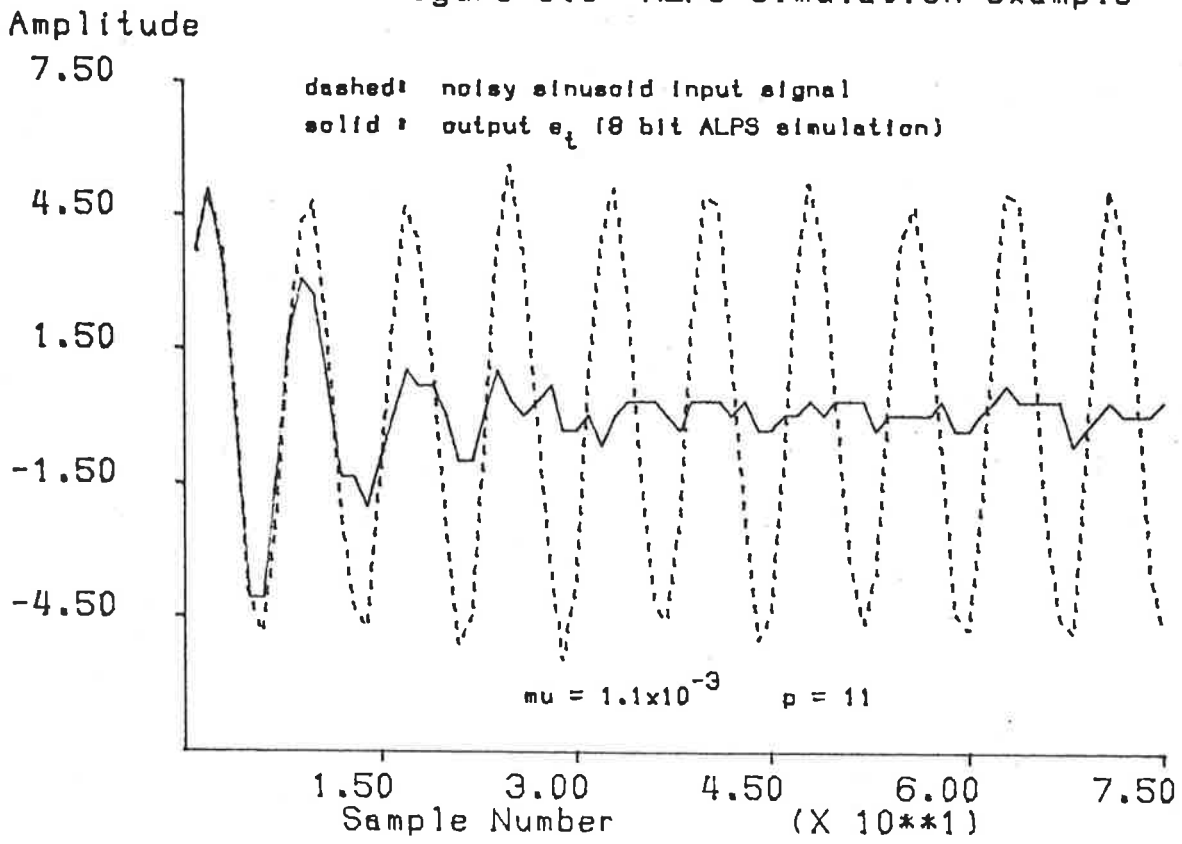


Figure 6.2 All-pole spectral estimates from ALPS

Figure 6.3 ALPS simulation example



simulation was very close to the 8 bit simulation result. Another consequence of the limited precision, which could be observed on other simulations and in practice, it that when the adaption constant is made too small the weight correction terms decrease below the least significant bit size of the weights and so adaption ceases. This indicates that higher precision for the weights, at least, would be useful. The inability to reduce μ often meant that the desirable condition for low spectral estimate bias and variance ($\delta << 1$, see section 2.4) could not be achieved (as in Figure 6.2).

After construction and testing of ALPS it transpired that the only unforeseen limitation was the speed of the host microprocessor. The filter hardware was capable of approximately 40kHz sampling rates for the signal processing tasks mentioned above, however the host could only keep up at about one tenth of this rate. This could be overcome by higher speed versions of the microprocessor, and by restructuring host

software.

6.3 Possible ALPS Extensions

A recirculating memory with only one currently accessible element can obviously be used in any signal processing application which must sequentially process a vector of data. With one such memory, applications such as scaling, mean value or variance estimation are possible. By using two recirculating memories, as in the VLSI data store described, dot products may be evaluated, thus enabling convolution, DFTs and IIR (all pole) or FIR (all zero, as in (2.7)) filtering. In addition the types of linear combinations used in (2.8) may be evaluated. Larger numbers of these memory elements would enable either greater precision to be obtained (two RSRs in parallel) and/or more variables to be stored (for example pole-zero filtering with 3 RSRs). Lattice filters may be implemented with two or more VLSI data store chips, as shown in Appendix D, section 3.

The concept of recirculating shift register memories is not new (for example see [19]), but no others in integrated circuit form are currently known. The technique is well suited to very large scale integration for the following reasons. Each dynamic shift register element, except the rightmost, only communicates with its neighbours. It may therefore be minimum size, low power and high speed. Process control requirements are simple compared to conventional random access memories. In most signal processing applications the memory contents will be accessed regularly, and so refreshing will not be a concern. If additional refreshing is required however, say due to very low sampling rates, the memory contents may be circulated periodically. This method is used in ALPS. Apart from simple fast memory elements the design dispenses with address decoding, sense amplifiers and similar circuitry needed in alternative memory designs.

This chapter has described a simple architecture and control scheme which is suited to a range of real time signal processing tasks, including adaptive filtering. The concept is well suited to integration, including the control and arithmetic elements.

Such a device could provide a low cost yet flexible solution to a number of applications previously mentioned.

Chapter 7 Radar Applications of Adaptive Lattice Filters

This chapter focuses on some specific applications of adaptive filters to the area of radar signal processing. The topics covered concern identification and classification of radar signals. The lattice structures are shown to be useful as adaptive prewhitening filters, whose coefficients may also provide information in some cases. Other radar applications of adaptive filtering techniques are possible but have not been covered in this chapter, e.g. adaptive radar antenna arrays [81]. Several results from Chapters 3 to 5 are used in the following, with both real and simulated radar signals being employed to demonstrate the lattice applications. A good general reference to radar is by Skolnik [69].

7.1 Moving Target Identification

A common occurrence in surveillance radar systems is the obscuring of wanted radar returns from objects of interest (hereafter called targets), by large returns from ground, sea or certain meteorological conditions. The latter unwanted returns are called clutter. Clutter removal or suppression is desirable, and may be achieved if the clutter and target signals differ sufficiently in some respect to allow analogue or digital signal processing techniques to separate them. A simple method of doing this uses the doppler frequency shift present in targets moving radially with respect to the radar. Assuming the clutter is stationary relative to the radar, its return signal from a specified range is substantially at D.C., and thus may be filtered from the signal leaving only the desired components. This technique is called Moving Target Identification (MTI, see [69] chapters 17,18).

The approach described above discriminates in the frequency domain and fails when the clutter has substantial high frequency components due to its movement. This may arise, for example, when a storm approaches an air surveillance radar, leading to poor performance when the radar is needed most. The conventional MTI is limited by

the assumption of a fixed clutter power spectrum. Haykin et al. [17, also 34,7] have used lattice filters to perform adaptive filtering of the radar return signal as an alternative solution, and reported good results. The idea of using an adaptive filter in this context might first have been suggested by Bühring [68].

The adaptive filter clutter suppressor operates in the following manner. In a (coherent) pulsed radar system the received signal is sampled many times after each pulse has been transmitted; each sample representing the return from a given range. As the radar scans in azimuth the time series from each range ring (with sampling rate equal to the Pulse Repetition Frequency (PRF)) is whitened by the adaptive filter, whose error output is then used for display or further processing. Clutter signals are present for a relatively long duration compared to the target signals since the former are due to distributed regions while the latter arise from point sources. For target returns the number of "hits" (samples) per target is a function only of the PRF, antenna beamwidth and scanning rate. The adaptive filter convergence rate is chosen to allow adaption to, and removal of, the clutter signals while not allowing sufficient time for removal of the targets. An interesting feature of this scheme is that no prior assumptions are made about the clutter and target power spectra; instead the discrimination is on the basis of temporal duration of the respective signals. Thus the new method should work equally well given rapidly moving clutter and almost stationary targets, as in the more usual converse situation.

The use of an adaptive lattice MTI on some real radar data is now described, and compared to results previously published. A simulation is then used to show the possibilities of the new method to an airborne search radar, where the problems caused by moving clutter are even more severe than cited above.

Lattice MTI using real data

The initial objective of these tests was simply to repeat the favourable results reported in [17], with local data. Unfortunately a lack of data under all operating

conditions, particularly those involving moving clutter, prevented achieving the marked improvements expected. Nevertheless the exercise led to some interesting observations and variations that are worth reporting. The simulations to be described shortly were also prompted by these tests.

Coherently demodulated received data from inphase and quadrature (I and Q) channels of an L band surveillance radar was digitised and stored. The number of pairs of samples from each radar pulse (range extent) and the angular sector (in azimuth) could be adjusted as required. The data was processed (off-line) by a minicomputer to determine the following MTI improvement factor.

$$I = \frac{(\text{peak target power / mean clutter power})_{OUT}}{(\text{peak target power / mean clutter power})_{IN}} \tag{7.1}$$

I is a function of the target and clutter signal statistics.

Initially the standard, one reflection coefficient per stage, gradient lattice was used (GRAD-HM, see Chapter 3, defined by (3.6), (3.7) and (3.10) and shown in Figure 1.2). One independent lattice was used for each of the I and Q channels. (A complex-valued version of the lattice was also tested but with little difference in performance, and this was expected since it was thought that the I and Q samples would be largely uncorrelated.) Four or five stage lattices were employed because the clutter power may be reasonably modelled with low order all pole models (e.g. see [14]). With typical radar data, whose clutter was close to stationary, the improvement (I) was found to be strongly dependent on the exponential weighting factor λ. As this approached unity I increased. In addition when the lattice MTI was compared to the conventional dual delay line MTI, simulated by

$$e_t = 0.5y_t - y_{t-1} + 0.5y_{t-2} \tag{7.2}$$

with relative improvement factor defined by

$$I_r = \frac{I\ (\text{lattice MTI})}{I\ (\text{conventional MTI})} \tag{7.3}$$

then I_r was slightly less than unity for $\lambda = .99$, and considerably less for smaller values.

A representative example of these results is shown in Figure 7.1 and Table 7.1. The figure represents a plot of signal power before and after MTI processing. Each symbol indicates the power in I and Q channels for the nominated range ring, averaged over 6 samples (equivalent to 1 degree in azimuth). A target is visible in range ring 8, below and on the edge of a region of heavy clutter due to nearby low ranges. The total region shown corresponds to an arc approximately 9 km wide (13 to 22 km), and 28 degrees in azimuth. The table summarises the mean clutter powers, peak target powers and relative improvement ratios (to the nearest dB) for two successive scans of the radar. (The last entry, GRAD-SL, will be explained shortly).

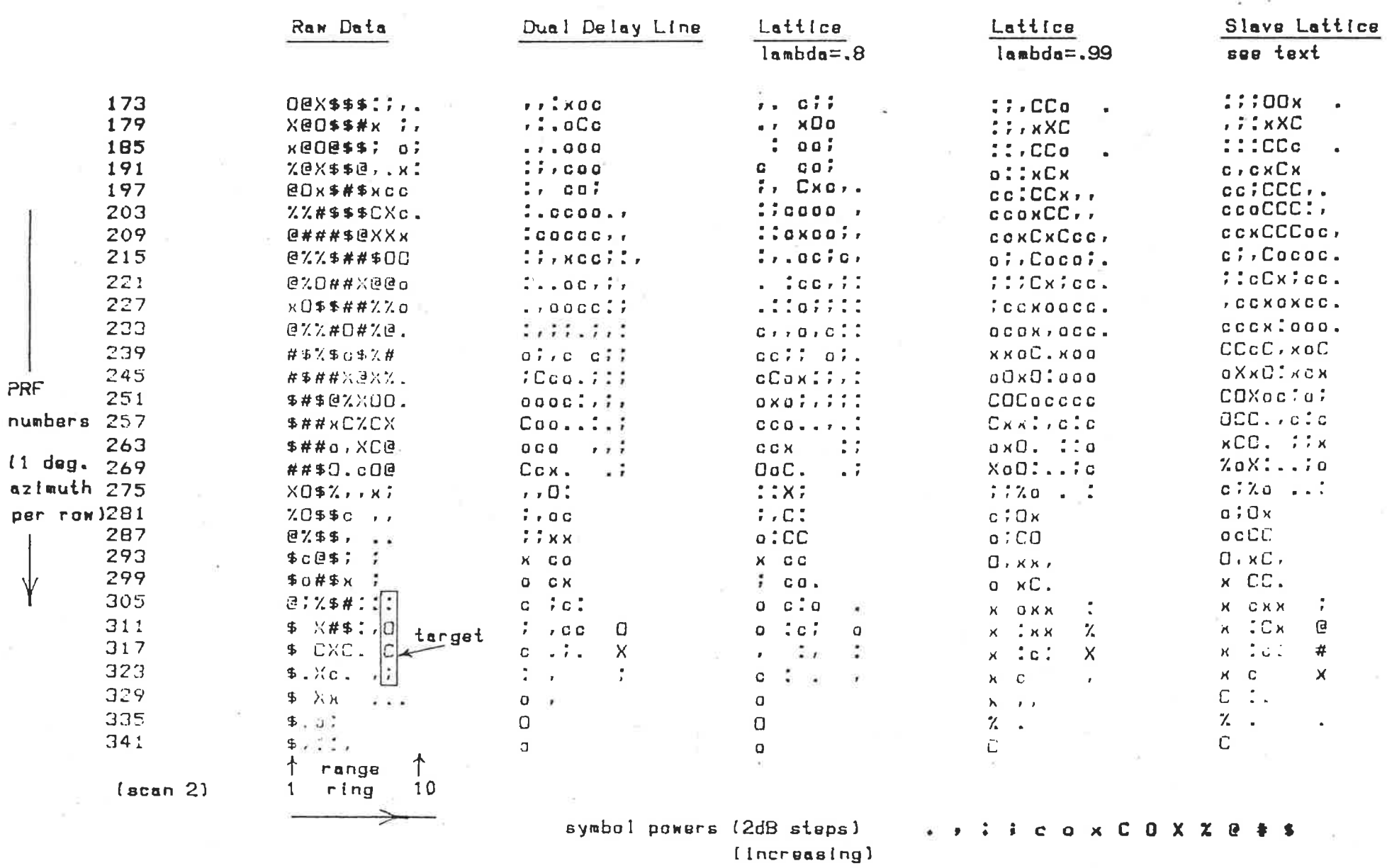
Table 7.1 MTI Performance Example

MTI type	mean clutter (4th ring)	peak target (**) scan 1	peak target (**) scan 2	I_r scan 1	I_r scan 2
none (raw data)	3530.	C	D	-17	-17
dual delay line	104.9	D	X	0	0
GRAD-HM $\lambda = .8,\ p = 4$	124.3	o	o	-7	-9
GRAD-HM $\lambda = .99,\ p = 4$	248.6	X	%	-2	-2
GRAD-SL $\lambda = .97,\ p = 4$	283.3	%	#	0	+2

(** target in 8th range ring, see Figure 7.1 for key to symbol powers)

Some comments on these seemingly poor results are appropriate. The optimum value of λ used in [17] of .6, was smaller than indicated from the above for at least two reasons. In that study the radar return was split into four time series to avoid non uniform sampling periods (employed to overcome doppler ambiguities). Initially it might be thought that to compensate for this effect the equivalent "unsplit" data window length could be deduced and applied in the present case. As an approximation, using

Figure 7.1 MTI performance example

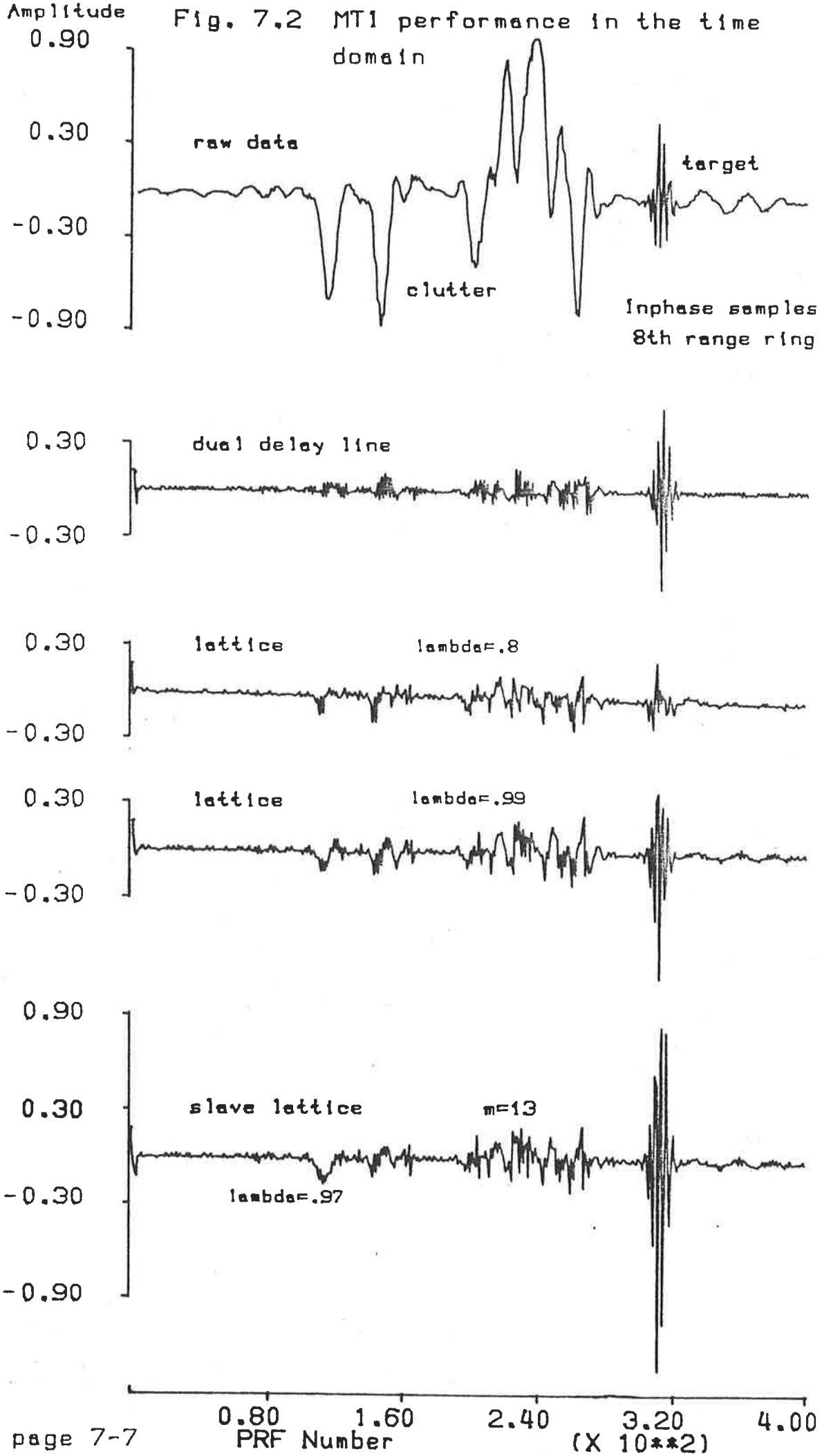


(5.23) or (3.12) to do this gives $\lambda \approx .9$. However this argument neglects the following target enhancement effect. The number of hits per target in [17] was reduced, from about the same number used in the present study, to a quarter of the original number (giving about 5 hits per target in each stream). This target duration is important because, as shown in section 3.3, the error convergence may be very rapid and not wholly dependent on the exponential weighting factor, especially if there are "unused" stages in the lattice. For a slowly increasing target amplitude the lattice may have time to adapt to the target and so reduce the improvement factor. This effect may be easily seen in the input and output time series of Figure 7.2, which covers the same data as the previous figure.

The conclusions to be drawn from these results are as follows. Target signal rise time has an important effect on lattice MTI performance, and should be as short as possible for best results. Clutter which is substantially stationary may be processed well by the conventional dual delay line canceller because of its double zero at D.C. The sample results of Table 7.1 and those of [17] Figure 6 are not inconsistent (the latter show absolute improvement rather than relative improvement factors), and it seems probable that for stationary clutter the lattice MTI was inferior to the dual delay line MTI in the previous study also. However their data obviously had a wide range of doppler shifts and therefore showed the lattice MTI under all conditions, so demonstrating its full potential. The present data on the other hand shows a worst case comparison between the lattice and conventional MTIs. The latter's performance can only deteriorate under different conditions, and indeed gives a degradation ($I < 1$) when the clutter peak is at a higher frequency than that of the target.

In an effort to improve the lattice MTI under stationary clutter conditions several ideas were tried. The most promising of these was to use a slave lattice, whose coefficients were determined by an adaptive master lattice. A delay of m samples between determining the reflection coefficients and passing them to the slave is designed to allow reasonably quick response in the master, but at the same time avoid removing

Fig. 7.2 MTI performance in the time domain



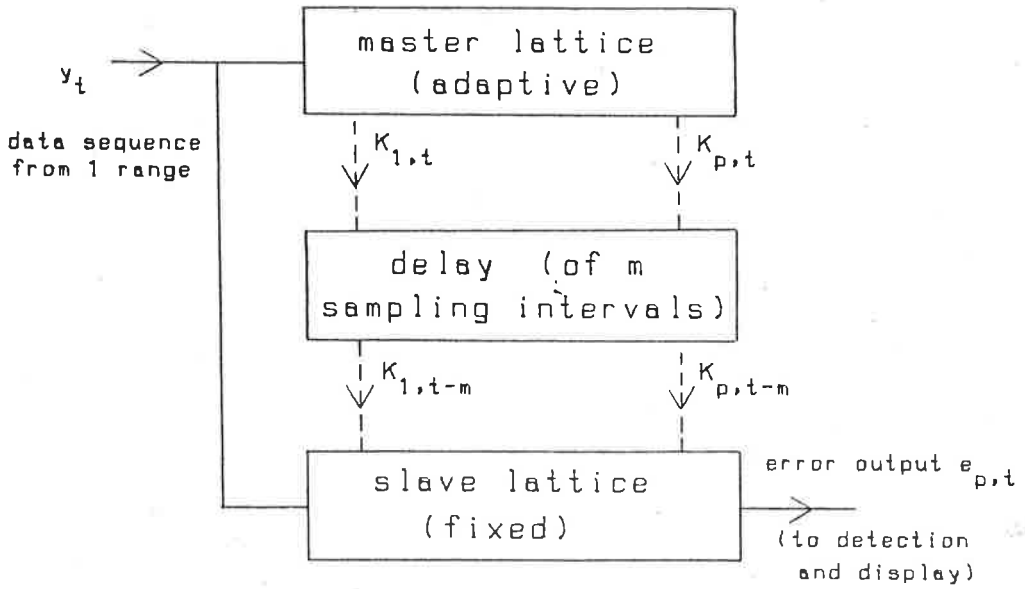


Figure 7.3 Slave lattice for MTI

targets from the slave output. The slave lattice is depicted in Figure 7.3.

Some results for the slave lattice MTI are shown in the previous table under the heading GRAD-SL. Although the mean clutter power is higher than the conventional result the peak target enhancement is even greater, so that the net effect is a slight relative improvement over the dual delay line canceller. Again this should be a worst case test for the slave lattice MTI. (A small amount of "slightly moving" clutter was available, in which the clutter peak frequency, normalised by the PRF, was about .07 instead of the usual 0 to 0.04. The slave lattice relative improvement increased by about 2dB for this data, giving approximately 3dB better performance than the dual delay line MTI.)

In the slave lattice scheme, as λ is held fixed and the delay m increased, the residue clutter power increases as the reflection coefficients become "more out of date". For clutter suppression the delay m should therefore be as small as possible. Setting m equal to 13 ensured that the target had reached its peak amplitude before any can-

cellation could start to occur. As m was held fixed and λ decreased to low values, tests showed that the clutter power actually increased. This is probably due to the master lattice responding to local perturbations in the data (at lower λ), which are no longer appropriate m samples later. A compromise λ of .97 was used in the example shown. Extensive tests using all types of clutter and target would be required to find the optimum values of m and λ.

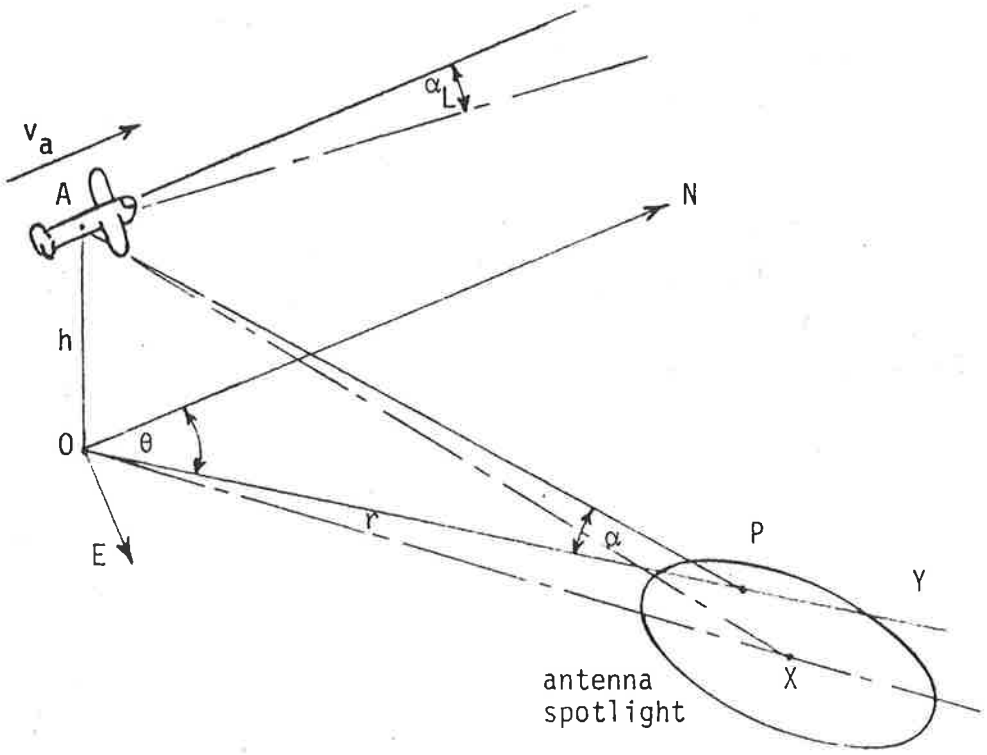
Following the investigations reported in the next section, a further scheme was developed for lattice MTIs. This technique has been only partially tested for the real data available, but would probably give performance at least equal to the dual delay line solution with stationary clutter. This idea is discussed towards the end of the following section.

7.2 Lattice MTI for Simulated Moving Radar

Clutter suppression for radars mounted on moving platforms may be more difficult than their stationary counterparts due to the large, and possibly variable, doppler shifts on clutter signals. The problem is worse when using a low PRF, as this gives a higher normalised clutter frequency. (The low PRF may be required, for example, to avoid range ambiguities). Techniques using extensions of the existing conventional methods have been developed for these situations, and are sometimes called Airborne Moving Target Indicators (AMTI). These methods, (e.g. Time Averaged Clutter Coherent Airborne Radar, TACCAR, see [69] chapter 17), generally use a clutter rejection filter of fixed shape, such as the dual delay line canceller, for which the notch is arranged to lie at the clutter peak frequency. This may be done by adjusting the local oscillator frequency so that the demodulated radar return spectra has a maximum at 0 Hz.

Apart from the fixed shape clutter spectrum the above techniques assume, they suffer from the frequency control requirement. If an open loop control system is used the velocity and azimuth indicators must be very accurate to avoid frequency errors; in a closed loop, automatic frequency control approach, noisy or weak clutter signals may

Figure 7.4a AMTI simulation geometry



cause problems. Additionally, such systems can only cancel one clutter peak, when in some situations more than one could be present. The lattice MTI would seem to be a candidate for AMTI applications. Its transfer function is adjusted adaptively and may track changes in the clutter location according to the best minimisation of the prediction error power. Some differences exist, however, in how the lattice must operate in this case compared to the last (standard MTI). These will be explained after the following model has been described.

The simulation model is shown in Figures 7.4a and 7.4b. An aircraft is assumed to be flying North at height h above the earth and constant speed v_a m/sec. Its antenna is scanning from East to North at fixed angle α_L below the horizon. At the instant shown the antenna's boresight is pointing towards X, at bearing θ_L from North. The surface below the radar is assumed to have uniform reflectivity, from any incidence direction.

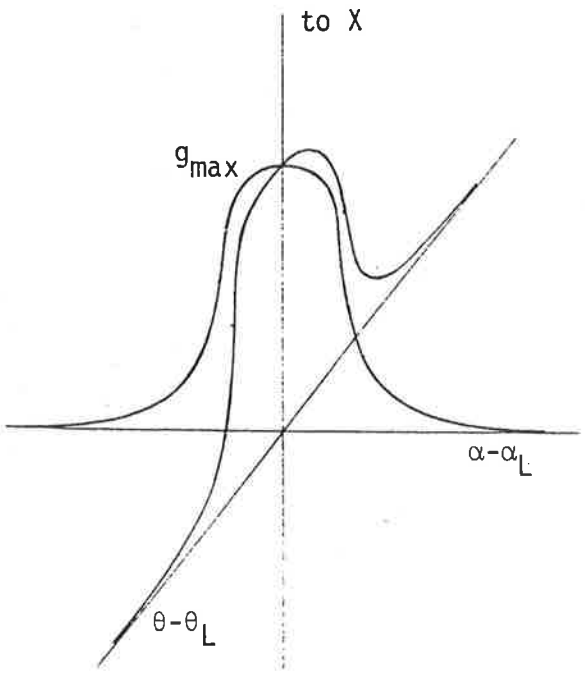


Figure 7.4b
Gaussian antenna model
(see (7.7))

To determine the clutter power spectrum consider the signal received from point P, at bearing θ and angle α below the horizon. The surface distance to P is r (flat earth assumed). The relative velocity towards A is

$$v_r = v_a \cos\alpha \cos\theta \tag{7.4}$$

and the doppler frequency of this signal is

$$f_d = \frac{2f_0}{c} v_r \tag{7.5}$$

where c is the speed of light and f_0 is the radar frequency. Points on the surface satisfying $\cos(\alpha)\cos(\theta) =$ constant therefore produce the same doppler shift and contribute to one element of the clutter power spectrum. Within the spotbeam region these points almost lie on the straight line OY, and to simplify the analysis this will be assumed to be the case. As α is always a small angle this is a good approximation provided θ does not also become too small. This restriction does not impair the simulation because the worst (most spread out) clutter spectrum occurs when θ is not small, when traditionally AMTI performance is worst.

The total clutter power (from all ranges) at given frequency f_d may be found from the radar equation by

$$P(f_d, \alpha_L, \theta_L) = k \int_0^\infty \frac{g(\theta, \theta_L, \alpha, \alpha_L)}{(r')^4} dr \tag{7.6}$$

where k is a constant, r' is the radial distance AP $= (r^2 + h^2)^{0.5}$, and g is the antenna power gain, a function of the directions of the boresight (AX) and the elements under consideration (AP). To model the antenna pattern assume a Gaussian distribution, as shown in Figure 7.4b.

$$g = g_{max} e^{-((\alpha-\alpha_L)^2+(\theta-\theta_L)^2)/2\sigma^2} \tag{7.7a}$$

The standard deviation σ is related to the 3dB beamwidth (b) by

$$\sigma = \frac{b}{2(2\ln 2)^{0.5}} \tag{7.7b}$$

Since α is a small angle, further assume that

$$\alpha = \frac{h}{r} \tag{7.8}$$

and that in (7.6) $r' = r$. Then

$$P(f_d, \alpha_L, \theta_L) = k g_{max} e^{-(\theta-\theta_L)^2/2\sigma^2} \int_0^\infty e^{-(\frac{h}{r}-\alpha_L)^2/2\sigma^2} r^{-4} dr \tag{7.9}$$

where θ is a function of f_d given by (7.4) and (7.5).

With the assumption of constant look angle α_L during one scan, the above integral need not be evaluated to obtain relative clutter power spectra at various θ_L. (In fact this definite integral may be solved using a degenerate hypergeometric function, see [70], 3.462 (1), 9.247 (1) and 9.254 (1), (2); however this does not help in determining the clutter power from individual range rings.) For convenience let w be the ratio of maximum doppler frequency to the aliasing frequency f_a (= PRF/2).

$$w = \frac{2 f_0 v_a}{c f_a} \tag{7.10}$$

Then the relative clutter power for look direction θ_L is (neglecting the $\cos\alpha$ term in (7.4) according to the straight line approximation)

$$P_{\theta_L}(f_d) \approx e^{-(\arccos(f_d/wf_a)-\theta_L)^2/2\sigma^2} \tag{7.11}$$

With the clutter modelled as a function of bearing θ_L, a (non stationary) Gaussian distributed time series with appropriate power spectrum, changing according to the scan rate of the antenna, was required to simulate the clutter signal. This was achieved as follows. At various angles θ_L the relative clutter power spectrum was evaluated from (7.11) and an inverse Fourier transform used to determine the autocorrelation function. A Levinson-Durbin algorithm (see Chapter 3) then fitted a fourth order all-pole model to the autocorrelation function, and the calculated LPCs were used to filter white Gaussian noise (using software described in Appendix B). This process was repeated for θ_L in approximately one degree increments, with the filter coefficients being linearly interpolated at each time step for intermediate angles. The synthesis filter gain was also adjusted continuously to ensure constant total clutter power.

The clutter signal thus generated represented the model parameters given in Table 7.2.

Table 7.2 AMTI Simulation Parameters

aircraft speed	v_a	50 m/sec (200 kph)
antenna beamwidth	b	2 degrees
radar PRF	$2f_a$	4.166 kHz
radar frequency	f_0	5 GHz (so w=.8 from (7.10))
vertical look angle	α_L	2 degrees (nominal)
horizontal look angle	θ_L	85.5-12.4 degrees
horizontal scan rate	$\frac{d\theta_L}{dt}$	34.7 deg./sec
		(120 samples/deg.)

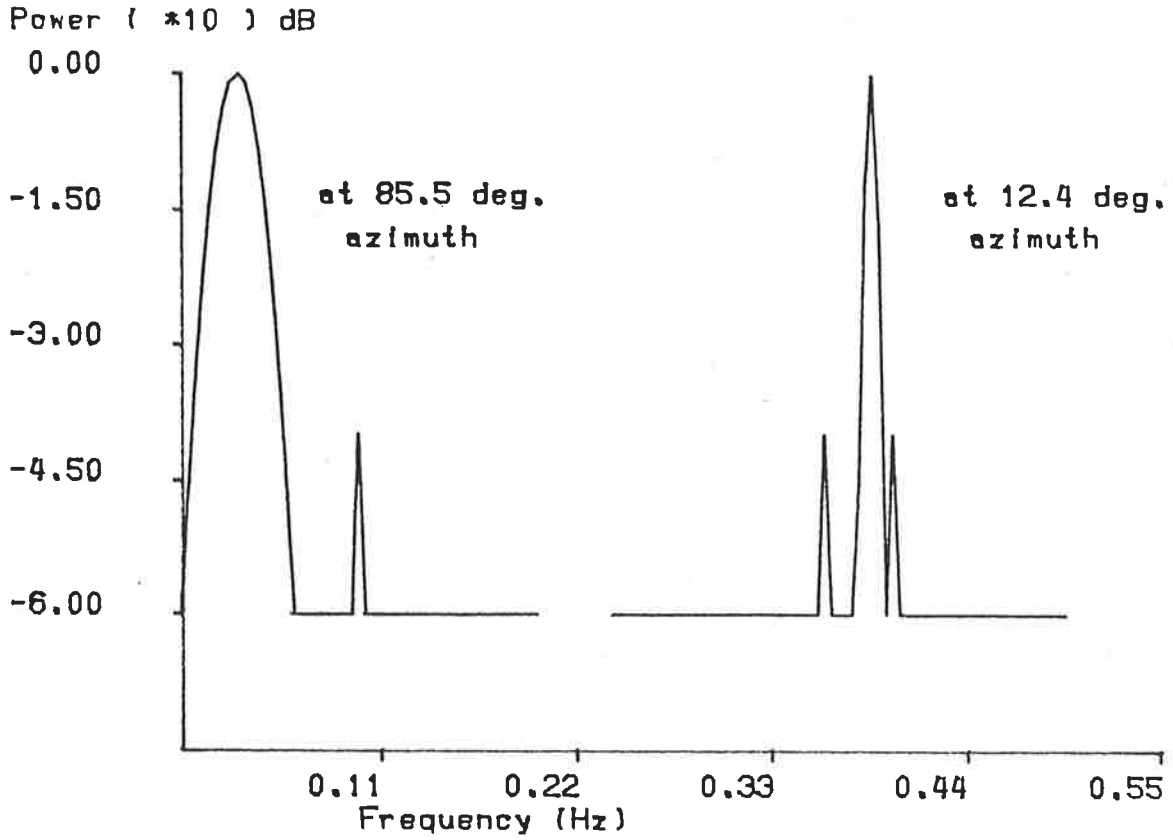
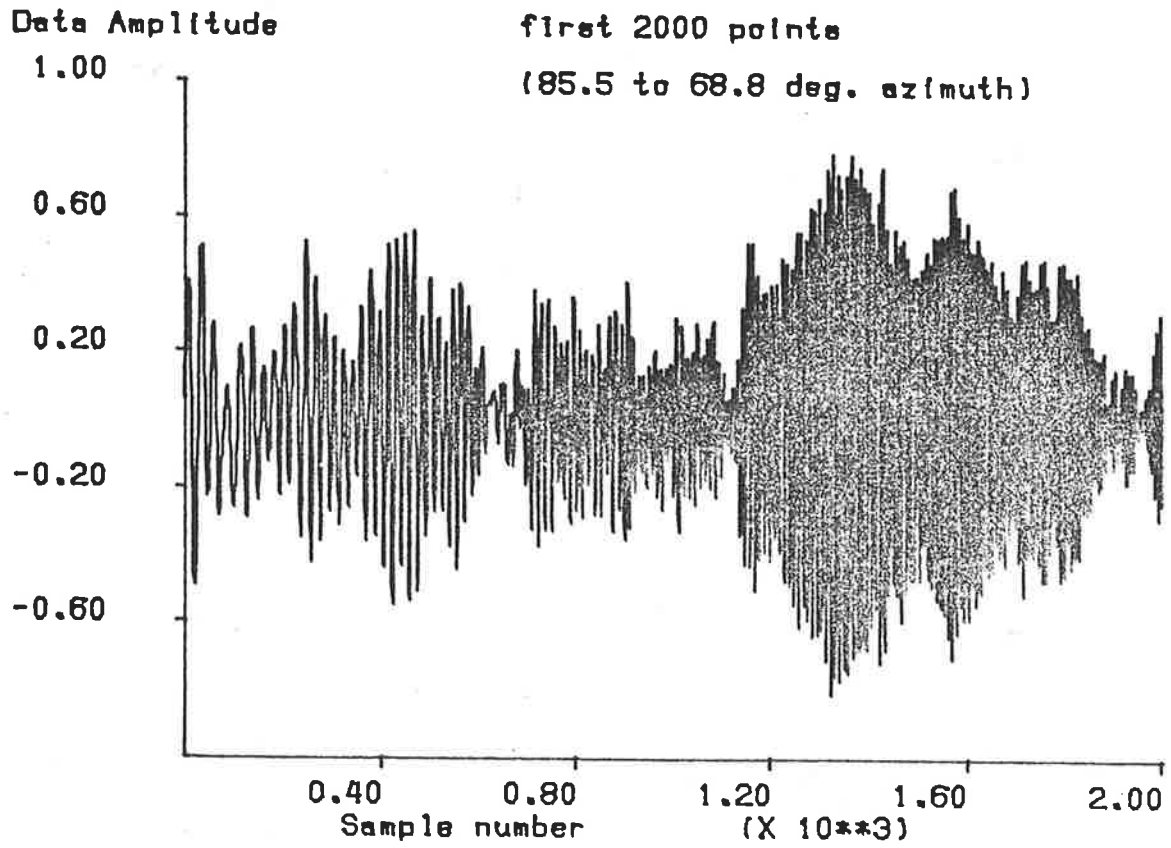


Figure 7.5 AMTI clutter power spectra

Figure 7.5 shows the model clutter power spectrum at the beginning and end of the horizontal scan. Again the frequency scale is normalised by the PRF. As mentioned above the minimum azimuth angle was restricted so that the straight line integral assumption remained valid. In practice the power spectrum for due North heading would be even sharper. The "sidelobes" and flat spectral floor visible in this figure were added to stabilize the synthesis process. They cause a more slowly varying synthesis filter gain. Figure 7.6 shows a realisation of the clutter signal using the model parameters given above, for approximately the first quarter of the sector scanned. "Targets" were superimposed on the clutter signal by adding a sinusoid whose envelope was determined according to the antenna gain (7.7a) and the scan rate.

The target signal duration relative to the period for which the clutter is locally stationary, indicates an important difference between this AMTI simulation and the

Figure 7.6 AMTI clutter time series



MTI data previously considered. In both cases the target duration may be considered as resulting from a convolution between the antenna gain pattern in azimuth, and a sinusoid of suitable amplitude and frequency. With the rapidly changing clutter statistics in the AMTI case there is a significant change in spectral characteristics over the space of the antenna beamwidth. As the lattice filter, at any one time, can model the clutter spectrum over only a relatively stationary period, its effective data window should cover a small azimuth sector, and thus be comparable with the target duration.

For these reasons the AMTI can not operate on the same principle of relative signal duration as used in lattices of the previous section. (In fact with small values of the relative doppler frequency w and narrow bandwidths b, the same type of operation could be employed, but since this problem has been considered it will be assumed this state does not exist. Conversely, in standard MTI applications the lattice MTI might

degrade when w is large.) The slave lattice could only be employed if the delayed filter coefficients were updated to compensate for the peak shift, which would be difficult in all but the two stage case.

A method of achieving successful lattice operation in these circumstances is simply to use a filter of suitably low order. The "minimum order clutter modelling" approach works as follows. As already stated the stationary clutter signal may be well modelled by a low order all-pole model. This is also true of the shifted clutter spectrum, where a higher order will be required since conjugate pole pairs are being modelled. The Levinson-Durbin four pole model described above leads to a worst case residue error 35 dB below the clutter power for the 2 degree beamwidth. (For $b = 6$ degrees it was 27 dB down.) Assuming the model order is known, a target of disparate frequency must remain uncancelled or, if large enough, remove two poles from the clutter model, thus shifting one conjugate pole pair in the z plane. Either of these possibilities will cause an error power increase and so provide a means of target detection.

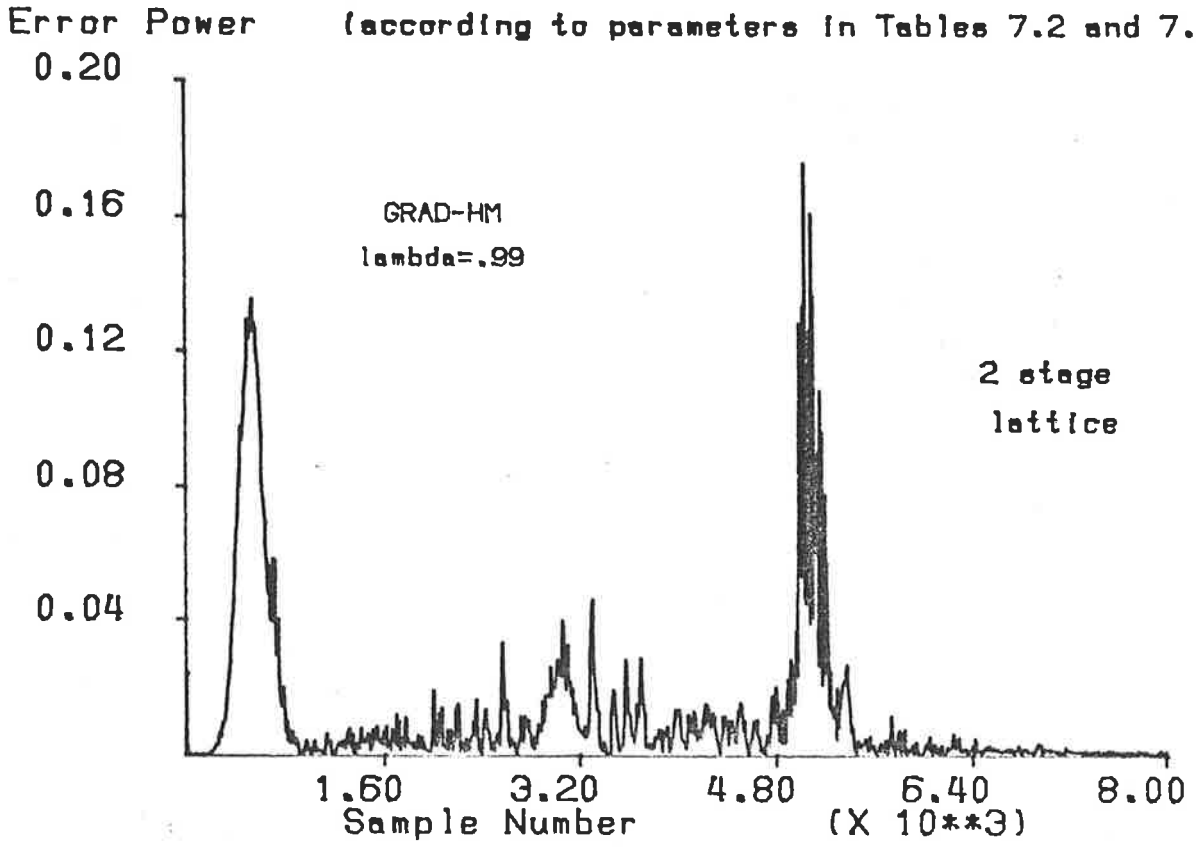
To illustrate this technique targets were added to the clutter signal partly shown in Figure 7.6, according to the following list:

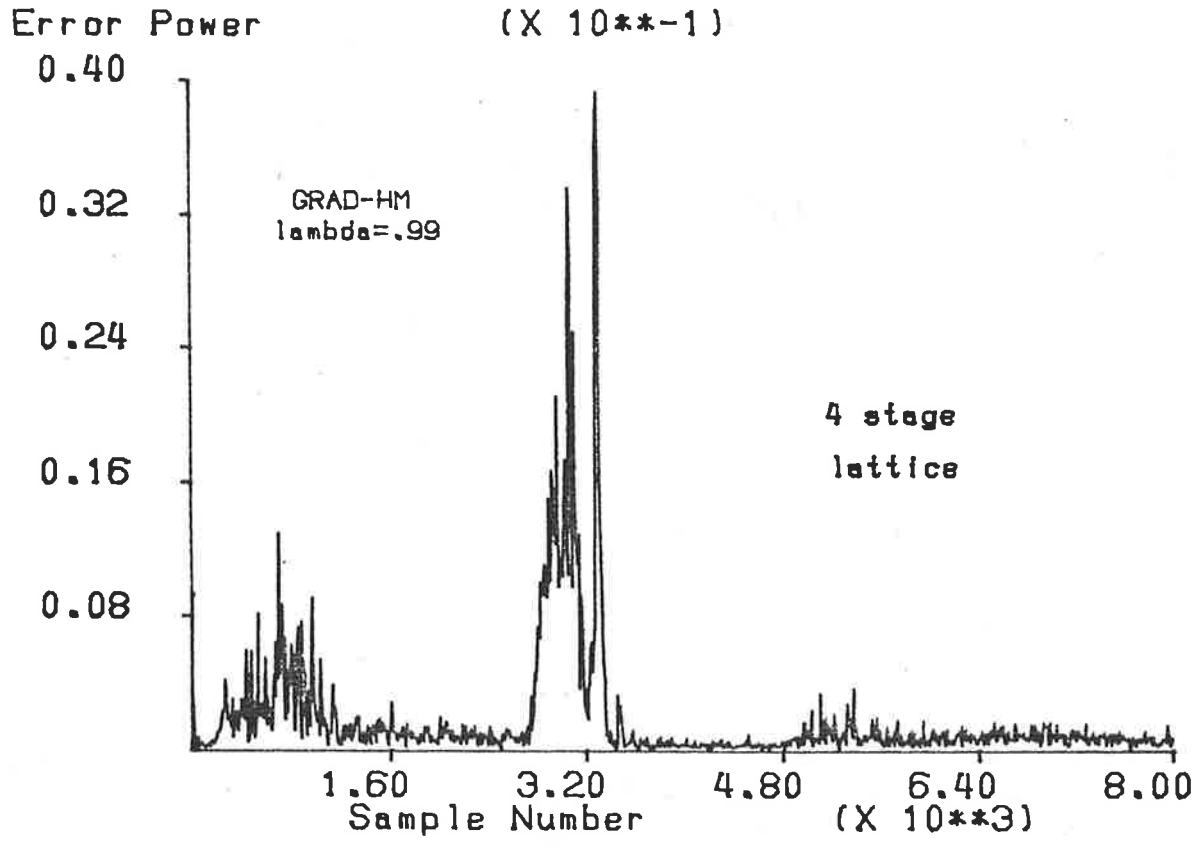
Table 7.3 AMTI Test: Target Characteristics

target	peak target/ mean clutter power	target peak location		Frequencies (normalised)		
		t	θ	target	clutter	difference
1	-12dB	550	80.1	.25	.07	.18
2	-24dB	3050	60.1	.45	.20	.25
3	-3dB	5050	43.4	.25	.30	.05

The last target is large but located very close to the clutter peak. The resulting signal was filtered by 2 and 4 stage GRAD-HM. The error powers, averaged over 10 sam-

Figure 7.7 Lattice output for AMTI signal





ples, are shown in Figure 7.7. The 2 stage lattice was adequate for clutter cancellation during the early and later parts of the time series and so produced a small error output. (Only two stages were needed because at low frequencies the broad clutter spectrum could be modelled with real poles only, while at high frequencies the clutter spectrum was much sharper and therefore needed only one conjugate pole pair.) The targets in this region stand out well, while the second target is obscured by the excess clutter prediction error. The 4 stage lattice, on the other hand, was able to completely cancel the third strong target where it has spare stages, but the second target was detected.

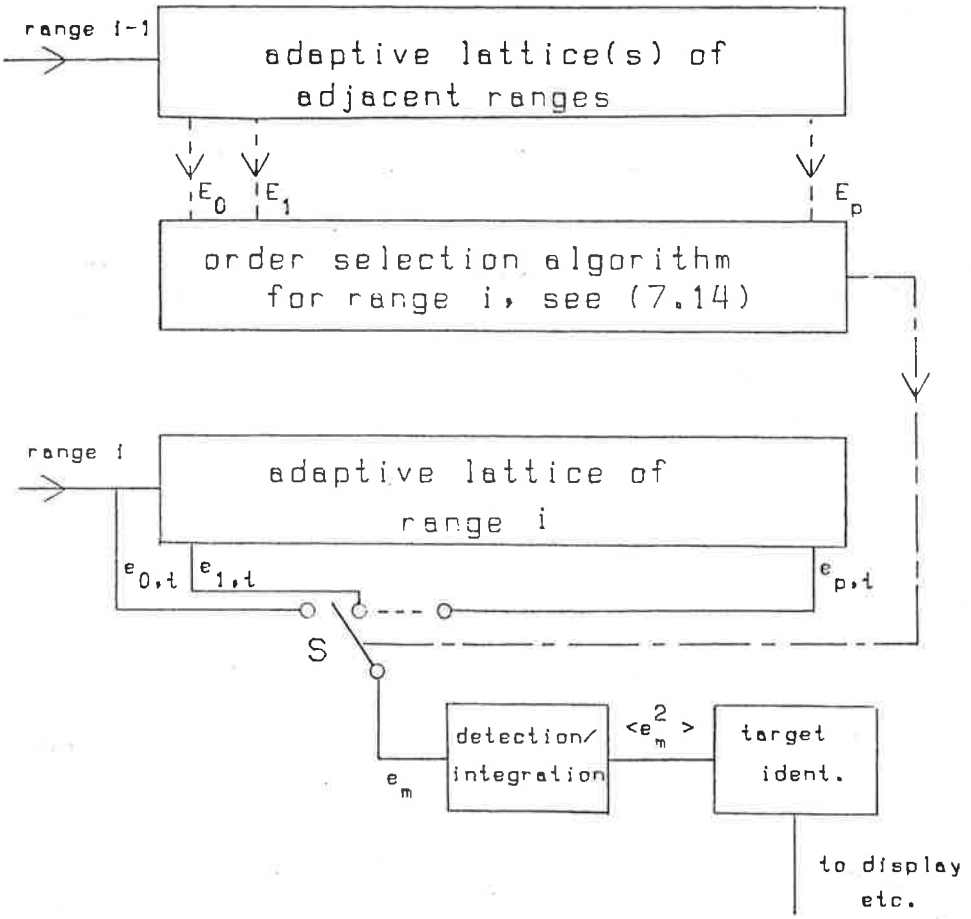
The success of this method is dependent on knowing the correct model order at any time. In addition this must be the clutter model order, not the composite signal (target plus clutter) model order. The ratios of second to fourth stage error powers, for the clutter signal in this simulation, averaged over 1000 samples, were

time	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	(thousands)
error ratio	2.1	3.8	8.2	17.3	18.8	5.1	2.1	1.1	

By choosing an appropriate threshold for this ratio, below or above which the two or four stage error output respectively is used, all targets in the example could be detected. In practice the clutter model order could be determined by examining the error powers of adjacent range rings, thereby avoiding unwanted effects due to targets at the current range. This is depicted in Figure 7.8. In the absence of clutter altogether, the controller would naturally select the zero-th order error.

Although the simulation gives promising results, this method should be tested on real AMTI data to confirm its usefulness. The assumptions concerning uniform reflectivity and Gaussian antenna pattern would be somewhat different from reality, at least over a land surface. This could lead to a higher order model being required. The order selection and approximate performance analysis could be on the following basis, which simply uses the fact that the adaptive lattice filter adjusts its coefficients to minimise the output error power stage by stage. Assuming a sinusoidal target return,

Figure 7.8 AMTI lattice using minimum order clutter model



whose frequency does not significantly overlap the clutter spectrum, the target will capture a conjugate pole pair from the $m-$th order clutter model if

case 1

$$P_T > E_{m-2} - E_m \tag{7.12}$$

where E_i is the $i-$th stage mean clutter power, P_T is the target power (m is the stage whose forward error is currently being selected by switch S in Figure 7.8). Then, because the (deterministic) target will be completely cancelled, the mean square output from

stage m will become E_{m-2}. Assume that for reliable target detection, the ratio

$$R = \frac{< e_m^2 > \text{ with target}}{< e_m^2 > \text{ without target}} \tag{7.13}$$

must be greater than, or equal to, R_{min}. The value of R_{min} will be a function of the error statistics and the degree of error integration. If (7.12) is true

$$R = \frac{E_{m-2}}{E_m}$$

thus m may be chosen according to

$$\text{maximum } m \text{ for which } \frac{E_{m-2}}{E_m} \geq R_{min} \tag{7.14}$$

The clutter error power ratio would be evaluated on adjacent range ring(s), from $m = 2, \ldots$ and the largest value used. (If none satisfy (7.14) use $m = 1$). The lattice algorithms contain error covariance variables which could be used for this purpose (e.g. $w_{n,t}$ in GRAD-HM, see (3.10b)). If the Sub-Clutter Visibility (SCV) is defined by

$$SCV = \frac{\text{clutter power}}{\text{detectable target power}} \tag{7.15}$$

then from the above

$$SCV_1 = \frac{E_0}{E_{m-2} - E_m} \tag{7.16a}$$

Smaller targets might also be detected under some conditions. With m again chosen according to (7.14) and $P_t < E_{m-2} - E_m$, then

case 2

$$R = \frac{E_m + gP_T}{E_m}$$

where g is the power gain of the lattice at target frequency (equal to the reciprocal of the all-pole spectral estimate defined in (2.10)), approximately unity or slightly less. With $R > R_{min}$ this gives

$$SCV_2 = \frac{gE_0}{E_m(R_{min} - 1)} \tag{7.16b}$$

The case 2 situation would be more likely when the ratio E_{m-2}/E_m is large. To derive an approximate general result for the SCV, if the E_{m-2} over E_m ratio is assumed to be close to R_{min} in case 1, and g is approximately unity in case 2, then combining cases 1 and 2:

$$SCV \approx \frac{E_0}{E_m(R_{min}-1)} \tag{7.16c}$$

The clutter modelling approach could also be used on non-airborne (standard) MTI. Less error averaging would be possible because of the smaller number of hits per target, and so R_{min} would need to be larger than above. Offsetting this fewer poles would be required (at least for stationary clutter), so removing two would have greater effect. To indicate possible performance some typical clutter (from range ring 4 in Figure 7.1) was adaptively filtered giving the following clutter prediction error powers. These are expressed as a percentage of the input clutter power E_0.

	E_1	E_2	E_3
GRAD-HM $\lambda = .8$	11.8%	6.9%	5.2%
GRAD-F+B $\lambda = .8$	9.5%	4.3%	2.8%
dual delay line canceller	-	3.0%	

Of initial interest here is the significant improvement of the two reflection coefficient (per stage) lattice over GRAD-HM. This is due to the small effective data window allowing GRAD-F+B to respond better to short term fluctuations in the data. The Exact Least Squares lattice ELS-UP (see Chapter 4) was also tested and was equal to the second gradient lattice at the significance shown above, which reinforces the observations of section 4.4. For MTI operation either a (typically) two or three stage GRAD-F+B lattice could be used, and its performance would be about the same as the conventional canceller for stationary clutter. To avoid cancelling targets the ratios of error powers in adjacent range rings could again be used to control the actual filter length.

Target Classification

Target classification is sometimes possible with relatively low resolution tracking

radars by examining the radar return for changes in amplitude, polarisation or doppler shifts caused by target shape and movement. Knowledge of possible target geometries and dynamics may enable identification of a particular radar return, or passively received signal, from a number of alternatives. As Casper says [69]

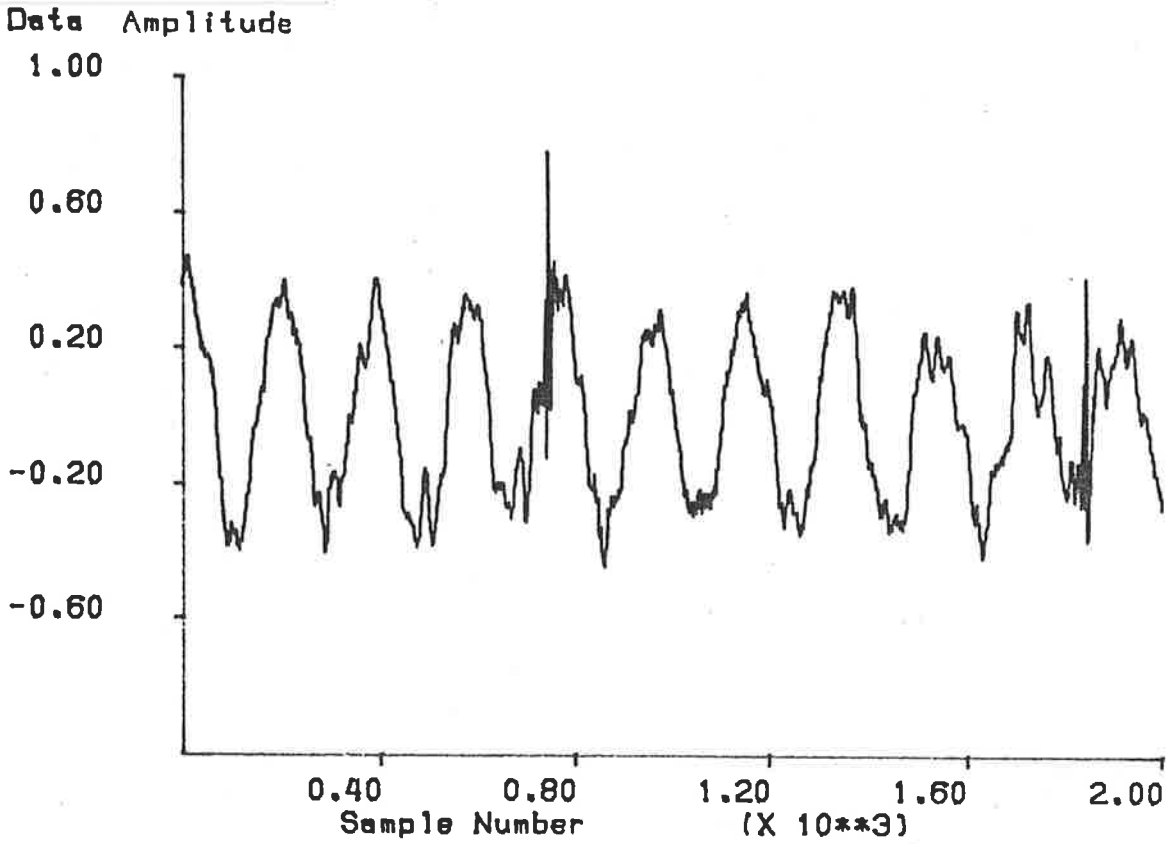
> "Because of interference, targets moving relative to the radar possess widely fluctuating returns if the targets consist of a number of scattering points. Asymmetric objects which tumble have essentially periodic modulation fluctuations. Propeller-driven aircraft and helicopter returns possess periodic modulation useful for classification."

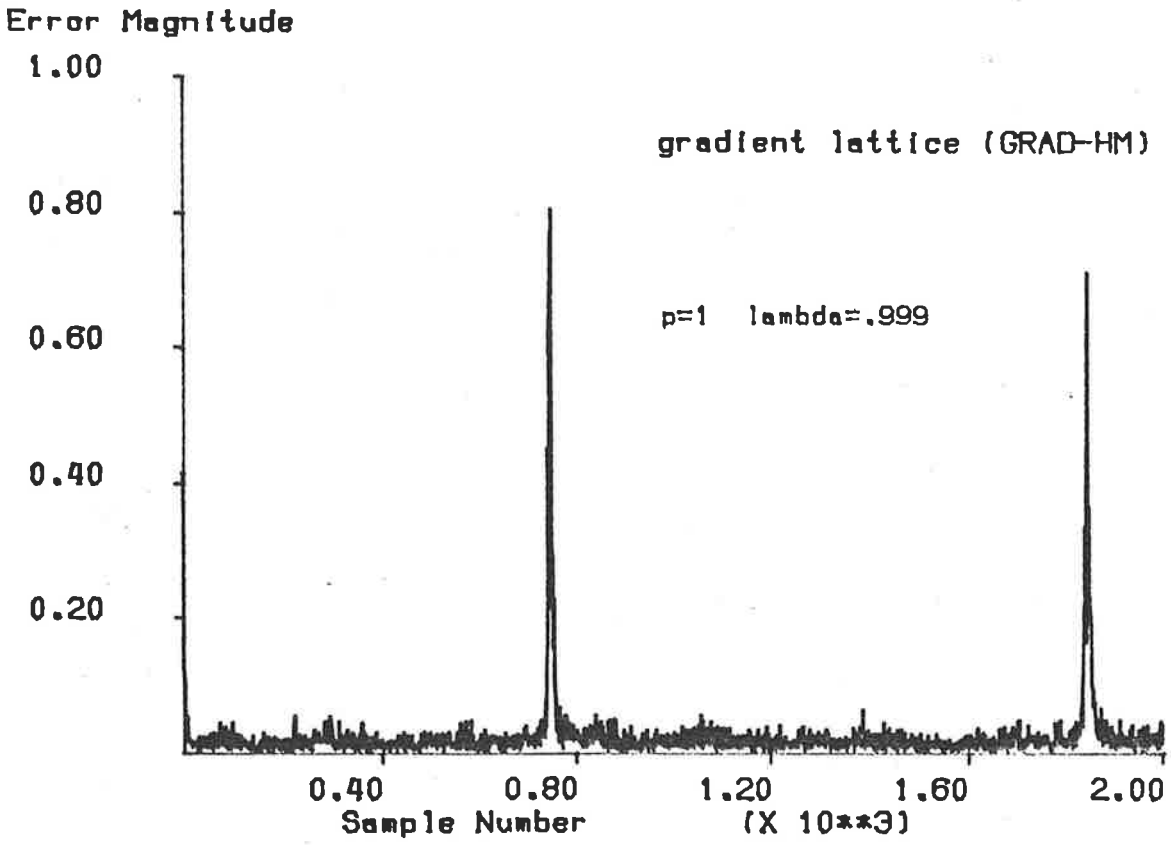
Studies have examined the doppler spectra of aircraft, for example [48], and been able to relate spectral characteristics to target components, such as compressor blades.

When the return signal contains some components which may be predicted as a linear combination of the past p samples, an adaptive filter may be used to separate these components. The doppler shift due to a target's relative motion with respect to the radar may be cancelled in this manner. If moving parts of the target sometimes present an almost specular reflection, these "flashes" might thus be seen in the error output, even though they would be otherwise difficult to identify. Figure 7.9 illustrates such a target return and the error output from a single stage adaptive lattice. In this case the flash rate is periodic and so might lead to target classification. Only one stage was required here because the doppler frequency was so low. This type of application is similar to the first type of MTI operation discussed, and has parallels with some medical and seismic applications mentioned in Chapter 1. The method works best when the non-deterministic components have very sudden onset. The Exact Least Squares lattice gain terms (γ^c, see Chapter 4) have been observed to be extremely sensitive to small but abrupt changes in the input statistics in some early simulations for this chapter.

Figure 7.9

Target classification example





Implementation Considerations

Computational, control and storage requirements of the above adaptive lattice filters could be quite easily met due to recent advances in VLSI technology. For example the "TFB" signal processing chip (see Appendix E), on present design specifications could perform five stage (complex) adaptive lattice filtering (using GRAD-HM), for each range ring at PRFs of at least 10 kHz. A number of these chips, plus a fast microprocessor controller, could be used to implement the lattice AMTI. This would replace a considerable amount of analogue circuitry required in existing AMTI radars. At low PRFs, such as used in standard MTI, one signal processor chip could process data from many range rings. Alternatively at very high PRFs, possible in target classification applications, each chip could implement one stage of an adaptive lattice. Data acquisition requirements for these applications would be quite modest as 10 or 12 bit accuracy could be used.

Use of the All-Pole Spectral Estimate

Another approach to some of the radar applications considered above would be to use the all-pole spectral estimate in place of, or in conjunction with, the error output. Haykin [14] has suggested identifying different types of radar clutter by MEM spectral estimation. An alternative lattice AMTI could use a gradient lattice of perhaps 6 to 8 stages and periodically dump the reflection coefficients to the control processor. This would calculate the LPCs (via the whitening filters of section 4.3), then the all-pole spectral estimate from (2.10) and finally examine spectral peaks to identify target peaks. Chapter 5 gives some insight into the statistics of these spectral estimates. Using the previous AMTI model parameters it seems (using (5.27d)), that the clutter peak would have a fairly high normalised variance, although other parts of the spectrum could be acceptable. The radar PRF and scan rates might be adjusted to suit the spectral stability requirements. While the frequency domain approach would require more processing, greater information could be obtained (such as a target's radial velocity

or the presence of multiple targets). Greater Sub-Clutter Visibility might be possible with this method for clutter signals not as well modelled with a small number of poles. In target classification applications all-pole spectral estimates could be used in conjunction with error output information to examine specific components of the data sequence.

This chapter has briefly examined some applications of adaptive lattice filters to a specific field. A surprising number of the analytical and simulation results from earlier chapters were found useful. More tests on a comprehensive range of real data would obviously be useful to confirm the promising applications discussed.

Chapter 8 Conclusions

The following material summarises the findings from previous chapters under a number of headings. These headings may not necessarily be in the same order as the preceding chapters. In addition some comments are included regarding areas of interest to adaptive filtering not covered in this thesis but worthy of further investigation.

Comparison of Adaptive Algorithms

Chapters 3, 4, 5 and 7 contain a number of comparisons between different adaptive algorithms. Little time has been spent in comparing the simple LMS algorithm of (2.7) and (2.8) with lattice algorithms, as the advantages of the latter are well documented, (e.g. [9,12]). Figure 3.3 presents one comparison showing mean square error output. These comments are not intended to deride the LMS algorithm since its computational requirements are much lower than alternatives. The first group of operation counts (arithmetic operations per sample per stage) in the table at the end of Chapter 4 must be compared to two multiplications, no divisions and 2 additions for the LMS algorithm. Thus the LMS alternative would be the preferred choice whenever possible; unfortunately for many applications involving nonstationarity, limited data or signals of high dynamic range it is not suitable.

The lattice structure owes its better performance to the inherent orthogonalisation and decoupling properties. These result in the backward errors being uncorrelated, and each stage adjusting independently of later stages. Chapter 3 attempts to show how the gradient lattice algorithms are derived from a common ancestry of expected error squared minimisation. A similar (but necessarily more summary) examination of the ELS algorithms shows them to be derived from a minimisation of the error squared over given data. Different assumptions (required due to finite amounts of data) during this minimisation give rise to either prewindowed or covariance ELS lattices. Despite the considerable differences in derivation basis and complexity between the gradient and

ELS approaches, two of these algorithms, GRAD-F+B and ELS-UP, are the same but for some gain terms, the values of which are generally close to unity (see section 4.4).

Since GRAD-F+B is much closer to the optimum (ELS) prewindowed lattice than GRAD-HM, it is not surprising in hindsight that whenever significant differences are encountered between gradient algorithms, the GRAD-F+B performs better. This is most notable in the impulse response tests of Figures 4.5 to 4.7, the all-pole spectral estimate distribution test of Figure 5.4 and the clutter suppression test using real data in section 7.2. The superiority of the GRAD-F+B has not been widely reported beforehand. This might have been due to fears about the possibility of its reflection coefficients exceeding one in magnitude. This behaviour was certainly observed for short periods during simulations but did not cause instability at any time. The unexpected slightly lower transient MSE from GRAD-F+B compared to ELS-UP might be related to this lack of constraint. Further investigations on these topics would be useful.

In many simulations all of the lattices tested showed similar performance. These situations were typified by an exponential weighting factor close to one (and therefore long effective data window) and Gaussian distributed random data, while the performance criteria were usually mean filter parameters (reflection or linear prediction coefficients) or mean square error output. The most important of these conditions is the first. Especially when λ is small, or at startup, significant performance differences between the algorithms considered could be expected. General conclusions about the "best" lattice algorithms are not possible because in each case the algorithms with superior performance involve a computational and control penalty (not considering special architectures, see section 4.4). It may be noted, however, that the computational increase is not large, so in situations requiring best performance the ELS class of algorithms would usually be chosen.

Before leaving the discussion concerning various adaptive algorithms, the Fast Kalman alternative [41] should be mentioned again. This was described briefly in section

4.1 but has not been explicitly stated or used in simulations. It may be considered as an "exact least squares" algorithm for the transversal linear predictor since it is mathematically equivalent to the Kalman filter (when applied to the linear prediction problem), but requires slightly fewer operations than the ELS unnormalised lattice algorithms. Because the Fast Kalman algorithm produces an optimum least square solution the results concerning convergence and all-pole spectral estimation would also be applicable. It has been suggested that because the Fast Kalman does not possess the stage-by-stage minimisation of the lattices, its actual performance in finite precision implementations might be inferior [44]. There appear to be few results available to confirm or refute this, and it would be a useful area for investigation. (For some preliminary results on finite word lengths in adaptive algorithms, though not Fast Kalman, see [78,79].)

Lattice Convergence

Section 3.3 discusses lattice convergence, both single and multiple stage. The former provides useful insights into factors effecting convergence, but is not easily extended to subsequent stages. The observations regarding the effect of unused stages on MSE convergence are shown to have practical implications in Chapter 7.

The approach taken for multistage convergence was to examine the optimum least square solution at different times according to the exponentially weighted error minimisation (3.26a), allowing prediction of lattice coefficients and error performance under given conditions. Simple solutions using the Levinson recursion on a "composite autocorrelation function" were possible when the latter could be evaluated. This was done for two useful cases: abrupt (known) change in the input's autocorrelation function, and "smooth" changes modelled for each lag as a polynomial with known coefficients. Good agreement with simulations was shown, which was certainly expected for the ELS algorithms and perhaps not surprising for the gradient solutions in view of the reasonable data window lengths. The AMTI lattice simulation of Chapter 7 could be extended to use this method to predict lattice error power response for nonstationary clutter

signals. This would avoid the need to generate time series of specified nonstationary power spectrum.

All-pole Spectral Estimates

Appendix A provides results on the (relative) all-pole spectral estimate, $(Q(\omega))$, bias and variance from adaptive transversal filters using the LMS algorithm. These are summarised in section 2.4. The derivation assumes that the adaptive coefficients are unbiased and their noise components uncorrelated, which are reasonable assumptions when the adaption constant μ is small enough. Chapter 5 derives similar (but probably more useful) results for spectral estimates derived from exponentially weighted adaptive lattices. The approach is more general yet simpler than that previously used. It is assumed that the LPC estimates' distribution will be equivalent to an optimum least squares solution. It is therefore strictly only relevant to ELS lattice algorithms, however as noted above, it is a good approximation for the gradient lattices, especially GRAD-F+B, in those cases where a significant number of independent data samples produce the estimate.

As explained in Chapter 5, the results for $Q(\omega)$ bias and variance represented in (5.9) ($\delta(\omega) << 1$) may be simplified to give (5.16). This simplification is a good approximation for most frequencies, but especially true at spectral peaks. Some of Akaike's earlier results on factorisation of the signal covariance matrix inverse may be employed (5.17) to simplify the calculation of $r(\omega)$ (and thus $\delta(\omega)$). (5.23) gives the relation between the exponential weighting factor for prewindowed lattices and the equivalent number of data samples in a rectangular window to obtain LPC estimates with the same statistics. Note that this is approximately twice the number of samples given by the usual time constant $(1-\lambda)^{-1}$ for exponentially weighted lattices. The latter actually applies to prediction error variances and partial correlations, and in some cases to reflection coefficients, as explained in Chapter 3.

Section 5.2 examines the $Q(\omega)$ distribution function when the condition $\delta(\omega) <<$

1 does not hold. The LPCs estimates are still assumed to have their asymptotic normal distribution. The predicted distribution function, when solved by numerical integration at a spectral peak frequency, gave reasonable agreements with simulated results for the example used. This involved a normally distributed signal of high spectral dynamic range, and used a small value of λ so that δ at the first peak was actually larger than 1.

The discussions of sections 4.3 and 5.1 concerning extraction of the predictor coefficients from lattices and their use in calculating $Q(\omega)$, explained that several alternatives are available, at least for two coefficients per stage lattices. Under stationary conditions it is probably best to use both the forward and backward coefficient estimates ("E" type in Chapter 5) to derive the most stable spectral estimates. As the effective data window length increases, the difference between the a_i and b_i decreases because each predictor is using a progressively higher proportion of common data. For a given window length the difference between predictors will be a function of model order and signal statistics. In practice the effect seems to be worth noting, otherwise gradient algorithms such as GRAD-HM may provide lower variance spectral estimates than ELS lattices using (say) only forward predictor estimates (see example in section 5.1).

The work concerning all-pole spectral estimation has been directed towards real-time adaptive predictor applications. If the only goal is to provide periodic all-pole spectral estimates of a relatively stationary signal, a simple autocorrelation function estimate could be used, followed by Levinson recursion and $Q(\omega\ )$ calculation. The results of Chapter 5 may still be used in this case to estimate variance etc. In other situations an adaptive filter might be required for its whitening properties (i.e. to make use of the error output(s)) and the spectral estimate could also be useful (e.g. some of the radar applications suggested in Chapter 7). Also the ELS lattices can provide prewindowed and covariance estimates of forwards and backwards LPCs, which might be preferred over the "autocorrelation method" estimates just described. This might, for example, enable higher resolution spectral estimates.

To summarise ways in which some of these results could be used in a practical situation, imagine that an adaptive exponentially weighted lattice is whitening a time series of unknown and slowly varying statistics. (The data is assumed to be reasonably represented by an all-pole model of either known order, or determined by one of the standard methods [1,23].) LPC and all-pole spectral estimates are calculated periodically, and some estimate of the $Q(\omega)$ stability is required. First the effective value of N could be found from (5.23). Then (5.18), (5.20), (5.19a) and (5.9b) could be used to estimate δ, the normalised transfer function variance, at the frequencies of interest (e.g. peaks). If δ was much less than 1, the bias and variance could be estimated with (5.16) or (5.9). (The former equation will give a maximum variance under-estimate of a factor of two, at frequencies removed from peaks.) If these results can not be employed, λ could be changed to reduce δ. When this action was not possible due to signal nonstationarity considerations, the numerical integration method could be employed to estimate spectral peak confidence levels. The expression for the $Q(\omega_m)$ distribution function (5.38) uses conveniently normalised variables which would allow a family of results to be precalculated. Each member of this family would be determined for one value of δ. Finally, various simplifications to the above are possible for second order models and these are explained in Chapter 5.

Radar Applications

Chapter 7 examines how adaptive lattice filters might be used in radar systems to achieve clutter suppression and target recognition. The former application has been studied by Haykin et al; the present work points out various difficulties not previously considered and how they might be overcome, together with some extensions. A method of clutter signal modelling is proposed for the Airborne Moving Target Indicator problem, and shown to work well in a simulation. A method of selecting model order (7.14) and estimating approximate detection performance (7.16c) is suggested. This scheme could be used to detect short-term sinusoid-like signals in the presence of a large in-

terfering signal of unknown statistics, whenever the latter may be well modelled by an all-pole model of known low order and is relatively stationary compared to the sinusoid duration. (If the signal to be detected has a sudden onset the problem is much easier and adaptive filters have been used frequently in this "event detection" role [15,16,17,39].) Comments are also made regarding implementational aspects and the use of all-pole spectral estimation in the radar applications considered.

Adaptive Algorithm Realisation

As mentioned previously the adaptive algorithms described are well suited to real-time signal processing applications subject to hardware constraints. Advances in integrated circuit technology over the past decade have made more adaptive filtering applications possible, and this trend is likely to continue at an increasing rate. Some attempts at ELS normalised lattice integration have already been made, as noted in Chapter 4. These designs use a specialised architecture and arithmetic algorithms to implement the square root and other operations efficiently.

Notwithstanding these contributions, many applications will no doubt use general signal processors of the type described in Appendix E. This would be an ideal candidate for implementing many adaptive filtering algorithms at useful sampling rates, as shown by the comments in Chapter 7. Chapter 6 describes a hardware implementation of the LMS tranversal filter. This was an interesting exercise in VLSI design and adaptive algorithm realisation, with a number of possibilities for expansion, as discussed.

Apart from the low-level implementations just mentioned, adaptive filtering algorithms will obviously be used on general purpose computers as well. Software used for many of the simulations from earlier chapters is described in Appendix B. Since these programs were not written specifically for the simulations shown, they have fairly general purpose application, and would be useful to anyone interested in this field.

The adaptive techniques discussed appear to show considerable promise for use

in many areas. Hopefully some of the results from this thesis will prove useful in this endeavour.

Appendix A

The following article concerns the statistics of all-pole spectral estimates derived from an adaptive transversal filter using the Widrow-Hoff LMS algorithm. It appeared in the IEEE Transactions on Acoustics, Speech and Signal Processing in April 1982. Section 2.4 provides a brief summary of the main results.

Davis, B.R. and Cowley, W.G. (1982) Bias and variance of spectral estimates from an all-pole digital filter.
IEEE Transactions on Acoustics, Speech and Signal Processing, v. 30 (2), pp. 322-329, April 1982

NOTE: This publication is included in the print copy of the thesis held in the University of Adelaide Library.

It is also available online to authorised users at:

http://dx.doi.org/10.1109/TASSP.1982.1163884

Appendix B Adaptive Filter Software

B.1 Introduction

A collection of computer programs is now described which allows simulation of various adaptive filters and comparison of the results with predicted responses. All of the adaptive algorithms investigated so far are included, plus some others. Software for signal generation, plotting of results using graphics facilities and various other utility routines are included. After a general introduction in this section, specific program descriptions are given in section B.2 and some examples of their use in section B.3. The software is written in Fortran-77 and has been run on a VAX 11/780 computer using the VMS operating system. Source code is readily available on magnetic tape.

The software may be broadly divided into two parts. One group of programs is concerned with using the adaptive algorithms on real or synthetic data and then displaying or analysing output such as estimates of coefficients, all-pole spectral estimates, MSE versus time etc. The main programs in this group are SIGS, FSIM and ESP. The second group of programs uses theory developed in the preceding chapters to predict adaptive filter responses, such as mean coefficient tracks or $Q(\omega)$ variance. Some of the main programs here are REVLEV, TESP and TRACKS. In many cases the predicted behaviour may be compared to that measured by simulations. Most programs store results on, or use input data from, formatted or unformatted data files. The latter type of files has been used where possible to reduce disk storage requirements.

The relationships between programs and their databases are depicted in Figure B.1. In general the flow of information is from left to right. For example a synthetic signal could be generated by program SIGS and used to test various adaptive algorithms with program FSIM. All-pole spectral estimates produced during this process would be stored on file "fsimop.dat", and could be displayed with program TEK10. In the figure the symbol "~" is used to indicate that data files may have a general name. Very briefly,

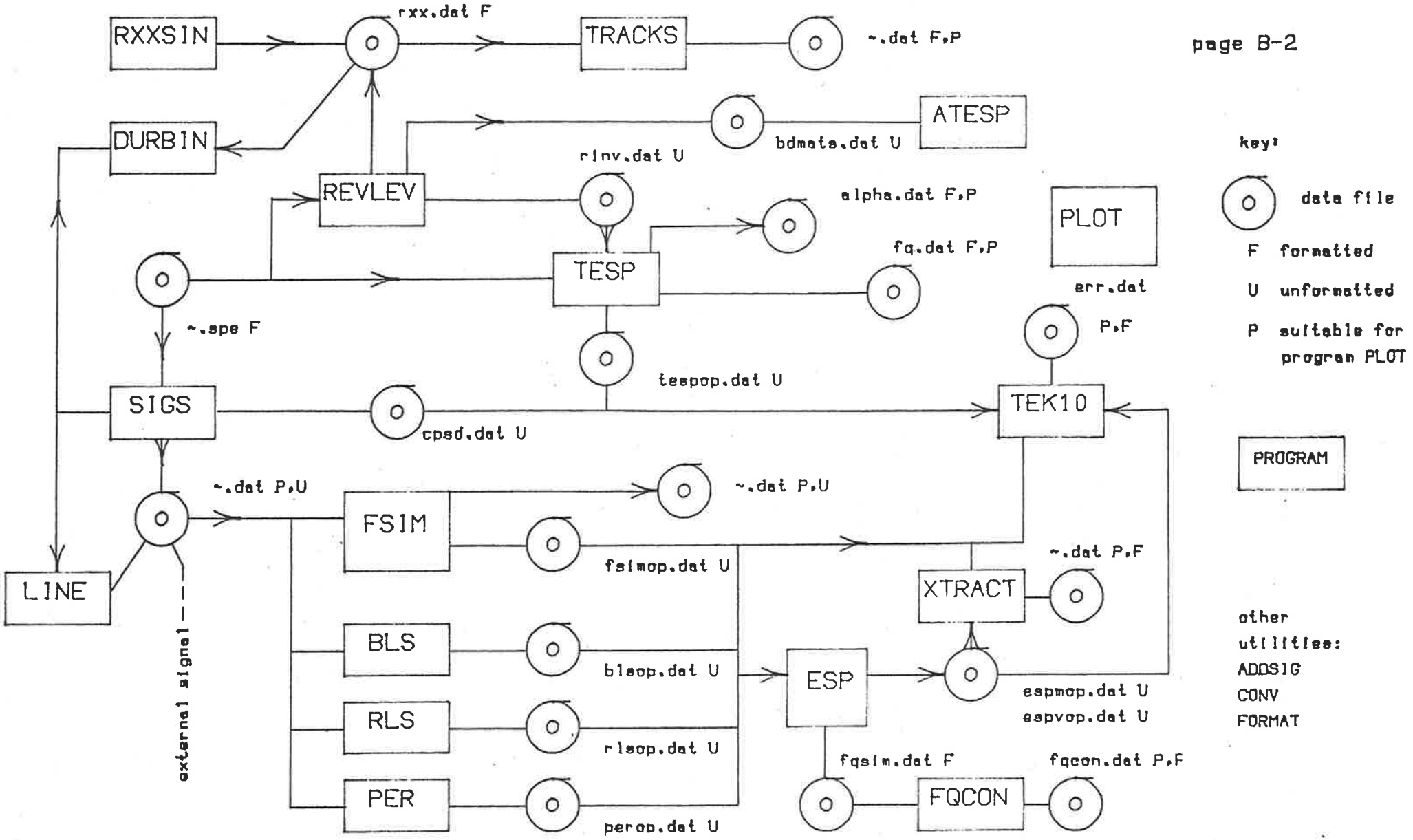


Figure B.1 Software organisation for adaptive filter investigations

the main programs have the following functions:

SIGS	signal generation
FSIM	adaptive filtering
ESP	statistical analysis of simulation results
REVLEV	autocorrelation function manipulation
TESP	prediction of all-pole spectral estimate statistics
TRACKS	convergence model from Chapter 3

There are two programs for displaying results on graphics terminals. Both of these use the Tektronics "PLOT10" software package, which is designed for Tektronics 4010, or look-alike, terminals. One program (PLOT) allows general plotting of a sequence of points on a connected-line graph such that the abscissa values are equally spaced. Many files shown in Figure B.1, and especially those not used as input for other programs, contain results suitable for program PLOT. The other program (TEK10) allows a sequence of spectral estimates to be displayed against time and frequency. Modified versions of these programs were used on a PDP 11/34 minicomputer with an augmented "PLOT10" library and National flat-bed plotter, to obtain hardcopy plots. Ascii data could be transferred between machines for plotting (VAX to PDP) or in the opposite direction in the case of real data.

To obtain ensemble averaging of results the VMS command file facilities were used. These allow a number of programs to be executed in sequence, as many times as required. By changing the random number seed each time around the command file loop different realisations of data could be generated; furthermore by renaming the result files all results could be retained for later analysis. An example of these "batch" runs is shown in section B.3 (see also ESP description in section B.2).

B.2 Program descriptions

The following descriptions include programs only for general adaptive filter in-

vestigations and not specific topics from Chapters 6 and 7. Some programs using block processing techniques are included for comparison. Program descriptions are arranged in alphabetical order. The entry under ARLIB describes subroutines which are stored in an object library and used by several programs.

ADDSIG

This program simply adds two unformatted (real-valued) data files, sample by sample, for as long as possible, and writes the results to new file.

ARLIB

ARLIB is an object library for VAX versions of adaptive filter subroutines. It is generated or updated using the VAX librarian. The library is composed from the following source files:

source file	subroutine(s)
arpsd.for	arpsd, fft, psdsum
group.for	group1, group2
graph.for	graph
qdisp.for	qdisp
qdisp2.for	qdisp2
range.for	range
grouf.for	grouf1, grouf2
hard.for	dummy hardon, hardof
invert.for	invert

brief description of modules (alphabetical order)

arpsd:	Calculates the all-pole spectral estimate from LPC estimates using zero-padded 256 point FFT to evaluate (2.10).
fft:	Complex valued FFT (written by B.R.Davis)
graph:	Graph plotting using augmented "plot10" software, originally written by R.C.Bryant but heavily modified. Includes axis labelling, scale marking, etc.
grouf1:	Version of "group1" for formatted files.
grouf2:	Version of "group2" for formatted files.
group1:	Reads first group of results from "fsimop" files.
group2:	Reads second group of results from "fsimop" files.
hardon:	Dummy version of "hardon" used on RSX system for hardcopy.
hardof:	Dummy version of "hardof" used on RSX system for hardcopy.
invert:	Matrix inversion by elementary row operations.
psdsum:	Prints summary of $Q(\omega)$ estimate: peak sizes and locations and Q minimum.
qdisp:	Q plotting in 3D (with hidden line removal), uses "graph".
qdisp2:	Version of "qdisp" without hidden line removal.
range:	Finds max. and min. values of data, used by "graph".

ATESP

This uses the alternative calculation of $r(\omega)$ and $s(\omega)$ in Chapter 5 by means of (5.19) and (5.20) (see descriptions of REVLEV and TESP). The program does not do the full variance calculation; it stops after showing the values of r and s, which may then be compared with those from TESP.

BLS

A block covariance solution is found for the best forward linear predictor by using (4.1) to (4.3). The sample covariance matrix is inverted by a direct method (routine

"invert" in ARLIB) rather than one of the "fast" methods. The results may be used to check the ELS-NC algorithm in FSIM.

notes: Link with ARLIB.

CONV

Converts data files: formatted to and from unformatted, integer valued to and from real valued (written by M. Gill, see also FORMAT).

DURBIN

This program uses the Levinson-Durbin algorithm (2.6) to solve the normal equations. The autocorrelation function r_0 to r_p may be either estimated from a data file or read from the file of autocorrelations ("rxx.dat") produced by REVLEV. In the former case a biased but positive semi-definite estimator is used ([22] section 3.6.4)

$$r_i = \frac{1}{n}\sum_{t=1}^{n-i} x_t x_{t+i}, \qquad i = 1, \ldots p$$

where r_i is the autocorrelation estimate at the i−th lag.

ESP

The name of this program stands for "Estimators' Statistical Performance". Its function is to examine simulation results generated by the main program FSIM over a large number of runs, each of which is stored in an "fsimop" file, and compile the following statistics:

1) MSE versus iteration number (i.e. the learning curve)

2) mean $Q(\omega)$

3) variance of $Q(\omega)$

4) $Q(\omega_d)$ distribution function for frequency ω_d.

5) mean LPC tracks versus time

6) location of $Q(\omega)$ peaks

These results, in all but the last case, are stored on file for later comparison with predicted results (see TESP).

The above list will now be explained in more detail, bearing in mind the operation of program FSIM. Assume that the algorithm used in FSIM was used n_s times between each set of coefficient and Q estimates, and that this process was repeated n_q times, thereby using data samples $y_1, \ldots y_{n_s}, \ldots y_{n_s n_q}$. The MSE results from FSIM are time averages (e.g. $\frac{1}{n_s}\sum_{t=1}^{n_s}(e_{p,t})^2$) for the final stage error from the first n_s iterations. ESP averages these results over a number of FSIM runs using different data realisations. This gives an ensemble and time average, as used in Figure 3.3. If n_s is one, the MSE is simply an ensemble average (e.g. Figure 3.5). The advantage of having both forms of averaging is that the number of ensemble averages needed to achieve acceptable stability is reduced; the disadvantage is loss of resolution in time.

The all-pole spectral estimates are calculated at 128 frequencies between 0 and π radians in FSIM. The mean and variance at each frequency is found by ESP, for each of the spectral estimates (i.e. those at $t = n_s, \ldots n_q n_s$). At the same time the Q distribution function is determined by comparing the value of each spectral estimate at the frequency of interest (ω_d) with the specified set q_i, $i = 1, \ldots k$ (i.e. is $Q(\omega_d) < q_i$, for $i = 1, \ldots k$?). This is repeated for estimates taken at different times, as above. The mean LPC estimates or reflection coefficients, whichever was selected for storage in FSIM, are also determined for each of these values of time.

All of these results are stored on file as follows:

file	results
espmop	MSE, mean $Q(\omega)$, mean a_i or K_i, $i = 1, \ldots p$
espvop	MSE, variance $Q(\omega)$
fqsim	$Q(\omega_d)$ distribution function

The first two of these are in "fsimop" format (see FSIM) and have filter types of S and V respectively. The last is formatted and is used by FQCON (see its description).

ESP assumes the FSIM result files from a batch job will be called "fsimop.1", "fsimop.2", etc. It reads as many of these data files as possible (up to "fsimop.99") storing the information partly on disk and partly in memory. A restriction of the latter

storage method is that the product $pn_q n_r$, where the last term is the number of ensemble averages (FSIM result files) and p is the model order as usual, must be less than 8000.

It is possible for ESP to skip over a complete group of n_q sets of results on "fsimop" files. This was used in Figure 3.3 for example, where initially n_s was 48 and n_q was 1, then n_s was made 1 and n_q twenty. Only the second group of results was analysed to obtain the figure.

FSIM

This is the main (and largest) program and implements all the adaptive filter algorithms considered. It includes whitening filters, all-pole spectral estimation, error averaging (in time), summarised result storage and graphics facilities. The algorithms available are:

code	
T	transversal filter using LMS algorithm, (full precision or 8 bit simulation of ALPS hardware)
G	gradient lattices.. GRAD-HM, GRAD-MIN, GRAD-F+B
U	unnormalised ELS (prewindowed).. ELS-UP
N	normalised ELS (prewindowed).. ELS-NP
C	covariance ELS (normalised) .. ELS-NC

The chosen algorithm uses data stored on an unformatted data file. After a desired period of operation the filter coefficients and time-averaged errors are displayed. The all-pole spectral estimate ($Q(\omega)$) is also calculated and displayed graphically or summarised by a printout giving peak locations and amplitudes. This whole process may be repeated an arbitrary number of times with "snapshots" of filter status being displayed. Final stage errors may be stored at each sampling interval if desired. The filter coefficients, mean square errors and spectral estimates are stored on file for latter use. This file, "fsimop.dat", is unformatted and also contains information such as the time and date, where the data came from etc. Other programs (RLS, BLS, PER and

SIGS) write results in the same format for subsequent processing by ESP and/or TEK10. An identifier is used to indicate the "filter type" in these files as follows:

B block (covariance method) least squares (from BLS)

C covariance sliding window ELS lattice (normalised)

G gradient lattices

M model all-pole spectrum (from SIGS) or predicted var(Q) (from TESP)

N normalised (prewindowed) ELS lattice

P periodogram spectral estimates (from PER)

U unnormalised (prewindowed) ELS lattice

R recursive least square solution (from RLS)

S statistics of batch test; mean results (from ESP)

T transversal (LMS) filters; full or 8 bit precision

V variance batch results (from ESP)

Two subroutines, "group1" and "group2", are used to read these files. The first retrieves a set of data common to all files, including the filter type. The second routine then knows how to read the next group of results, whose format depends on the source.

The algorithms in FSIM are defined by the following equations:

LMS	(2.7) and (2.8)
GRAD-HM	(3.6), (3.7) and (3.10)
GRAD-MIN	(3.6), (3.7), (3.20), (3.21) and (3.22)
GRAD-F+B	(3.6), (3.7), (3.20) and (3.21), see Figure 3.2
ELS-UP	(4.9a)-(4.9h), see Figure 4.1
ELS-NP	(C.1) and (C.2), see Figure C.1
ELS-NC	(C.3) and (C.4), see Figure C.3

and the "whitening filters" (last six algorithms only) for determining the LPCs are defined in section 4.3 or Appendix C.

Initialisation:

In general all variables (backward errors, error covariances etc.) are set to zero

before filtering commences. For the first few samples the ELS lattices are "turned on" one stage at a time, as specified in the relevant references (e.g. [12]). Thus the specification "for $i = 1, \ldots p$" preceding (4.9a) actually applies at or after $t = p$; before this time the algorithm uses "for $i = 1, \ldots t$" where y_1 is the first sample. The gradient lattices were made to use this startup method as well, and their performance was improved. Initialisation required at each new sample is given with the appropriate algorithm (e.g. (4.8)).

Testing:

The program has been tested, as far as possible, to ensure that the algorithms are coded correctly. This is quite important with the more complex lattice structures and whitening filters. Some of the testing methods were: comparison of algorithms against each other (e.g. covariance and prewindowed solutions are equal when a data file with leading zeros is used), comparison with other programs (e.g. BLS), and testing with specific time series, such as impulse responses (see also the example in section B.3).

Noise compensation:

For GRAD-HM only, a facility exists to compensate for the presence of white noise in the input signal. This has not been described in earlier chapters, and a compensation factor of zero would normally be used, thereby disabling the option. For completeness a brief description of this method is now given.

Noise effectively adds zeros to an all-pole signal and this may cause a poor fit to the all-pole model. When this occurs the performance of many of the adaptive filter applications discussed so far is impaired, for example spectral estimates not only lose dynamic range, as would be expected, but also lose resolution and become biased. Two possible solutions are to increase the model order, or to use a pole-zero model. Another approach used in block Linear Prediction methods amounts to subtracting an estimate of the noise power from the leading diagonal of the signal autocorrelation matrix. FSIM implements a method related to this, as suggested by Kay [58], except that the latter is for a block technique and FSIM performs continuous adaption. Each

reflection coefficient update is modified to account for the estimated noise contributions in the numerator and denominator of (3.10c). As the technique decreases stability, the full estimate of the noise power should not be used; the noise compensation factor sets the maximum fraction of the noise power estimate which is employed.

notes:

(1) FSIM must be linked with ARLIB and PLOT10 libraries (see file "fsim.bld").

(2) It is usually necessary to wait for p or sometimes $p+1$ samples to be used following startup before an all-pole spectral estimate is determined.

FORMAT

Converts "fsimop.dat" files from unformatted to formatted versions. The latter may be read by TEK10 using the GROUF routines (see ARLIB). This facility allows (formatted) file transfer between computers and was used to obtain hardcopy plots of high quality.

notes: Only works for S, V and M filter types.

FQCON

As explained in the description of ESP, the measured all-pole spectral estimate distribution function is stored on file "fqsim.dat", for the frequency requested. This contains a separate distribution for each of the n_q sets of spectral estimates calculated. FQCON simply averages these distributions, over the sets of spectral estimates requested, and stores the resulting mean distribution on file "fqcon.dat". (This was used, for example, for Figure 5.4.)

LINE

Adds sinusoidal component(s) to nominated data file containing time series data. The sinusoid amplitude and frequency may be specified plus the range over which it should be added to the existing data. Estimated signal to noise ratios are given.

PER

Periodogram method of spectral estimation using WOSA technique (weighted

overlap segment averaging, see references in [58]). The number of averages per estimate and the percentage overlap may be specified. The data is zero padded with an equal number of zeros to ensure maximum resolution. A Hanning window is used in each FFT.

notes: Link with ARLIB. For compatibility with TEK10 use up to 128 data points per FFT.

PLOT

The program provides a general purpose plotting facility using "PLOT10" graphics software and Tektronics 4010 or similar terminals. Up to three connected-line graphs may be plotted at once and the abscissa values must be equi-spaced. Data files may be formatted or unformatted, one real value per record. The ordinate axis may be automatically or manually scaled, and labelled as desired.

notes: Must be linked to the "PLOT10" library.

REVLEV

A reverse Levinson-Durbin iteration (5.14) is used to determine the LPCs for predictors of all orders, starting from the model LPCs (i.e. all-pole synthesis filter coefficients), which are stored in the specification file used by SIGS (.spe). The forward prediction error powers are also calculated during this process (5.18). These results are stored on file "dbmats.dat" for use by program ATESP.

The normalised autocorrelation function is then determined using (5.15), from which (2.5b) allows the unnormalised autocorrelations to be found. These are stored on file "rxx.dat". The required model order may be higher than that of the all-pole signal; in this case the higher order autocorrelations may be found from the $p-$th order LPCs and autocorrelations as follows. Consider the all-pole synthesis filter of Figure 2.1 and its mathematical description (2.1). If this equation is multiplied by y_{t-k} (y_{t-k} is uncorrelated with e_t if $k > 0$) and expected values are taken, we have

$$r_k = -\sum_{i=1}^{p} a_i^p y_{k-i}$$

from which the values of r_k may be found for $k > p$.

The autocorrelation matrix inverse is calculated with a simple inversion approach using elementary row operations. Part of the arithmetic here was made double precision to avoid numerical accuracy problems. The inverted matrix is stored on file "rinv.dat" for use by program TESP. This allows calculation (in TESP) of the predictor transfer function variance (var$A(\omega) = r(\omega)$) and other variables by using (5.6b), (5.2b) etc. ATESP, the alternative version of TESP, uses the autocorrelation matrix inverse factorisation (5.17) in terms of LPCS of lower orders to achieve the same end (with (5.19) and (5.20)). The matrix inversion routine is checked by multiplying the R matrix by its inverse and displaying the result.

notes: Link with ARLIB.

RLS

This program implements the conventional recursive least square solution for the optimum all-pole model. This uses a sequence of matrix multiplications, additions and scalar divisions to update the matrix inverse, as mentioned in Chapter 4. The resulting equations are very similar to those of the Kalman filter. The algorithm is given in section 7.2 of [40]. The program generates all-pole spectral estimates which may be plotted if a graphics terminal is used. Results are stored in the usual format.

notes: Must be linked with ARLIB and PLOT10 libraries (see file "rls.bld").

RXXSIN

The autocorrelation function of a sinusoid in white noise is calculated by this short program and stored on file "rxx.dat". This is done using (3.32). The results are used by DURBIN or TRACKS.

SETDAT

Creates unformatted real-valued data files, one record per line, from keyboard input.

SIGS

This program generates a normally distributed white sequence which is then filtered using specified pole/zero coefficients. The normally distributed samples are produced by summing 15 uniformly distributed numbers produced by a random number generator and then demeaning the result. The noise variance is $\frac{1}{12}$. The following types of signals can be generated:

(1) stationary pole/zero signals,

(2) "generally stationary" pole/zero signals in which constant filter coefficients are used for nominated periods, and then suddenly changed to produce the "parameter jump" signals,

(3) nonstationary pole/zero signals in which the pole coefficients (and gain) are adjusted each sampling interval, this was used, for example, to make the "random chirps" used in Chapter 3,

(4) impulse responses for pole/zero filters.

The pole and zero coefficients may be entered manually but are usually stored on a filter specification file whose suffix ends in "spe", e.g. ub4p.spe. This is a formatted file with the following structure:

(q is the number of zeros, p is the number of poles, g is an additional gain or scaling factor for the synthesis filter, and $b_1, \ldots b_q$ are the coefficients of the zero's transfer function, while $a_1, \ldots a_p$ are the coefficients of the pole's transfer function.)

(a) for all but the third signal types listed above: $(g > 0)$

$$\begin{array}{ll} & q, p, g \\ (\text{if } z > 0) & b_1, \ldots b_q \\ (\text{if } p > 0) & a_1, \ldots a_p \end{array}$$

This group of up to three lines may be repeated as required. The difference equation is

$$y_t = g(e_t + \sum_{i=1}^{q} b_i e_{t-i}) - \sum_{i=1}^{p} a_i y_{t-i}$$

This is implemented in cannonic form in SIGS. The output power spectrum is calculated from the pole/zero transfer function and stored on file "cpsd.dat", which is in "fsimop.dat" form.

(b) for the continuously nonstationary signals: $(g < 0)$

$$\begin{array}{ll} & q, p, g \\ (\text{if } z > 0) & b_1, \ldots b_q \\ (\text{if } p > 0) & a_1, \ldots a_p \quad \text{(initial)} \\ (\text{if } p > 0) & a_1, \ldots a_p, g \text{ (final)} \end{array}$$

Again this group of lines may be repeated as required. The magnitude of g in the first line is the initial value of gain; the final value of gain comes from the 4th line. The same difference equation applies but now the a_i are time varying according to (for n samples)

$$a_{i,t} = (a_i)_{\text{initial}} + ((a_i)_{\text{final}} - (a_i)_{\text{initial}}) \left(\frac{t}{n-1} \right), \quad t = 0, \ldots n-1, \quad i = 1, \ldots p$$

and similarly for the gain g.

For all except the impulse responses, the synthesis filter is run for 2000 samples before writing results to file. This avoids any startup transients.

notes: (1) Link with library ARLIB. (2) Manual input of filter coefficients can not be used with the third signal type; a filter specification file (.spe) must be employed.

TEK10

As mentioned previously this plotting program shows a sequence of all-pole spectral estimates in frequency and time. It offsets successive estimates and uses a masking method to give a three dimensional effect and not show hidden lines. The program can also plot MSE curves, and it stores this data on file "err.dat" so that it may be reused more flexibly by PLOT. It reads files in the "fsimop" format (see FSIM).

notes: This program also uses "PLOT10" and so must be linked to the appropriate library, as well as ARLIB (see file "tek10.bld").

TESP

The name of this program stands for "Theoretical version of ESP". It calculates the predicted all-pole spectral estimate variance and distribution function, which may then be compared with the batch simulation results determined by ESP.

The variance is determined with the R matrix inverse (stored on file "rinv.dat", see REVLEV) and the model LPCs from the filter specification file (.spe). The name of the latter file is stored in file "rinv.dat". (5.2b), (5.6b), (5.7b), (5.9) and (5.16) are used in these calculations.

The distribution function is calculated using (5.38) by numerical integration for the frequency of interest. The numerical integration is done using Simpson's rule. The values of the real and imaginary transfer function variance, σ_{re} and σ_{im}, plus the correlation coefficient ρ are also shown (via (5.30)), which allows some assumptions from section 5.2 to be checked.

TRACKS

This program incorporates the convergence model presented in Chapter 3. It will calculate the predicted LPC, reflection coefficient and MSE trajectories versus time for the two cases considered. In the first case, that of step change in input statistics, the two autocorrelation functions must be entered. This may be done manually but is more easily achieved by having them on file in the standard format ("rxx.dat"). These files are produced by programs RXXSIN and REVLEV. Recall that the model assumes the signal is uncorrelated across the step change in second order statistics; this will only be partially true for the "parameter jump" tests such as shown in Figure 4.4. Nevertheless the overlap contribution is often small and good agreement with simulations has been achieved. In the second case the smoothly changing autocorrelation function is assumed, in this program, to be modelled by a quadratic for each lag, i.e. $m = 2$ and the general recursive result (3.36) is not included. Only first order modelling was needed for the "linear chirp" example shown in Figure 3.7. Three coefficients, $c_{0,i}$, $c_{1,i}$ and $c_{2,i}$ need to be entered for each lag, $i = 0, \ldots p$.

Once the above data has been entered, the composite autocorrelation function is determined using (3.31) for the first case, or (3.34) and expressions for $S_{0,t}$ to $S_{2,t}$ in the second case. The Levinson-Durbin algorithm is then employed to find the coefficient values at the current value of t. The MSE is also determined from (3.43). These calculations may be repeated at different values of time and the results stored on file.

notes: The model does not predict lattice behaviour during the overlap period for case 1 situations. This generally causes little inconvenience since this only lasts for p samples. A warning message is given by TRACKS if this situation occurs.

XTRACT

TEK10 is the plotting program for "fsimop" files but it can not display all the information contained in these files. Most programs store either LPC or reflection coefficient estimates as well in their "fsimop"-type result files. These may be extracted (one at a time) by XTRACT to generate a sequence of (say) the K_3 estimates at regularly spaced intervals. This selected data may then be plotted using PLOT (e.g. Figure 3.6a).

B.3 Examples

Some typical examples of interactive program use are given on the following pages. The programs are largely self-explanatory so far as user input is required. User responses have been underlined and some additional comments included on the right hand side of the page. A example of a command file used to generate data for ensemble averaging is also given.

```
$ type p2.spe                                          sample filter
0,2,1.                                                 specification file
-1.2,.92
$
$ run sigs                                             generate 2 pole signal
  enter the random number seed
  (use -ve number to get impulse response)
5678
 enter the output file name
p2.dat
 choose coeffs from existing file? [Y,N]
y
 enter the spec file name
p2.spe
 how many samples are required?
1000
coeffs of trans funct denominator are ..
 -1.20000   0.92000
      1000 signal samples calculated
calculated (output) signal variance =      0.8909
maximum estimate is     13.5 dB above mean
(spectral) dynamic range is     33.7 dB
FORTRAN STOP
$
$ run fsim
 enter the filter type (T,G,N,U,C) and order, eg G5    ELS-UP simulation using
u4                                                     same data
 enter the run description
example for appendix b
 enter name of input data file
p2.dat
 store forward error output/gamma term on file? [Y,N]
n
 store ai or ki on fsimop file? [A,K]
a
 enter the form of output required ..
     -1    no output
      0    graphics display of q estimate
      1    filter coeffs and errors
      2    as for 1 plus q spectral summary
      3    as for 2 plus all q estimates
1
filter type is U
 enter the exponential weight factor
.995
 use "frozen"[F] or "exact"[E or A] whitening filter
a
 enter number of iterations per Q, and number of Qs
200,2
itrn    1 gamma c terms:
 0.000
itrn    2 gamma c terms:
 0.000  0.000
```

```
itrn    3 gamma c terms:
 0.480  0.480  0.000
itrn    4 gamma c terms:
 0.661  0.480  0.480  0.000
itrn    5 gamma c terms:
 0.516  0.510  0.389  0.389
itrn    6 gamma c terms:
 0.995  0.206  0.204  0.129
itrn    7 gamma c terms:
 0.574  0.389  0.127  0.127
itrn    8 gamma c terms:
 0.956  0.559  0.277  0.101
itrn    9 gamma c terms:
 0.961  0.857  0.559  0.257
itrn   10 gamma c terms:
 0.823  0.823  0.805  0.495
itrn   18 gamma c terms:
 0.995  0.731  0.501  0.496
itrn   19 gamma c terms:
 0.916  0.841  0.726  0.496
```

FSIM prints the gamma c gains if the highest order term is < 0.5, this occurs at startup or if the input signal statistics change

```
sample number =   200
refl"n coeffs (or partial correl"n)
  -43.2407   36.8170   -1.1163   -0.0666
average squared (forw.) errs for each stage:
  0.6422E+00  0.3783E+00  0.6343E-01  0.6123E-01  0.5974E-01
lpc"s are
  -1.29511   1.02756  -0.11579  -0.00767
final stage forw. err. power =       -1.445 dB
             applies when data is from "SIGS" program
```

partial correlations

$\Delta_{1,200}$ .. $\Delta_{4,200}$

$a_{1,200}$ .. $a_{4,200}$

```
sample number =   400
refl"n coeffs (or partial correl"n)
 -109.5804  102.8640   -0.2364    0.4419
average squared (forw.) errs for each stage:
  0.1301E+01  0.7947E+00  0.8527E-01  0.8517E-01  0.8421E-01
lpc"s are
  -1.20816   0.97975  -0.05406   0.03091
final stage forw. err. power =        0.045 dB
             applies when data is from "SIGS" program
 continue  or restart  (C,R)?
r
cpu time (secs) per iteration =  0.00209
 enter the filter type (T,G,N,U,C) and order, eg G5
g4
 enter the run description
GRAD-HM test
 enter name of input data file
p2.dat
```

restart simulation using gradient lattice

```
 store forward error output/gamma term on file? [Y,N]
y
 enter name of error/gamma  file
p2.err
 store ai or ki on fsimop file? [A,K]                    store final stage error
a                                                        on file "p2.err"
 enter the form of output required ..
     -1    no output
      0    graphics display of q estimate
      1    filter coeffs and errors
      2    as for 1 plus q spectral summary
      3    as for 2 plus all q estimates
2
filter type is G
 enter gradient weighting[<1] and noise comp fact[0-1]
   and alg. type (1=GRAD-HM, 2=GRAD-MIN, 3=GRAD-F+B)
.995,0,1
 use "frozen"[F] or "exact"[E or A] whitening filter
a
 enter number of iterations per Q, and number of Qs
200,2


sample number =   200
refl"n coeffs (or partial correl"n)                      K1,200  ..  Kq,200
   -0.6288    0.8864   -0.1243   -0.0224
average squared (forw.) errs for each stage:
  0.6422E+00  0.3868E+00  0.6921E-01  0.6699E-01  0.6561E-01
lpc"s are
  -1.29351   1.01065  -0.09522  -0.02241
final stage forw. err. power =       -1.038 dB
             applies when data is from "SIGS" program
 peak     20.5dB     freq=  0.145
 min     -10.6dB      freq=  0.496                       Q peak summary


sample number =   400
refl"n coeffs (or partial correl"n)
   -0.6171    0.9337   -0.0152    0.0262
average squared (forw.) errs for each stage:
  0.1301E+01  0.8005E+00  0.8798E-01  0.8784E-01  0.8693E-01
lpc"s are
  -1.20787   0.97681  -0.04691   0.02624
final stage forw. err. power =       -0.184 dB
             applies when data is from "SIGS" program
 peak     25.5dB     freq=  0.145
 min     -10.3dB      freq=  0.496
 continue  or restart  (C,R)?
n
cpu time (secs) per iteration =  0.00365
FORTRAN STOP
$
```

```
$ run revlev
 enter the spec file name
p2.spe                                          LPCs for all orders
 LPCs are ..                                    (1st and 2nd in this case)
  -0.625
  -1.200    0.920
 forward prediction err powers: 0 -          2
  0.890   0.543   0.083
 normalised autocorrelations are (from o-th lag)..
 1.000000  0.625000 -0.170000
 R(0)=   0.8903136
 what model order is required?
4
  R matrix is ..
     0.890    0.556   -0.151   -0.694
     0.556    0.890    0.556   -0.151              autocorrelation matrix
    -0.151    0.556    0.890    0.556
    -0.694   -0.151    0.556    0.890
 rows swapped ..          2          3
 inverse R is ..
    12.000  -14.400   11.040    0.000
   -14.400   29.280  -27.648   11.040
    11.040  -27.648   29.280  -14.400
     0.000   11.040  -14.400   12.000
 rank of Rinv =           4
 inverse test check..
     1.000    0.000    0.000    0.000
     0.000    1.000    0.000    0.000
     0.000    0.000    1.000    0.000
     0.000    0.000    0.000    1.000
FORTRAN STOP
$
$ run tesp
 enter N, the number of samples (or the effective
          number) used by filter to produce Qs      corresponds to lambda=.995
400
 p2.spe     used to generate Rinv
  with order =           4                         r(0)  .. r(127.pi/128)
 r(as a function of freq.) is :
   0.003  0.003  0.003  0.003  0.003  0.003  0.003  0.003  0.003  0.003
   0.002  0.002  0.002  0.002  0.002  0.002  0.002  0.002  0.002  0.002
   0.002  0.001  0.001  0.001  0.001  0.001  0.001  0.001  0.001  0.001
   0.001  0.001  0.001  0.001  0.001  0.000  0.000  0.000  0.001  0.001
   0.001  0.001  0.001  0.001  0.001  0.001  0.001  0.001  0.002  0.002
   0.002  0.002  0.003  0.003  0.003  0.004  0.004  0.005  0.005  0.005
   0.006  0.006  0.007  0.007  0.008  0.009  0.009  0.010  0.010  0.011
   0.012  0.013  0.013  0.014  0.015  0.016  0.016  0.017  0.018  0.019
   0.020  0.021  0.021  0.022  0.023  0.024  0.025  0.026  0.027  0.028
   0.029  0.029  0.030  0.031  0.032  0.033  0.034  0.035  0.036  0.036
   0.037  0.038  0.039  0.040  0.040  0.041  0.042  0.042  0.043  0.044
   0.044  0.045  0.045  0.046  0.046  0.047  0.047  0.048  0.048  0.048
   0.049  0.049  0.049  0.049  0.050  0.050  0.050  0.050
```

```
 max Q    240.9899     at freq index of          37
 largest delta =   0.1166968     at freq index of          37
 largest |s|/r   =    1.000000     at freq index of           1
 |s|/r at largest delta =  2.5833165E-02
 assume alpha=1 for var(Q) calc? [Y,N]
y
 largest normalised var=   0.2333936     at freq index of          37
 enter freq. index for numerical int (-ve to stop)
37
 parameters of the Ar,Ai pdf ...
 sigma.re, sigma.im, rho =  1.5618406E-02  1.5501738E-02  2.4714591E-02
 assuming the following ..
 sigma, eta, 1/eta**2 =   1.5560197E-02  6.4417012E-02   240.9899
 enter the number of integration steps (even)
200
 enter first q, step size and number of steps
100,50,6
 q=    100.0000     F(q)=   1.4259636E-02
 q=    150.0000     F(q)=   0.1587420
 q=    200.0000     F(q)=   0.3865971
 q=    250.0000     F(q)=   0.5786434
 q=    300.0000     F(q)=   0.7117121
 q=    350.0000     F(q)=   0.7991847
 q=    400.0000     F(q)=   0.8565834
 enter freq. index for numerical int (-ve to stop)
-1
FORTRAN STOP
$
$ run setdat
 enter the file name
fibon.dat
 enter the sample values, 1 per line (-99 to stop)
1
1
2
3
5.00001
8
13
-99
 how many zeros would you like on the end ?
0
FORTRAN STOP
$
$ run fsim
 enter the filter type (T,G,N,U,C) and order, eg G5
c2
 enter the run description
fibonacci test
 enter name of input data file
fibon
 store forward error output/gamma term on file? [Y,N]
n
```

predicted distribution function (by numerical integration) at peak frequency

what are the prediction coefficients for the Fibonacci sequence ?

use 5.00001 to ensure a small prediction error (otherwise whitening filter will crash !)

use covariance estimator so that no windowing assumptions are made

```
 store ai or ki on fsimop file? [A,K]
a
 enter the form of output required ..
     -1    no output
      0    graphics display of q estimate
      1    filter coeffs and errors
      2    as for 1 plus q spectral summary
      3    as for 2 plus all q estimates
2
filter type is C
 enter sliding window length (<500)
6
 use "frozen"[F] or "exact"[E or A] whitening filter
a
 enter number of iterations per Q, and number of Qs
6,1


sample number =     6
 exact a(i)    1.000000      -0.9999974       -1.000003
 exact b(i)  -0.9999952       0.9999915        1.000000
refl"n coeffs (or partial correl"n)
    0.9923    1.0000
average squared (forw.) errs for each stage:
  0.6688E+00  0.2326E+00  0.2762E-04
lpc"s are
  -1.00000  -1.00000
final stage forw. err. power =      -34.797 dB
             applies when data is from "SIGS" program
 min      -7.0dB      freq=  0.250
 continue  or restart  (C,R)?
n
cpu time (secs) per iteration =  0.00781
FORTRAN STOP
$
```

almost right!
forward predictors
say (see (3.3))
"current element
minus previous two
elements is approx.
zero "

```
  type stats.44
$! stats.44
$!  for fig 4.4 K1 convergence tests
$set noverify
$!  use p1ns1 or p1ns2
$datf="p1nS1"
$filt="G2"
$wht="A"
$EXP=".98,0.,3"
$!   initialise ..
$seed=77
$datfs=datf+".spe"
$datfa=datf+"*.dat;*"
$count=0
$del fsimop.*;*
$!
$start:
$seed=seed+113
$open/write sig sig.com
$!
$write sig "$run sigs"
$write sig seed
$write sig datf
$write sig "Y"
$write sig datfs
$WRITE SIG "150"
$write sig "160"
$!
$write sig "$run fsim"
$write sig filt
$write sig "run for fig 3.4  "
$write sig datf
$write sig "N"
$write sig "K"
$write sig "-1"
$write sig EXP
$write sig wht
$write sig "10,30"
$write sig "N"
$close sig
$!
$@sig
$delete 'datfa'
$purge sig.com
$count=count+1
$rename fsimop.dat fsimop.'count'
$if count.lt.20 then goto start
$dir/size fsimop.*;*
$exit
$
```

sample command file used to generate ensemble averages for Figure 3.4

after this has run as a batch job, the files "fsimop.1", .. "fsimop.20" can be processed by ESP

Appendix C Normalised Exact Least Square Algorithms

Chapter 4 discusses the Exact Least Square unnormalised prewindowed lattice algorithm (ELS-UP) and points out some of the similarities to certain gradient algorithms examined in Chapter 3. A normalised form of this algorithm exists (ELS-NP) and is given in this appendix. Covariance forms of the ELS lattices are available, and the normalised sliding-window form (ELS-NC) is also given here. Both of these algorithms are implemented in program FSIM. Derivations for these adaptive lattices are not trivial and may be found in [6] and [10] for the prewindowed and covariance forms respectively. In section C.3 some aspects of their behaviour are discussed and some performance examples presented. Again only the scalar case is considered.

C.1 ELS-NP

As stated in section 4.2, the standard prewindowed algorithm, (4.8) and (4.9), may be normalised in a way that involves more than just magnitude normalisation. The result is surprisingly simple and elegant, with the number of update equations reduced by a factor of two, to only three. The forward and backward error covariances and the gamma terms disappear during the combined magnitude and "angle" normalisation. A disadvantage is that additional types of arithmetic operations are required; in particular the following function is very common and is called the complementary form of the variable, denoted

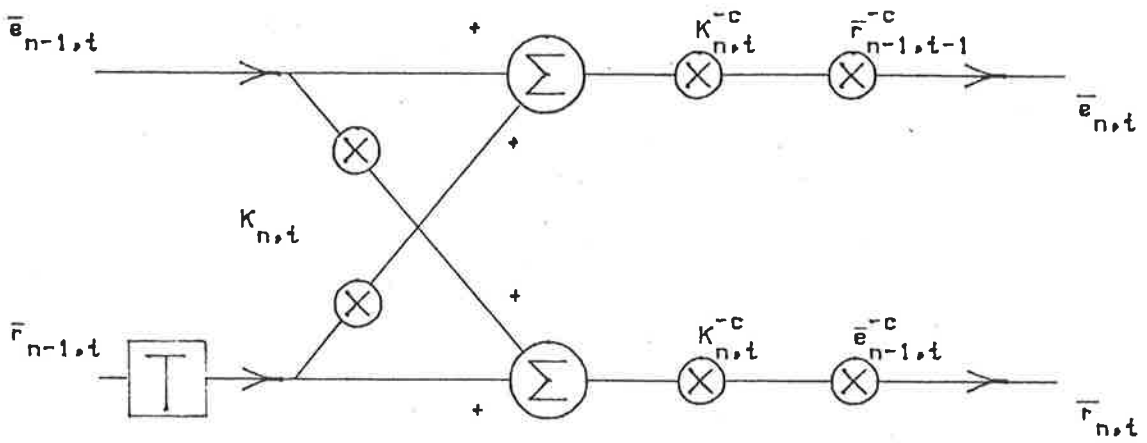
$$x^c = (1 - x^2)^{0.5}$$

with x^{-c} meaning the reciprocal of x^c.

The ELS-NP algorithm uses normalised backward and forward errors whose magnitudes do not exceed unity. These will be denoted by $\bar{r}$ and $\bar{e}$. The initialisation for each new time sample is

$$R_t = \lambda R_{t-1} + y_t^2 \qquad (C.1a)$$

Figure C.1 ELS normalised prewindowed lattice (ELS-NP) n-th stage



$$\bar{e}_{0,t} = \bar{r}_{0,t} = \frac{y_t}{R_t^{0.5}} \qquad (C.1b)$$

λ ,as usual, is the exponential weighting factor, ≤ 1, and p is the model order. Then, for $n = 1, \ldots \min\{p, t\}$

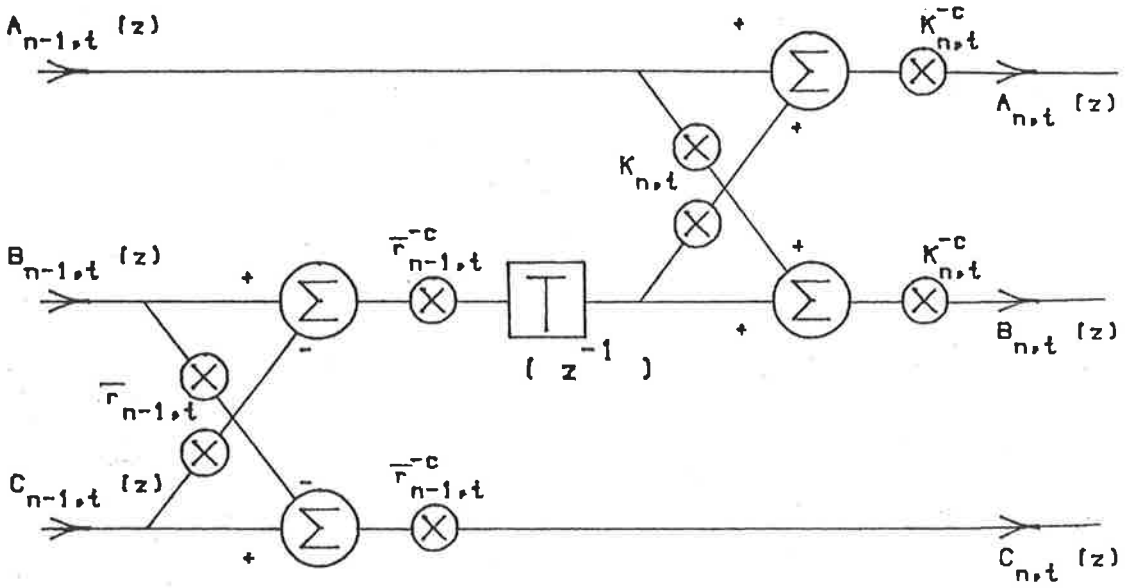
$$K_{n,t} = K_{n,t-1}\bar{e}^{c}_{n-1,t}\bar{r}^{c}_{n-1,t-1} - \bar{e}_{n-1,t}\bar{r}_{n-1,t-1} \qquad (C.2a)$$

$$\bar{e}_{n,t} = (\bar{e}_{n-1,t} + K_{n,t}\bar{r}_{n-1,t-1})K^{-c}_{n,t}\bar{r}^{-c}_{n-1,t-1} \qquad (C.2b)$$

$$\bar{r}_{n,t} = (\bar{r}_{n-1,t-1} + K_{n,t}\bar{e}_{n-1,t})K^{-c}_{n,t}\bar{e}^{-c}_{n-1,t} \qquad (C.2c)$$

Some signs have been changed in these equations, from [6], to make the lattice structure consistent with previous prewindowed cases; the effect is to change the signs of the reflection coefficients. R_t and the backward errors are all initially zero, before startup. Figure C.1 shows one stage of this lattice. The exact whitening filter is shown in Figure C.2, which indicates the relation between the predictor transfer functions $A(z)$ and $B(z)$ for adjacent orders. This follows the practice used in [10], where the vectors of LPCs are converted to polynomials in z^{-1}, and it is shown how the projection approach can also produce update equations for the whitening filter transfer functions. The predictor values may be deduced in a manner similar to that described in section

Figure C.2 ELS-NP whitening filter (n-th stage)



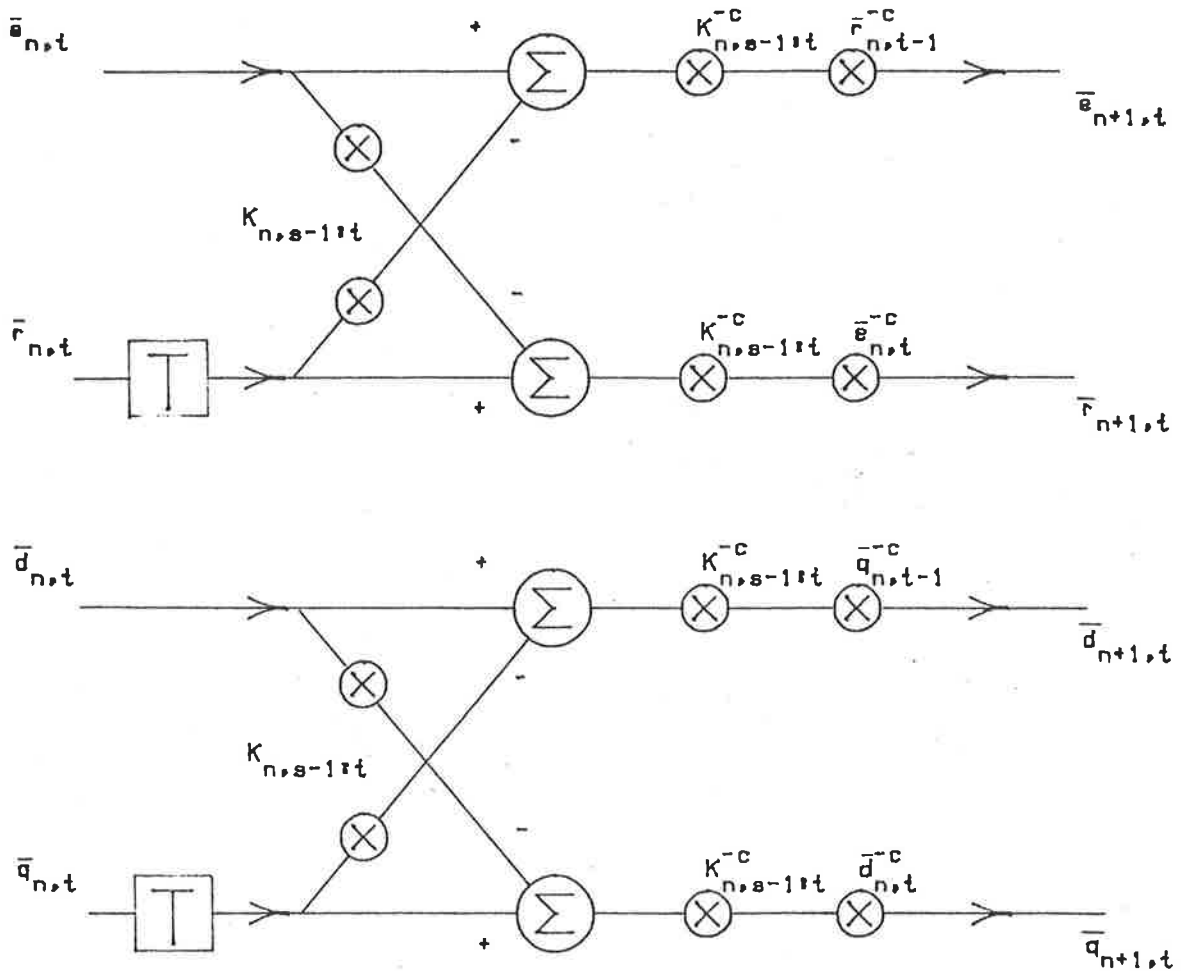
4.3, i.e. to take the impulse response of the exact whitening filter (only one stage of which is shown in Figure C.2). The impulse is applied at the A_0 and B_0 inputs.

C.2 ELS-NC

This algorithm uses the "covariance" assumption regarding predictor estimates, which, as explained in section 4.1, makes no assumptions about the data samples outside the specified window. Both "growing memory" and "sliding window" algorithms exist for the covariance case. The second is more appropriate for adaptive estimators and is given here. The optimum forward and backward predictors are determined with equal weight given to all prediction errors in the window. As this rectangular window slides forward with each new sampling interval, the oldest data sample in the window drops out. Initially the data in the window is assumed to be zero, and while the window is "filling up" the covariance lattice is equivalent to a prewindowed lattice with $\lambda = 1$. Note that to minimise the squared errors $\bar{e}_{t-w}, \ldots \bar{e}_t$ the data samples used are $y_{t-w-p}, \ldots y_t$ for the $p-$th order forward predictor. Thus the value of s, the time index of the first

Figure C.3 ELS normalised covariance lattice

(n+1)th stage



sample used (see (4.1a)), is $t-w-p$. Interestingly, the lattice structure consists of two almost independent prewindowed (normalised) lattices, one fed with the current data sample and the other with y_t delayed w times. This is shown in Figure C.3 and the following equations.

For each new sample:

$$\bar{e}_{0,t} = \bar{r}_{0,t} = \left(\sum_{i=t-w}^{t} y_i^2 \right)^{-0.5} y_t \qquad (C.3a)$$

$$\overline{d}_{0,t} = \overline{q}_{0,t} = \left(\sum_{i=t-w}^{t} y_i^2 \right)^{-0.5} y_{t-w} \qquad (C.3b)$$

Then for $n = 0, \ldots p-1$

$$K_{n,s-1:t} = \overline{e}^c_{n,t} K_{n,t-1} \overline{r}^c_{n,t-1} + \overline{e}_{n,t} \overline{r}_{n,t-1} \qquad (C.4a)$$

$$K_{n,t} = (K_{n,s-1:t} - \overline{d}_{n,t} \overline{q}_{n,t-1}) \overline{d}^{-c}_{n,t} \overline{q}^{-c}_{n,t-1} \qquad (C.4b)$$

$$\overline{e}_{n+1,t} = (\overline{e}_{n,t} - K_{n,s-1:t} \overline{r}_{n,t-1}) K^{-c}_{n,s-1:t} \overline{r}^{-c}_{n,t-1} \qquad (C.4c)$$

$$\overline{d}_{n+1,t} = (\overline{d}_{n,t} - K_{n,s-1:t} \overline{q}_{n,t-1}) K^{-c}_{n,s-1:t} \overline{q}^{-c}_{n,t-1} \qquad (C.4d)$$

$$\overline{r}_{n+1,t} = (\overline{r}_{n,t-1} - K_{n,s-1:t} \overline{e}_{n,t}) K^{-c}_{n,s-1:t} \overline{e}^{-c}_{n,t} \qquad (C.4e)$$

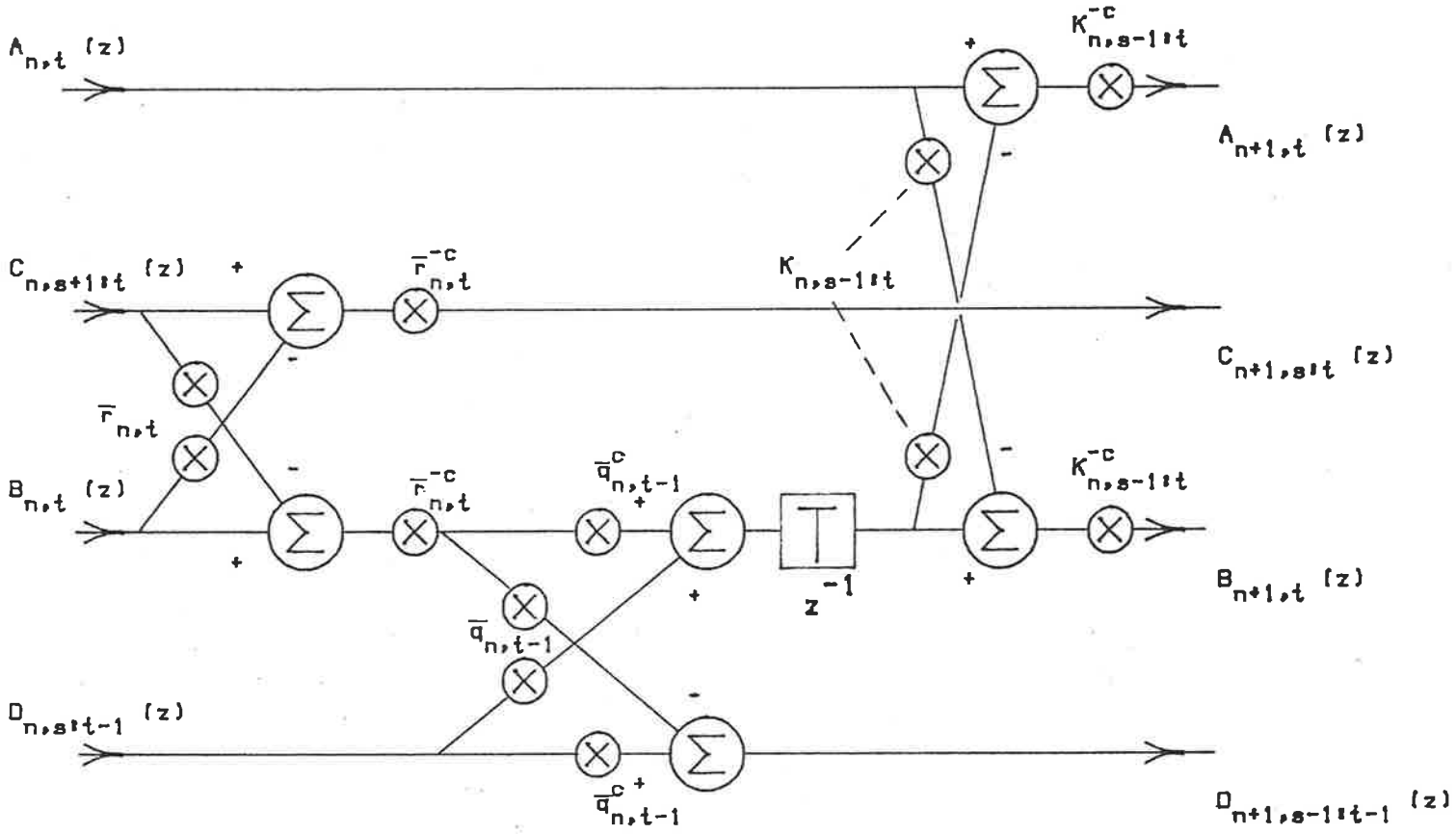
$$\overline{q}_{n+1,t} = (\overline{q}_{n,t-1} - K_{n,s-1:t} \overline{d}_{n,t}) K^{-c}_{n,s-1:t} \overline{d}^{-c}_{n,t} \qquad (C.4f)$$

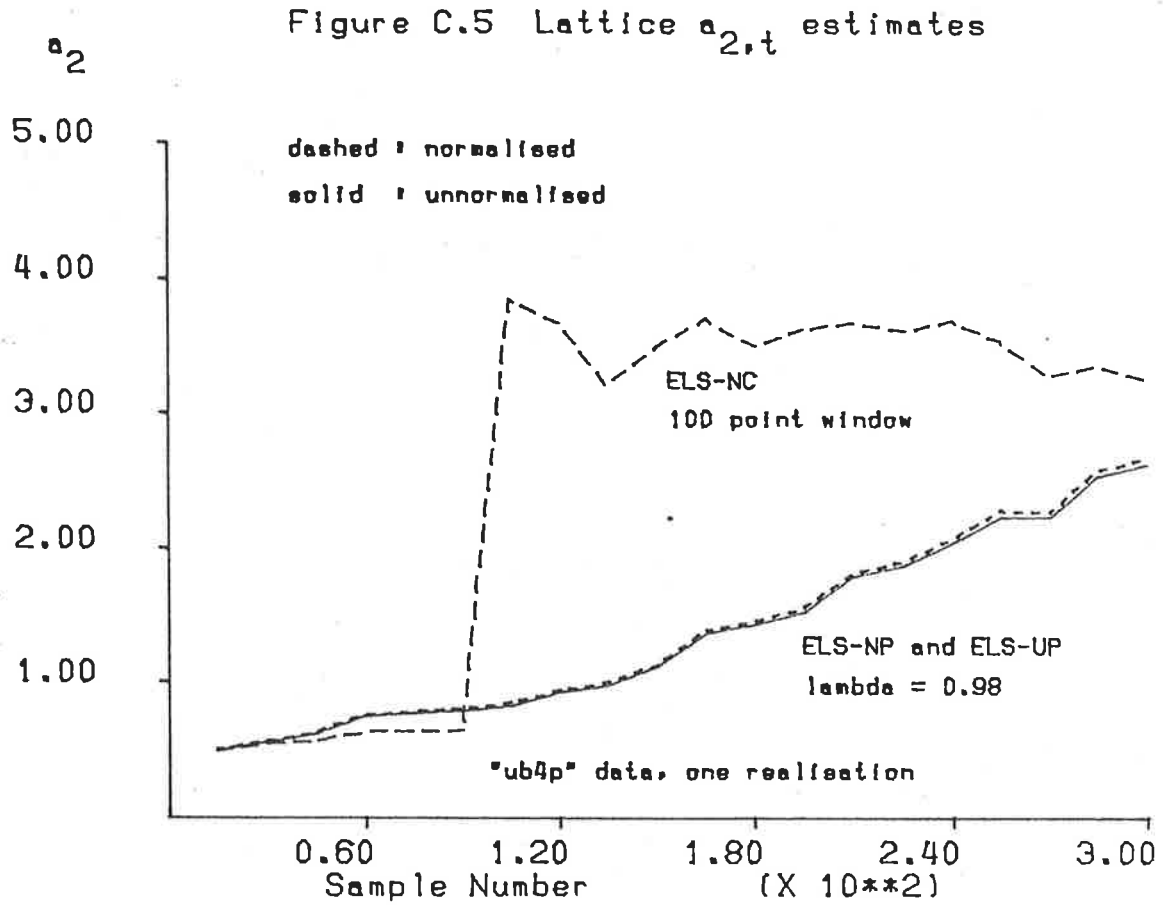
The three subscript notation again indicates the order and then the range of data samples over which the (say) reflection coefficient depends. The second subscript is omitted if equal to s $(= t - w - n)$. The exact whitening filter for ELS-NC is shown in Figure C.4. Again the actual LPCs may be determined by considering the response when an impulse is applied to A_0 and B_0. Intuitive understanding of the operation of these normalised whitening filters is difficult compared to their unnormalised counterparts. Although their derivation is complex, as Porat says, it may be obtained by an "apparently mechanical procedure", once the projection operator update formulas are available.

C.3 Further Notes on the Normalised ELS Forms

The normalised and unnormalised ELS algorithms of course give the same (optimum) predictor estimates when used over the same data. Because of the high numerical accuracy of the simulation program FSIM, little difference in LPC estimates could be observed between ELS-UP and ELS-NP. In practical filtering implementations the normalised versions might perform much better that their unnormalised counterparts, although they require a specialised architecture, as mentioned in Chapters 4 and 8. It is possible to unnormalise the normalised variables (see, for example, [12] appendix 3).

Figure C.4 ELS-NC whitening filter, (n+1)th stage





The main differences between the exponentially weighted prewindowed lattice and the sliding window covariance lattice are firstly in the additional assumptions made by the prewindowing method, and secondly, the difference in extent of the data samples used. Assuming λ and N, the number of samples in the sliding window, are related according to (5.23), then the LPC estimates will have the same statistics provided N and t are large enough and the time series is stationary. In this case the prewindowing assumptions will have been "forgotten" due to the exponential weighting. However, if the signal is only locally stationary, and perhaps exhibits "jumps" in its nonstationarity, then the covariance lattice is likely to perform better. Also if t is not large enough the startup effects may be significant. This is particularly true with more coherent signals. Figure C.5 shows an example of $a_{2,t}$ estimates for $t = 15, 30, \ldots 300$ for signal "ub4p" (see section 4.4). The lattices were arranged to have the same size effective data window

according to (5.23). The covariance lattice shows a sudden change in estimates when the sliding window becomes full. (The "growing memory" covariance algorithm would have been a better choice during the first 100 samples.) The prewindowed lattice shows no such dramatic change and obviously takes a longer time to approach the ideal value of 3.8106. ELS-UP and ELS-NP estimates are virtually identical, as expected.

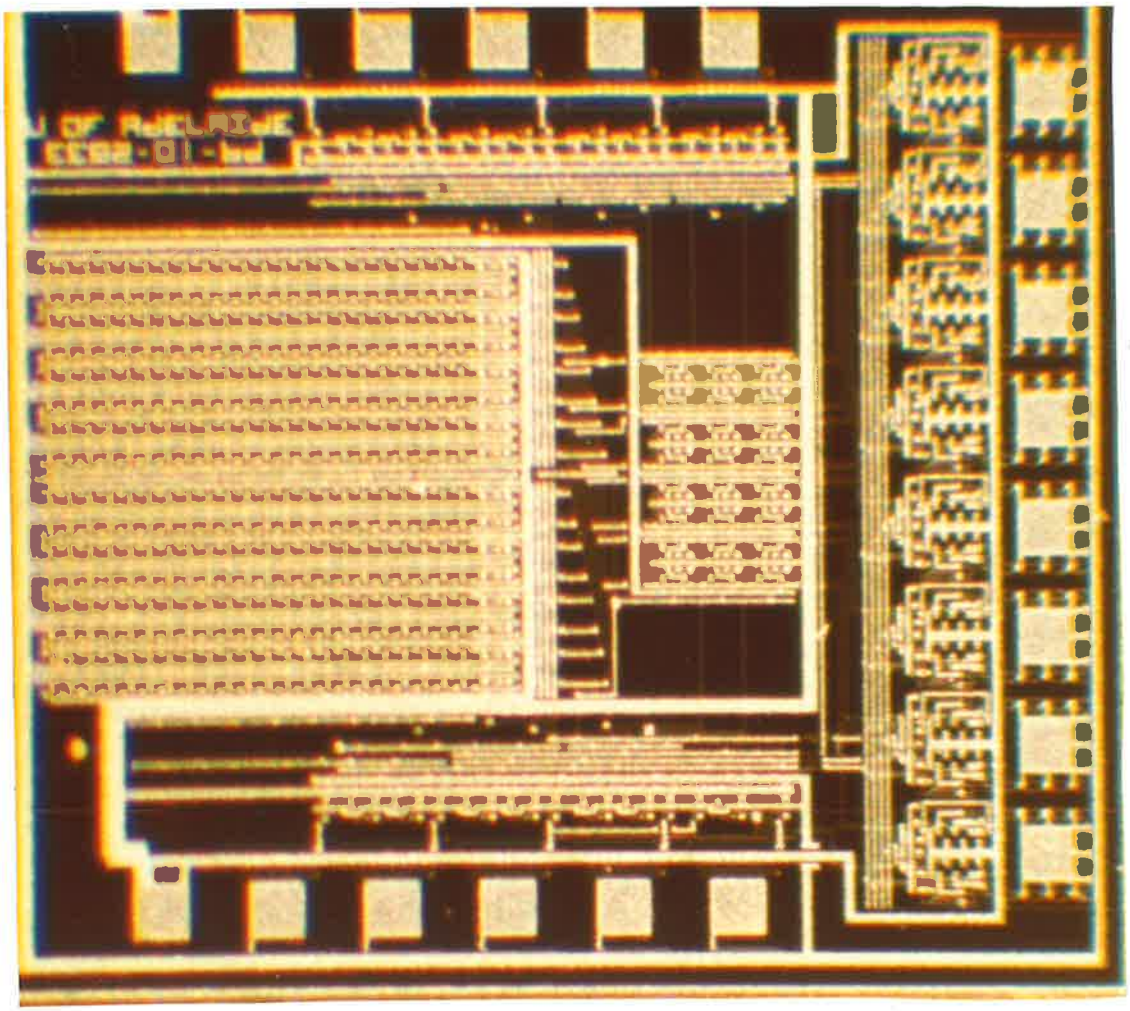
Appendix D Supplementary Information concerning ALPS

This appendix is divided into three sections, each of which gives additional information concerning the Adaptive Linear Predictor System (ALPS) described in Chapter 6. The first section gives some details of how the the ALPS data store was fabricated in integrated circuit form. Section D.2 provides an example of part of a state sequence diagram for the LMS algorithm, and the last section demonstrates how an adaptive lattice algorithm could be implemented using the recirculating memory technique. A number of final year students assisted the author with the ALPS project and their assistance is gratefully acknowledged. Chris Carroll and Tony Chapman designed many components of the integrated circuit data store, and John Stobie and George Bergholtz assisted with hardware construction and host microprocessor software.

D.1 ALPS Data Store

As previously indicated this portion of the ALPS hardware was implemented as part of a Multi-Project Chip (MPC) exercise. Figure D.1 contains a photograph of the completed device and identifies the main components. An nMOS process was used with 5 micron minimum line widths. The device contains approximately 1800 transistors and is 2.41 mm by 2.65 mm.

The VLSI design was performed using simple design rules [71] and a low-level embedded layout language called "Belle". Standard input pads and augmented library cell tristate pads were employed. Software for design rule checking, circuit extraction and switch level simulation was used to test the design. The output from the design process was a file in "Caltech Intermediate Format" (CIF) describing the mask geometry for each layer of the integrated circuit. After fabrication the project chips were tested in the ALPS hardware using special finite state sequences to exercise each part of the project. No design errors were found. The maximum reliable clock frequency was 6.5 Mhz and the power dissipation was 165 mW.



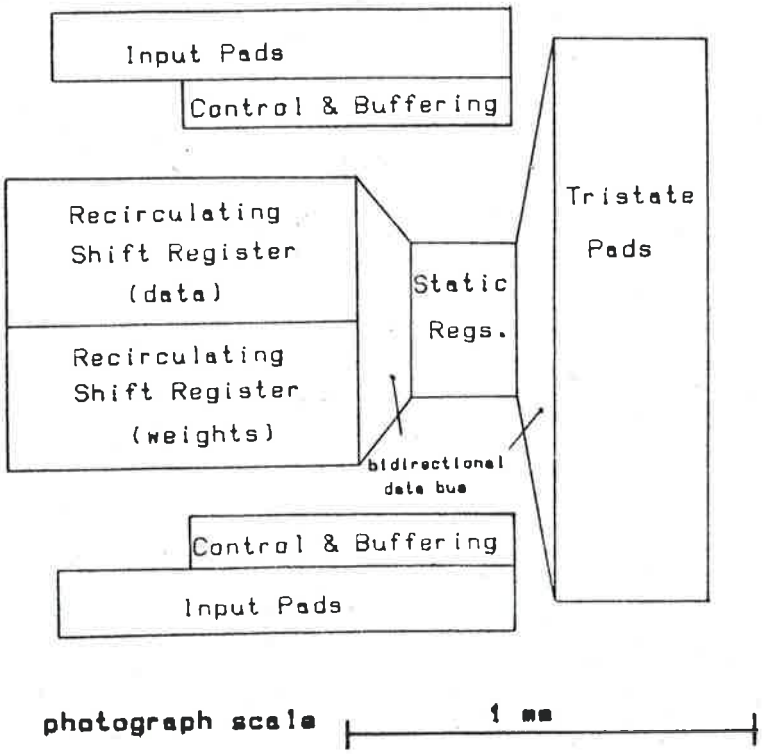


Figure D.1 Photograph and layout of ALPS data store

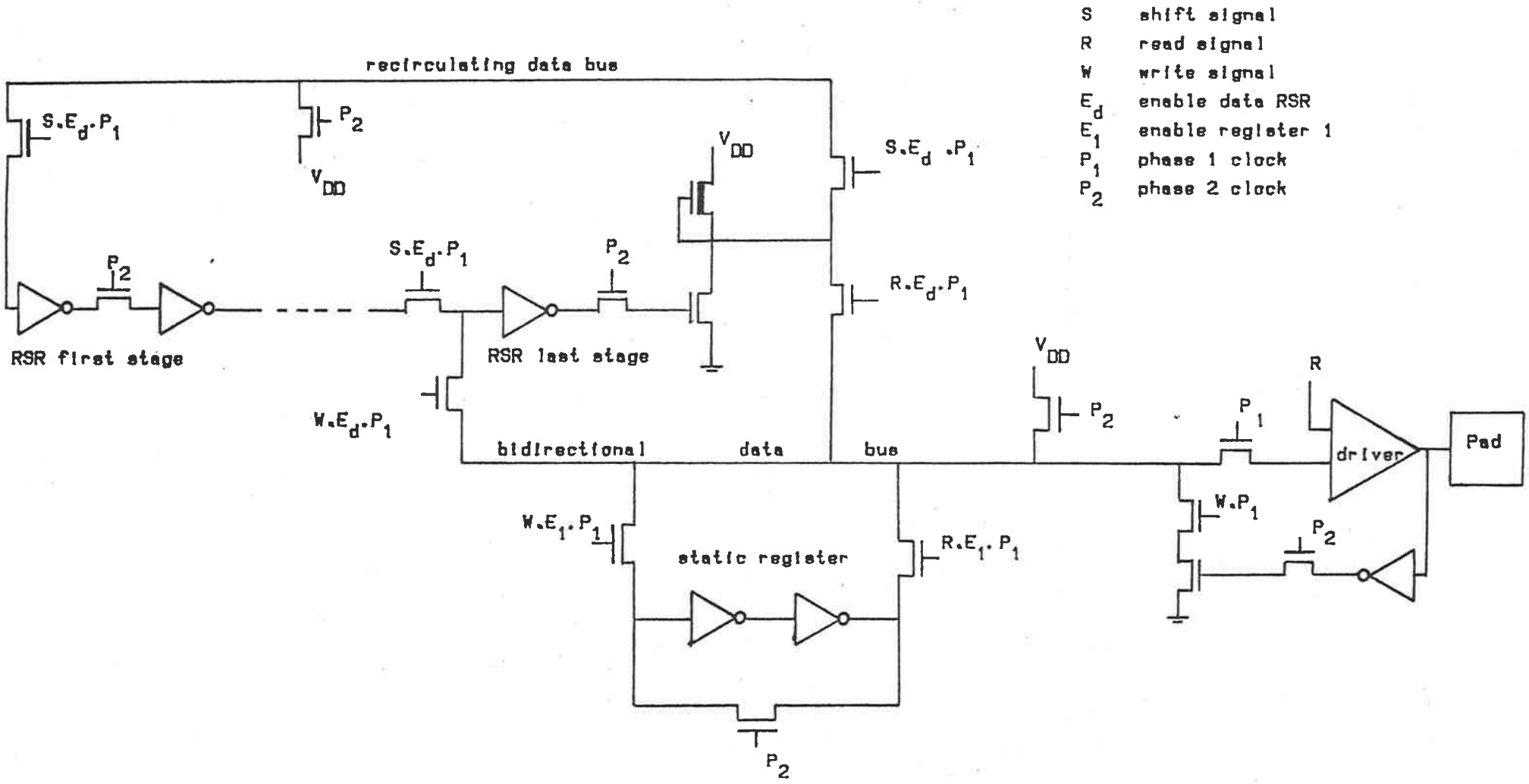


Figure D.2 Simplified circuit diagram for ALPS data store

A simplified circuit diagram of the integrated circuit is shown in Figure D.2. The timing scheme uses two-phase non-overlapping clocks, with all data transfers during phase 1, and precharging and refreshing on phase 2. Data is internally transferred between components on a bidirectional pre-charged data bus. For example if the static register is writing to the internal bus, then during phase 1 if the read (R) and enable (E_1) signals are high, when the register output is low the bus will be discharged via the pass transistor and the final inverter pull down transistor. The signals R, W and S shown in the figure are mutually exclusive.

D.2 State Sequence Example

Figure D.3 shows approximately a quarter of the finite state sequence generally used in ALPS. This sample is now described to give an indication of how mathematical algorithms are translated into control sequences in ALPS. The portion shown evaluates (2.7); other state sequences initialise the data store contents with the nominated data values, dump the adaptive weight values to the host processor, and perform the LMS weight update, (2.8). Section 6.2 describes the data store contents before and during the LMS algorithm computations.

Evaluating (2.7) uses states numbered from 25 to 41 (out of the total of 64 states). Output signals from the control memory are shown next to those states for which they are active. Some of these signals are shown in Figure D.2. Those used here are

signal	meaning
S	shift (right)
R	read (from data store)
W	write (to data store)
Ed	enable data RSR
Ew	enable weight RSR
In	arithmetic processor instruction, n = 0 to 7

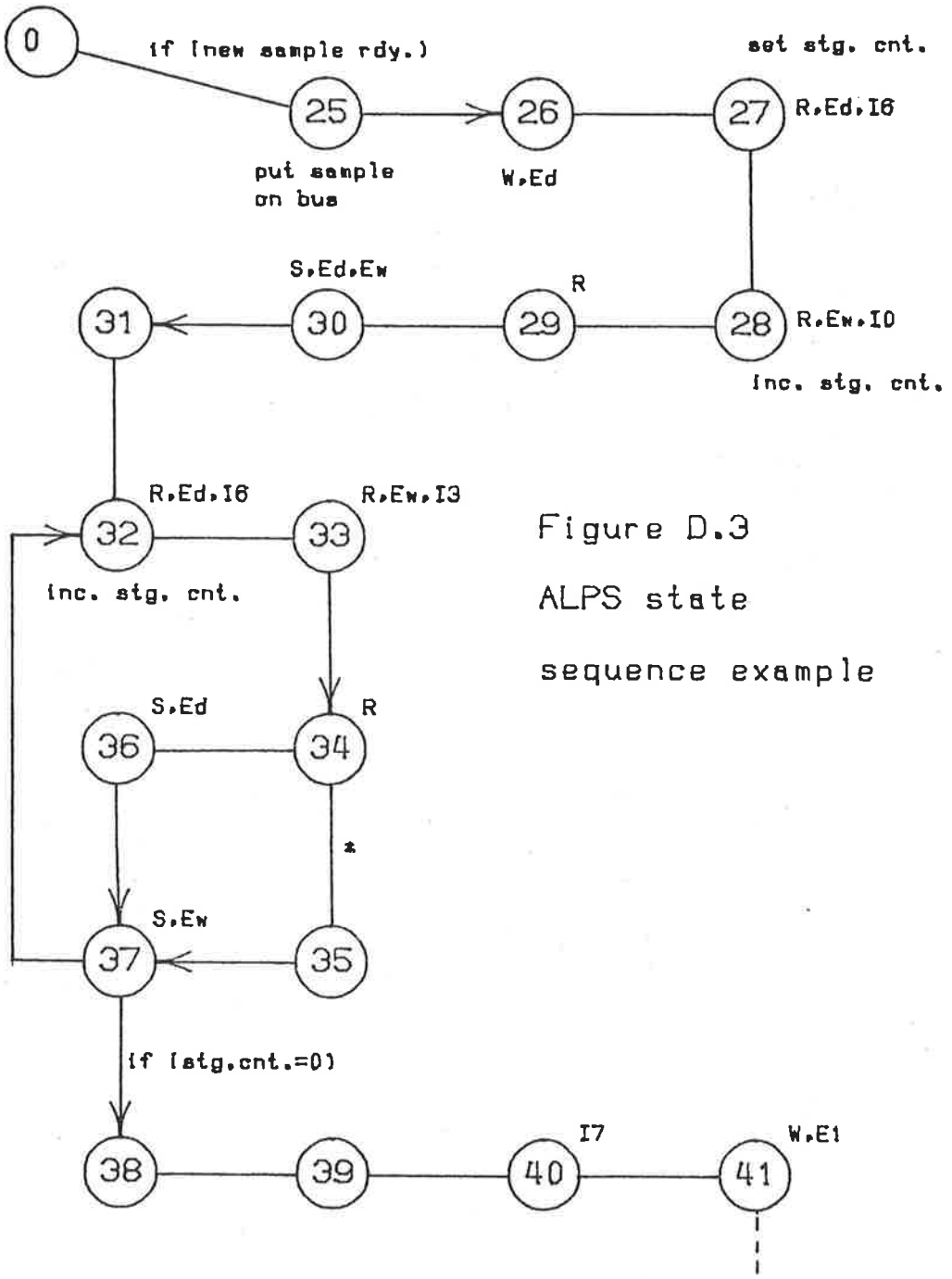


Figure D.3
ALPS state
sequence example

When the new data sample y_t is available at the host-filter interface (see Figure 6.1) the condition for transition to state 25 becomes true, and if the controller is currently in state 0 this changes to 25 on the next clock cycle. The data sample is read from the interface onto the filter data bus and written to the data RSR (state 26). The same data is then read from the data store and an instruction (I6) is sent to the arithmetic processor to accept the first of two operands for a multiplication operation.

The second operand, the zero-th order weight (=1), is sent to the arithmetic processor at state 28 with an instruction to start the multiply operation and transfer the result to the accumulator when finished. During the multiply time both RSRs are right shifted. The loop from state 32 to 37 repeats these operations, except that the instructions I6 and I3 cause each multiplication result to be added to the accumulator to form the sum of products. The stage counter is incremented once each time around the loop so that the appropriate number of loops can be completed. The path through state 35 is only taken on the last pass through the loop if non-adaptive filtering has been requested. This leaves the data RSR ready to accept the next data sample in its right-most element. When the result (e_t) is ready it is read from the arithmetic processor (I7) and written to the first static register.

D.3 Adaptive Lattice Implementation on ALPS

Gradient lattice algorithms could be readily implemented using the recirculating memory approach discussed in Chapter 6. To achieve this on ALPS two data store chips would be used, thereby giving circulating memory for four data vectors. The following notes indicate how the GRAD-F+B algorithm could be coded in such a scheme.

Assume that the backward errors, forward and backward error covariances and error cross-correlations are stored in the four memories as follows. (This shows the data immediately after the arrival of the new data sample y_t).

$$
\begin{array}{lll}
r_{p-1,t-1} & \cdots & r_{1,t-1}\ \ r_{0,t-1}\ (A) \\
w^e_{p,t-1} & \cdots & w^e_{2,t-1}\ \ w^e_{1,t-1}\ (B) \\
w^r_{p,t-1} & \cdots & w^r_{2,t-1}\ \ w^r_{1,t-1}\ (C) \\
v'_{p,t-1} & \cdots & v'_{2,t-1}\ \ v'_{1,t-1}\ (D)
\end{array}
$$

Denote the righthandside elements, which are the only ones accessible, as A, B, C and D. Five of the static registers will be used for temporary storage of results as follows:

$M1$ current forward error $e_{n-1,t}$

$M2$ delayed backward error $r_{n-1,t-1}$

$M3$ new backward error $r_{n,t}$

$M4$ current backward reflection coefficient $K^r_{n,t}$

$M5$ current forward reflection coefficient $K^e_{n,t}$

The initialisation for each new data sample is

$$y_t \rightarrow M1, M3$$

then

$$\begin{aligned}
A &\rightarrow M2 && (**)\\
M3 &\rightarrow A &&\\
\lambda B + M1 * M1 &\rightarrow B && \text{for (3.20}a\text{)}\\
\lambda C + M2 * M2 &\rightarrow C && \text{for (3.20}b\text{)}\\
\lambda D - M1 * M2 &\rightarrow D && \text{for (3.20}c\text{)}\\
D \div C &\rightarrow M4 && \text{for (3.21}a\text{)}\\
D \div B &\rightarrow M5 && \text{for (3.21}b\text{)}\\
M2 + M5 * M1 &\rightarrow M3 && \text{for (3.7)}\\
M1 + M4 * M2 &\rightarrow M1 && \text{for (3.6)}
\end{aligned}$$

Following this group of operations, all memories would be shifted to the right once and the whole sequence repeated (starting from $(**)$) p times. The coding of this algorithm is surprising simple as the three time update equations (3.20) are well matched to the data allocation scheme and memory access capabilities. The only slight complication occurs with backward error order updates (3.7), as these errors must be

stored for use in the next sampling interval. This is done by transferring the old backward error to M2 at the start of each loop, and using M3 as a temporary storage location for the new (next stage) backward error during the shift.

Multiplication by the exponential weight λ could be achieved by a shift and subtraction:

$$\lambda x = x - 2^{-m}x$$

This would slightly restrict the permissible values of λ and require hardware for right shifting by m bits, but save considerable time in multiplication operations. GRAD-HM could be coded in a similar manner and only three memories would be needed. Although the former algorithm generally performs better, in fixed point implementations the possibility of reflection coefficients exceeding one in magnitude should be provided for. This could be done with some guard bits and extra shift operations quite readily.

Appendix E VLSI Signal Processor

A signal processor is currently being designed at the Adelaide University Department of Electrical Engineering for fabrication on one VLSI circuit [86]. Some of the main features of its design are: a novel architecture featuring a high degree of parallelism and suited to complex-valued operations; programmable operation allowing reasonably complicated algorithms to be used (e.g. adaptive lattice filters); 16 bit arithmetic operations, including division and 32 bit sum of products accumulation; suitability for stand-alone or array applications; and finally, fast operation. It is intended that the design will be fabricated by a two micron CMOS process and is likely to contain more than 10^5 devices. The project is called "TFB", which stands for Transform and Filtering "Brick".

E.1 Evolution of the TFB Architecture

In approximately the middle of 1983 the signal processing and VLSI groups within this department decided to work jointly towards a design specification for a signal processing chip capable of taking full use of advanced VLSI design and fabrication facilities. This was made possible by an agreement with the Microelectronics Centre of North Carolina (MCNC). The signal processing participants, including several staff members, postgraduates and the author, evolved a design suited to three target applications: FFTs for speech processing, convolution for acoustic ranging, and adaptive lattice filtering for radar applications. This architecture featured four separate data memories and associated buses, plus four multipliers and accumulators. The arithmetic elements were positioned between appropriate buses so that complex-valued arithmetic could be handled easily, or so that real operations could proceed in parallel.

After review of this proposal by the VLSI group, some significant changes to the layout and interconnection between circuit elements were made. The most important of these was to connect the four previously independent buses into one circular (ring) bus.

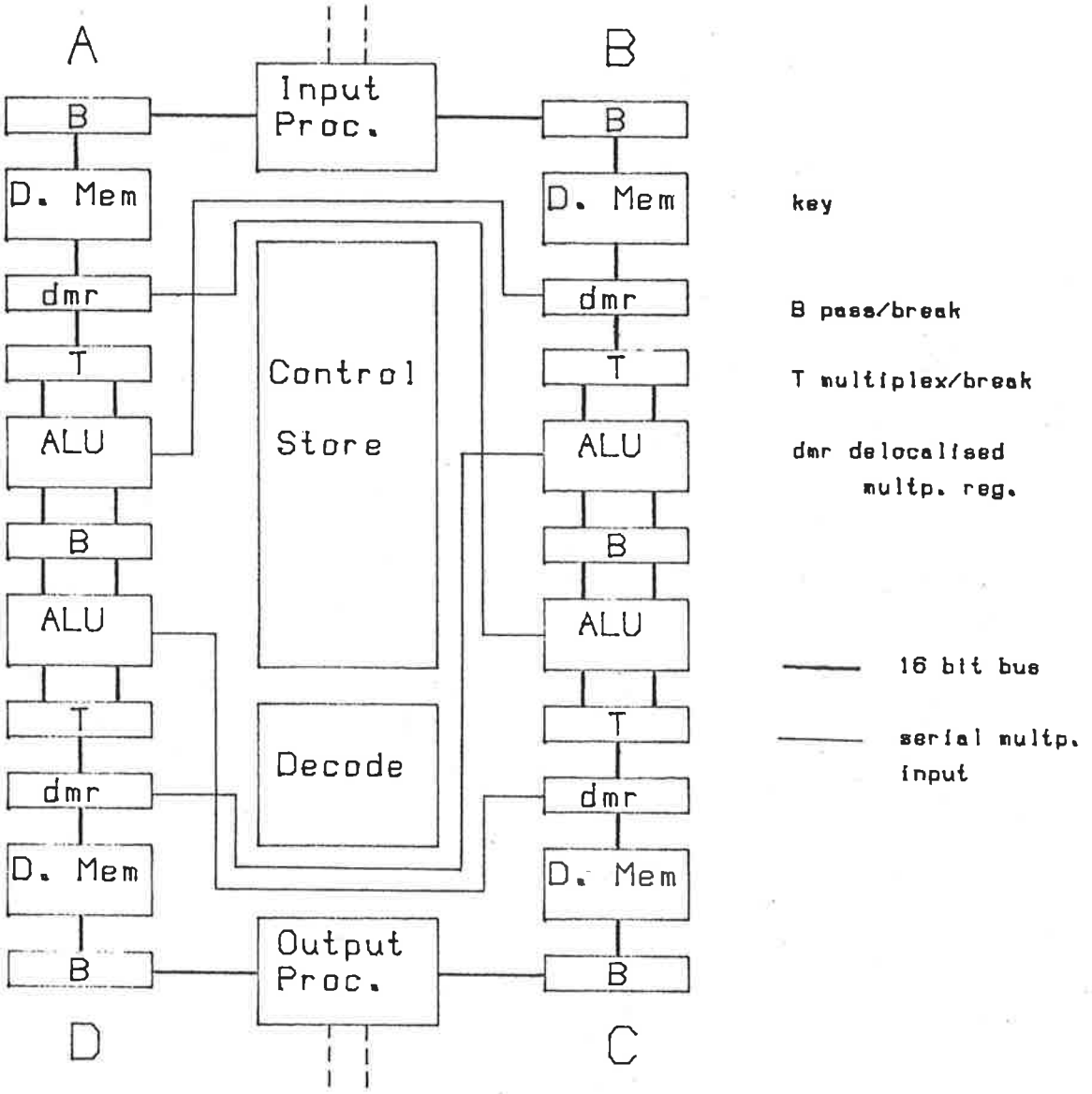


Figure E.1 TFB Architecture

This contains switches so that the bus may be segmented or continuous, which allows great flexibility in sequential or parallel data flow. Figure E.1 shows TFB architecture and layout. The following section contains brief descriptions of the main components in the design.

E.2 TFB Components

The ring bus shown in the figure contains six pass/break switches labelled "B"

and four multiplex/break switches labelled "T". The ALU portion of the ring bus is 32 bits wide and the T switches control which, if either, of the 16 bits are connected to the remaining 16 bit ring bus. By controlling each of the ring bus switches independently very flexible operation may be achieved; for example with all switches open, four items of data may be moved to 8 locations in the one operation and a complex multiply initiated. Alternatively a single item of data may be moved to the opposite side of the device with a choice of two routes. Most of the ring bus is "embedded" in the data memory or arithmetic components, thereby using chip area efficiently.

The four data memories are 16 bits wide and 32 words long. Two memory pointers are associated with each memory for addressing the current word to be read from, or written to. These pointers may be autoincremented or decremented by various step sizes.

The ALUs each contain a multiplier/divider, adder/subtractor and an accumulator. The multipliers are sequential devices so that most of the same circuitry may be reused for the divider. Multiply operations require two operands; one comes from the lower 16 bits of the ALU bus and the other comes serially from the "delocalised multiplier registers" (dmr). This achieves the requirements of the original architecture, but eliminates bus crossovers. To perform a complex multiplication, for example, the operands a and b may be stored in the following data memories:

quadrant A	real (a)
quadrant B	real (b)
quadrant C	imag (b)
quadrant D	imag (a)

To set up the multiplication each of these data words is latched into the multiplier and dmr is its quadrant. The multipliers are started on the next clock cycle, thereby

evaluating the following

quadrant A	real (a) real (b)
quadrant B	real (b) imag (a)
quadrant C	imag (b) real (a)
quadrant D	imag (a) imag (b)

The real part of the product is obtained by subtracting the multiplier D result from that of multiplier A, and similarly for the imaginary component. The proposed multiplier design takes 9 clock cycles and produces a 32 bit result. The divider will use a 32 bit dividend and 16 bit divisor, and will take 17 clock cycles. Other arithmetic operations and data transfers will take only one clock cycle.

On opposite sides of the ring bus are an input and output processor. It is envisaged that these will contain enough local intelligence to operate almost independently from the other components, and allow 8 or 16 bit data transfers, either synchronously or asynchronously. Synchronous operation in a linear or multidimensional array of TFB chips will enable very high data throughput rates.

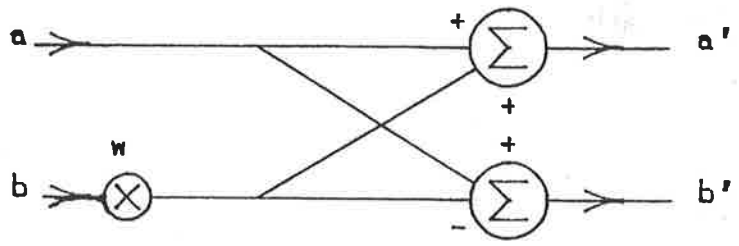
In the centre of the TFB design lies the program memory (control store) and associated decoders, loop counters etc. The control store has 128 words, each 64 bits wide, of random access memory. The width is needed to control the large amount of hardware that may be used in parallel. Words in control store may either represent instructions involving data (conditional or unconditional), or "load immediate" instructions. The latter class allows internal registers such as memory pointers, loop counters, the program counter or control masks to be initialised.

E.3 Examples of TFB applications

(a) FFT Processors

One TFB processor could be used as the arithmetic element in a simple FFT processor. A fast microprocessor controller and memory would also be required. $\frac{N}{2} \log_2 N$

butterflies are required per N point FFT, and each butterfly requires one multiplication

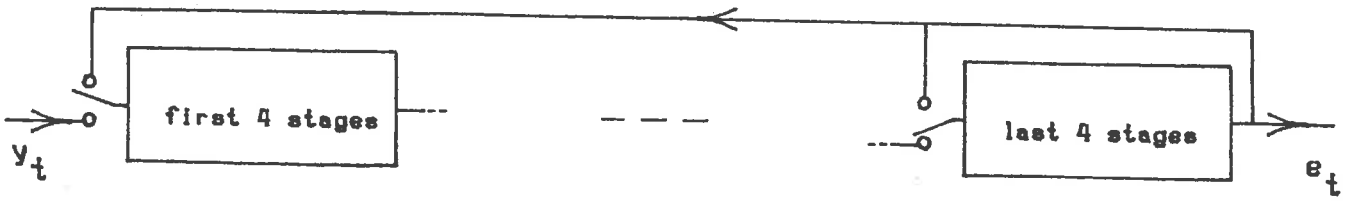


by a complex weight w, followed by addition and subtraction as shown, [85] where a, b are the input data pair and a', b' are their transformed values. For a 1024 point FFT, 8MHz clock rate and 10 clock cycles between multiply starts (with all additions, subtractions and data transfers being done in parallel with the multiplications) the time taken is 6.4 milliseconds. This corresponds to a throughput rate of 160 thousand samples per second.

Alternatively a two dimensional array of TFB chips could process the FFT at very high rates. For the 1024 point case again, 32 chips in each column (32 complex samples per chip) and 10 columns (one per stage of the FFT), then assuming pipelining, the throughput rate approaches 50 megasamples per second.

(b) LMS algorithm

The simplest adaptive algorithm is the transversal filter LMS algorithm given by (2.7) and (2.8). If real-valued data is being used, a minimum of four stages per TFB processor would seem sensible to take advantage of the four internal multipliers. The algorithm could therefore be implemented in a linear array of TFB chips, four stages per chip, to obtain the greatest speed, or a single chip could be used (with $p \leq 63$). With this algorithm the error e_t must be evaluated before the coefficient updates can be started. In the array case the first chip would initially calculate the partial error $a_0 y_t + \cdots + a_3 y_{t-3}$. The final error e_t would be available after the last chip in the chain had completed summing the partial sum of products from preceding chips. Each chip



would then perform weight updates for its four weights. The array configuration can obviously achieve greater overall throughput rate than the single chip implementation, but some time is lost in the former case due to the propagation of the partial sums of products down the processor chain. This loss becomes worse as the chain gets longer. On the current TFB specifications and assuming a 31 stage model, the sampling rate for the LMS algorithm would be approximately 36 kHz for the single chip and 125 kHz for the 8 chip array. To avoid additional multiplexing hardware in the array case, the input and output ports could be configured in the byte transfer mode, thereby effectively giving two input ports. One would be used for data samples and the other for the current error. This would reduce the throughput rate to 100 kHz. The array solution could also implement more than four stages per processing element, and use fewer elements, to obtain intermediate performance. FIR digital filtering could obviously be implemented in very similar ways except without the coefficient updates.

(c) Complex-valued GRAD-HM

In Chapter 7 some applications suitable for the complex-valued adaptive lattice are discussed. This is good example of how TFB can be used to implement a more complicated algorithm. It can be realised, with several stages per TFB, at approximately 15 microseconds per stage per complex data point. The main difficulty in coding the algorithm is that two memory pointers exist for each data memory and sometimes more than this number of variables are required. This can be overcome by incrementing pointers in steps of (say) 8 to access different groups of data when necessary, and by careful use of memory so that only the minimum number of variables has to be stored (e.g. only storing one copy of the backward errors). The division operation is required

(3.10c), and would preclude this algorithm from many alternative signal processing chips. More than half the program memory is required for this algorithm.

Bibliography

1 Makhoul J., "Linear Prediction: A Tutorial Review", Proc. IEEE, Vol. 63, No. 4, pp. 561-579, April 1975

2 Widrow B. et al., "Stationary and Nonstationary Learning Characteristics of Adaptive Filters", Proc. IEEE, Vol. 64, pp. 1151-1161, Aug. 1976

3 Widrow B. et al., "Adaptive Noise Cancelling: Principles and Applications", Proc. IEEE, Vol. 63, pp. 1692-1716, Dec. 1975

4 Griffiths L.J., "Rapid Measurement of Digital Instantaneous Frequency", IEEE Trans. on ASSP, Vol. ASSP-23, pp. 207-222, Apr. 1975

5 Box G.E.P. and G.M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco, 1976

6 Lee D.T.L., M. Morf and B. Friedlander, "Recursive Least Squares Ladder Estimation Algorithms", IEEE Trans. on ASSP, Vol. ASSP-29, pp. 627-641, Jun. 1981

7 Gibson C.J.and S. Haykin, "Learning Characteristics of Adaptive Lattice Filtering Algorithms", IEEE Trans. on ASSP, Vol. ASSP-28, pp. 681-691, Dec. 1980

8 Makhoul J.I. and L.K. Cosell, "Adaptive Lattice Analysis of Speech", IEEE Trans. on ASSP, Vol. ASSP-29, pp. 654-659, Jun., 1981

9 Hodgkiss W.S. (Jr.) and J.A. Presley (Jr.), "Adaptive Tracking of Multiple Sinusoids Whose Power Levels are Widely Separated", IEEE Trans. on ASSP, Vol. ASSP-29, pp. 710-721, Jun. 1981

10 Porat B., B. Friedlander and M. Morf, "Square Root Covariance Ladder Algorithms", IEEE Trans. on Aut. Con., Vol. AC-27, pp. 813 -829, Aug. 1982

11 Youn D.H., N. Ahmed and G.C. Carter, "On Using the LMS Algorithm for Time Delay Estimation", IEEE Trans. on ASSP, Vol. ASSP-30, pp. 798-801, Oct., 1982

12 Friedlander B., "Lattice Filters for Adaptive Processing", Proc. IEEE, Vol. 70, pp. 829-867, Aug. 1982

13 Friedlander B., "Lattice Methods for Spectral Estimation", Proc. IEEE, Vol. 70, pp.990-1017, Sep. 1982

14 Haykin S., B.W. Currie and S.B. Kesler, "Maximum Entropy Spectral Analysis of Radar Clutter", IEEE Proc., Vol. 70, pp. 953-962, Sep., 1982

15 Stearns S.D. and L.J. Vortman, "Seismic Event Detection Using Adaptive Predictors", Proc. IEEE Int. Conf. of ASSP Soc., pp. 1058-1061, 1981

16 Lee D.T.L. and M. Morf, "A Novel Innovations Based Time-Domain Pitch Detector", Proc. IEEE Int. Conf. of ASSP Soc., pp. 40-44, 1980

17 Gibson C. and S. Haykin, "Performance Studies of Adaptive Lattice Prediction-Error Filters for Target Detection in a Radar Environment Using Real Data", Proc. IEEE Int. Conf. of ASSP Soc., pp. 1054-1057, 1981

18 Demytko N. and L.K. Mackechnie, "A High Speed Digital Adaptive Echo Canceller", Aust. Telecom. Res., Vol. 7, No. 1, 1973

19 Davis B.R., "A Review of Echo Cancellers", Bell Labs, Tech. Mem., TM-77-1344-13, June 1977

20 Zeidler J,R. et al., "Adaptive Enhancement of Multiple Sinusoids in Uncorrelated noise", IEEE Trans. on ASSP, Vol. ASSP-26, pp. 240 -253, Jun 1978

21 Ralston A, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1965

22 Haykin S. (Ed.), *Non Linear Methods of Spectral Analysis*, Springer-verlag, "Topics in Applied Physics", Vol. 34, Berlin, 1979

23 Ulrych T.J. and T.N.Bishop, "Maximum Entropy Analysis and Autoregressive Decomposition", Rev. Geophys., Space Phys., Vol. 13, pp. 183-200, Feb 1975

24 Kailath T., "A View of Three Decades of Linear Filtering Theory", IEEE Trans. on Inform. Thy., Vol. IT-20, pp. 146-181, March 1974

25 Van den Bos A., "Alternative Interpretation Of Maximum Entropy Spectral Analysis", IEEE Trans. on Inform. Thy., Vol. IT-17, pp. 493-494, July 1971

26 Berg J.P., "Maximum Entropy Spectral Analysis", 37th Ann. Intern. Meeting of Soc. of Explor. Geophys., Oklahoma, Oct. 1967

27 Ables J.G., "Maximum Entropy Spectral Analysis", Astron. Astrophys. Suppl. Series, Vol. 15, pp. 383-393, 1974, reprinted in *Modern Spectrum Analysis*, Ed. D.G. Childers, IEEE Press, New York, 1978

28 Kay S.M., "Spectrum Analysis - A Modern Perspective", Proc. IEEE, Vol. 69, pp. 1380-1419, Nov. 1981

29 Clarke R.J., "AUSMPC 5/82 Designer Documentation", CSIRO Divn. of Comp. Res., Aug. 1982

30 Kim J.K. and L.D. Davisson, "Adaptive Linear Estimation for Stationary M-dependent processes", IEEE Trans. Inform. Thy., Vol. IT-21, pp. 23-31, Jan. 1975

31 Makhoul J., "A Class of All-Zero Lattice Digital Filters: Properties and Applications", IEEE Trans. on ASSP, Vol. ASSP- 26, pp. 304-314, Aug. 1978

32 Makhoul J., "Stable and Efficient Lattice Methods for Linear Prediction", IEEE Trans. on ASSP, Vol. ASSP-25, pp. 423-428, Oct. 1977

33 Griffiths L.J., "A Continuously-Adaptive Filter Implemented as a Lattice Structure", Proc. IEEE Int. Conf. of ASSP Soc., pp. 683-686, 1977

34 Gibson C. and S. Haykin, "A Comparison of Algorithms for the Calculation of Adaptive Lattice Filters", IEEE Int. Conf. of ASSP Soc., pp. 978-983, 1980

35 Itakura F. and S. Saito, "Digital Filtering Techniques for Speech Analysis and Synthesis", Seventh Int. Conf. on Acoustics, pp. 261-264, Budapest, 1971

36 Honig M.L. and D.G. Messerschmitt, "Convergence Properties of an Adaptive Digital Lattice Filter", IEEE Trans. on ASSP, Vol. ASSP- 29, pp. 642-653, Jun., 1981

37 Honig M.L., "Convergence Models for Lattice Joint Process Estimators and Least Squares Algorithms", IEEE Trans. on ASSP, Vol. ASSP-31, pp. 415-425, Apr., 1983

38 Papoulis A. *Probability, Random Variables and Stochastic Processes*, McGraw-Hill Kogakusha, Tokyo, 1965

39 Larimore M.G. and B.J. Langland,, "Recursive Linear Prediction for Clock Synchronization", Proc. IEEE Int. Conf. of ASSP Soc. pp. 572-575, 1981

40 Goodwin G.C. and R.L. Payne, *Dynamic System Identification; Experimental Design and Data Analysis*, Academic Press, Mathematics and Science in Engineering series, Vol. 136, New York, 1977

41 Falconer D.F. and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization", IEEE Trans. on Commun., Vol. COM-26, pp. 1439-1446, Oct. 1978

42 Halkias C.C., G. Carayannis, J. Dologlou, and D. Emmanoulopoulos, "A New Generalized Recursion for the Fast Computation of the Kalman Gain to Solve the Covariance Equations", Proc. IEEE Int. Conf. of ASSP Soc., pp. 1760-1763, 1982

43 Morf M., D.T. Lee, J.R. Nickolls and A. Vieira, "A Classification of Algorithms for ARMA Models and Ladder Realizations", Proc. IEEE Int. Conf. of ASSP Soc., pp. 13-19, 1977

44 Satorius E.H. and J.D. Pack, "Application of Least Squares Lattice Algorithms to Adaptive Equalization", IEEE Trans. on Commun., Vol. COM-29, pp. 136-142, Feb. 1981

45 Soderstrom T., L. Ljung and I. Gustavsson, "A Theoretical Analysis of Recursive Identification Methods", Automatica Vol. 14, pp. 231-244, 1978

46 Ljung L., "Recursive Identification Techniques", Proc. IEEE Int. Conf. of ASSP Soc., pp. 627-630, 1982

47 Shensa M.J., "Recursive Least Squares Lattice Algorithms – a Geometric Approach", IEEE Trans. on Aut. Con., Vol. AC-26, pp. 695-702, June 1981

48 Hynes R. and R.E. Gardner, "Doppler Spectra and X-band Signals", Suppl. to IEEE Trans. on Aerosp. and Elect. Sys., Vol. AES-3, No. 6, Nov. 1967

49 Medaugh R. and L.J. Griffiths, " A Comparison of two Fast Linear Predictors", Proc. IEEE Int. Conf. of ASSP Soc., pp. 293-296, 1981

50 Medaugh R. and L.J. Griffiths, "Further Results of a Least Squares and Gradient Adaptive Algorithm Comparison", Proc. IEEE Int. Conf. of ASSP Soc., pp. 1412-1415, 1982

51 Morf M. and D.T. Lee, "Recursive Least Squares Ladder Forms for Fast Parameter Tracking", Proc. IEEE Int. Conf. Decis. and Cont., pp. 1362-1367, 1978

52 Akaike H., "Power Spectral Estimation Through AR Model Fitting", Ann. of Instit. for Stat. Maths., Tokyo, pp. 407-419, 1969

53 Mann H.B. and A. Wald, "Statistical Treatment of Linear Stochastic Difference Equations", Econometrica, Vol. 11, pp. 173-220, July 1943

54 Parzen E., "Time Series Analysis", Annal. of Math. Stat., Vol. 32, pp. 951-989, 1961

55 Rice S.O., "Mathematical Analysis of Random Noise", Bell Sys. Tech. J., Vol. 23,24, reprinted in *Selected Papers on Noise and Stochastic Processes*, Ed. N. Wax, Dover Publications, New York, 1954

56 Baggeroer A.B., "Confidence Intervals for Regression (MEM) Spectral Estimates", IEEE Trans. on Inf. Thy., Vol. IT-22, pp. 534-545, Sep. 1976

57 Anderson T.W. and A.M. Walker, "On the Asymptotic Distribution of the Autocorrelations of a Sample from a Linear Stochastic Process", Ann. Math. Statist., Vol. 35, pp. 1296-1303, 1964

58 Kay S.M., "Noise Compensation for Autoregressive Spectral Estimates", IEEE Trans. on ASSP, Vol. ASSP-28, No. 3, pp. 292-303, June 1980

59 Kailath T., *Lectures on Wiener and Kalman Filtering*, Springer-Verlag, 1981, inc. Appendix 2: "Alternatives In Discrete-Time Recursive Estimation", reprinted from Int. J. Cont., Vol. 32, No. 2, pp. 311- 328, 1980

60 Friedlander et al., "New Inversion Formulas for Matrices Classified in Terms of their Distance from Toeplitz Matrices", Lin. Alg. Appl., Vol. 27, pp. 31-60, 1979

61 Morf M., "Fast Algorithms for Multivariable Systems", Ph.D. thesis, Stanford Univ., 1974

62 Morf M., L. Ljung and T. Kailath, "Fast Algorithms for Recursive Identification", Proc. IEEE Conf. on Decis. and Cont., Florida, Dec. 1976

63 Kalman R.E., "A New Approach to Linear Filtering and Prediction Problems", J. of Basic Eng., pp. 35-45, Mar. 1960

64 Ahmed H.M., M. Morf, D.T. Lee and P.H. Ang, "A VLSI Speech Analysis Chip Set Based on Square-Root Normalised Ladder Forms", Proc. IEEE Int. Conf. of ASSP Soc., pp. 648-653, 1981

65 Volder J.E., "The CORDIC Trigonometric Computing Technique", IRE Trans. on Elect. Comp., Vol. EC-8, pp. 330-339, Sep. 1959

66 Satorius E.H. and S.T. Alexander, "Channel Equalization Using Adaptive Lattice Algorithms", IEEE Trans. on Commun., Vol. COM-27, pp. 899-905, Jun 1979

67 Gibson C. and S. Haykin, "Non-Stationary Learning Characteristics of Adaptive Lattice Filters", Proc. IEEE Int. Conf. of ASSP Soc., pp. 671-674, 1982

68 Bühring W., "An Adaptive Pre-whitening Filter for Radar Clutter Suppression", Proc. of NATO Advanced Study Instit. on Sig. Proc., pp. 141-142, London 1972

69 Skolnik M.I. (Ed.), *Radar Handbook*, McGraw-Hill Book Co., 1970

70 Gradshteyn I.S. and I.M. Ryzhik, *Tables of Integrals, Series and Products*, trans. from Russian by Scripta Technica Inc., Ed. A. Jeffery, Academic Press, New York, 1965

71 Mead C. and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley series in Computer Science, 1980

72 Frost O.L., "An Algorithm for Linearly Constrained Adaptive Array Processing", Proc. IEEE, Vol. 60, No. 8, pp. 926-935, Aug. 1972

73 Gabriel W.F., "Adaptive Arrays- An Introduction", Proc. IEEE, Vol. 64, No. 2, pp. 239-272, Feb. 1976

74 Glover J.R. (Jr.), "Adaptive Noise Cancelling Applied to Sinusoidal Interferences", IEEE Trans. on ASSP, Vol. ASSP-25, No. 6, pp. 484- 491, Dec. 1977

75 Rickard J.T. and J.R. Zeidler, "Second Order Output Statistics of the Adaptive Line Enhancer", IEEE Trans. on ASSP, Vol. ASSP-27, No. 1, pp. 31-39, Feb. 1979

76 Treichler J.R., "Transient and Convergent Behaviour of the Adaptive Line Enhancer", IEEE Trans. on ASSP, Vol. ASSP-27, No. 1, pp. 53-62, Feb. 1979

77 Widrow B., P.E. Mantey, L.J. Griffiths and B.B. Goode, "Adaptive Antenna Systems", Proc. IEEE, Vol. 55, No. 12, pp. 2143-2159, Dec. 1967

78 Satorius E.H., S.W. Larisch, S.C. Lee and L.J. Griffiths, "Fixed-point Implementation of Adaptive Digital Filters", Proc. of the IEEE Int. Conf. of ASSP Soc., pp. 33-35, Boston, 1983

79 Samson C. and V.U. Reddy, "Fixed Point Error Analysis of the Normalised Ladder Algorithm", Proc. IEEE Int. Conf. of the ASSP Soc., pp. 1752-1755, 1982

80 Done W.J., "Use of the Fast Kalman Estimation Algorithm for Adaptive System Identification", Proc. IEEE Int. Conf. of the ASSP Soc., pp. 886-889, 1981

81 Brennan L.E. and L.S. Reed, "Theory of Adaptive Radar", IEEE Trans. on Areosp. and Elect., Vol. AES-9, No. 2, pp. 237-252, March 1973

82 Lacoss R.T., "Data Adaptive Spectral Analysis Methods", Geophysics, Vol. 36, pp. 661-675, Aug. 1971

83 Widrow B. and J.M. McCool, "A Comparison of Adaptive Algorithms based on Steepest Descent and Random Search", IEEE Trans. on Ant. and Prog., Vol. AP-24, No. 5 (part of special issue on Adaptive Antennas), pp. 615-637, Sept. 1976

84 Wiener N., *Extrapolation, Interpolation and Smoothing of Stationary Time Series*, John Wiley and Sons Inc., New York, 1949

85 Oppenheim A.V. and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975

86 Eshraghian K., A. Dickinson, J. Rockcliff, G. Zyner, R.E. Bogner, R.C. Bryant, W.G. Cowley, B.R. Davis and D.S. Fensom, " A New CMOS VLSI Architecture for Signal Processing", submitted to VLSI Pacific Asian Region Conference (PARC), Melbourne, May 1984

List of Principal Authors

Ables	27	Griffiths	4
Ahmed	64	Griffiths	33
Akaike	52	Halkias	42
Anderson	57	Haykin	14
Baggeroer	56	Haykin	22
Berg	26	Hodgkiss	9
Box	5	Honig	37
Brennan	81	Honig	36
Bühring	68	Hynes	48
Clarke	29	Itakura	35
Davis	19	Kailath	59
Demytko	18	Kailath	24
Done	80	Kalman	63
Eshraghian	86	Kay	58
Falconer	41	Kay	28
Friedlander	13	Kim	30
Friedlander	60	Lacoss	82
Friedlander	12	Larimore	39
Frost	72	Lee	16
Gabriel	73	Lee	6
Gibson	67	Ljung	46
Gibson	17	Makhoul	31
Gibson	7	Makhoul	8
Gibson	34	Makhoul	1
Glover	74	Makhoul	32
Goodwin	40	Mann	53
Gradshteyn	70	Mead	71

List of Principal Authors (continued)

Medaugh	49
Medaugh	50
Morf	51
Morf	43
Morf	62
Morf	61
Oppenheim	85
Papoulis	38
Parzen	54
Porat	10
Ralston	21
Rice	55
Rickard	75
Samson	79
Satorius	66
Satorius	44
Satorius	78
Shensa	47
Skolnik	69
Soderstrom	45
Stearns	15
Treichler	76
Ulrych	23
Van	25
Volder	65
Widrow	3
Widrow	83
Widrow	77
Widrow	2
Wiener	84
Youn	11
Zeidler	20