NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ADAPTIVE TWO DIMENSIONAL
RLS ALGORITHMS

by

Armando M. P. de Jesus Sequeira

March 1989

Thesis Advisor Charles W. Therrien

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings	
Security Classification Authority		3 Distribution Availability of Report Approved for public release: distribution is unlimited.	
Declassification Downgrading Schedule			
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
Name of Performing Organization val Postgraduate School	6b Office Symbol *(if applicable)* 32	7a Name of Monitoring Organization Naval Postgraduate School	
Address *(city, state, and ZIP code)* onterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No	Project No / Task No / Work Unit Accession No

Title *(include security classification)* ADAPTIVE TWO DIMENSIONAL RLS ALGORITHMS

Personal Author(s) Armando M. P. de Jesus Sequeira

Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
ster's Thesis	From To	March 1989	93

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or po-
on of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
d	Group	Subgroup	RLS, adaptive, two dimensional, arma, geometric

Abstract *(continue on reverse if necessary and identify by block number)*

A Two-Dimensional Fast Recursive Least Squares (2-D FRLS) algorithm is presented using a geometrical formulation
ed on the mathematical concepts of vector space, orthogonal projection, and subspace decomposition.
By appropriately ordering the 2-D data, the algorithm provides an exact least-squares solution to the deterministic Normal
iations. The method is further extended to the general FIR Wiener filter and to ARMA modeling. The size and shape
he support region for both the MA and AR coefficients of the filter can be choosen arbitrarly. The ARMA parameter
mation problem is also considered for the case when the system input is not available.
Computer simulations are presented to illustrate the applications of the algoritm for 2-D parameter estimation, system
ntification and image coding.

Distribution Availability of Abstract nclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
Name of Responsible Individual rles W. Therrien	22b Telephone *(include Area code)* (408) 646-3347	22c Office Symbol 62Ti

FORM 1473,84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page
Unclassified

Approved for public release; distribution is unlimited.

Adaptive Two Dimensional RLS Algorithms

by

Armando M. P. de Jesus Sequeira
Lieutenant, Portuguese Navy
B.S.E.E., Portuguese Naval Academy, 1981

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
ELECTRICAL ENGINEER

from the

ABSTRACT

A Two-Dimensional Fast Recursive Least Squares (2-D FRLS) algorithm is presented using a geometrical formulation based on the mathematical concepts of vector space, orthogonal projection, and subspace decomposition.

By appropriately ordering the 2-D data, the algorithm provides an exact least-squares solution to the deterministic Normal equations. The method is further extended to the general FIR Wiener filter and to ARMA modeling. The size and shape of the support region for both the MA and AR coefficients of the filter can be choosen arbitrarly. The ARMA parameter estimation problem is also considered for the case when the system input is not available.

Computer simulations are presented to illustrate the applications of the algoritm for 2-D parameter estimation, system identification and image coding.

C.1

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

I wish to express my sincere appreciation to Professor Charles W. Therrien, my thesis advisor, for his assistance, patience and all the encouragement provided during the preparation of this thesis. I also want to thank Professor Murali Tummala, my second reader, for his availability and support. Last, but certainly not least, I wish to thank my wife Maria do Céu who made our stay in Monterey wonderful.

I. INTRODUCTION

Adaptive algorithms have been used successfully for many years in a wide range of digital signal processing applications involving non-stationary data. In these applications it is desired to follow closely the variations of the parameters characterizing the process, by updating (in real time if possible) estimates of these parameters as soon as new data is available. Real time implementation of these algorithms only recently became possible with the latest capabilities of the VLSI technology and is partly a result of the development of computationally affordable algorithms based on a very elegant mathematical formulation. This formulation is known as fast recursive least squares (FRLS) and is based on a geometric approach. The derivation of algorithms based on the geometric approach uses the concepts of linear vector spaces, orthogonality, projection matrices, and their relation with least squares prediction [Ref. 1, 2, 3]. Motivation for the development of similar algorithms to process two-dimensional (2-D) data is a consequence of the very interesting results reported lately in the literature on adaptive filters for one-dimensional (1-D) signals in what concerns their reduced complexity and excellent behavior even in non-stationary environments [Ref. 4, 5]. The development of Fast RLS algorithms for 2-D data based on the geometric approach is what is addressed in this thesis.

A. PROBLEM FORMULATION

A major problem with the extension of Fast RLS algorithms to two dimensions is that causality is not inherent in 2-D systems. This problem was overcome by associating the past of a 2-D signal with the region of support of a recursive filter mask (usually quarter plane or non-symmetrical half plane). By appropriately ordering the 2-D data, a two-dimensional Fast Recursive Least Squares (2-D FRLS) algorithm

is developed using a geometrical formulation where the vector spaces and all the notions associated with them are defined to reflect the 2-D nature of the data. An exact least squares solution to the deterministic Normal equations is provided for all of the all-pole (AR), all-zero (MA), or pole-zero (ARMA) models. The size and shape of the support region for both the MA and AR coefficients of the filter can be chosen arbitrarily as long as the overall system is recursively computable. The ARMA parameter estimation problem is also addressed for the case when the system input is not known.

B. THESIS OVERVIEW

The remainder of this thesis is organized as follows. Chapter 2 provides a summary of the most common adaptive filtering techniques. Only brief reference is made the Least Mean Square (LMS) algorithm due to its simplicity and slower convergence properties. However a 2-D version of this technique that was recently reported in the literature is mentioned. Most of the chapter reviews the basic ideas of the Recursive Least Squares (RLS) algorithm. This provides preparation for chapter 3 where a fast 2-D version of this algorithm is developed in detail.

Chapter 4 is dedicated to applications of the new algorithm to the 2-D problems of systems identification, image coding, and parameter estimation. Different models (AR, MA, and ARMA) are considered.

Chapter 5 summarizes the results obtained and suggests some possible improvements for the new algorithm. Mathematical derivations related to the geometrical formulation that are essential to the method, but too tedious to be inserted in the body of the thesis, are grouped in the Appendix.

II. ADAPTIVE FILTERS

A. PERFORMANCE CRITERIA

In adaptive filtering the performance criterion is usually based on the minimization of a cost function dependent on the filter coefficients to be determined. The most common performance criterion is the minimization of the mean square error (MSE) associated with the signal to be estimated [Ref. 5, 6, 7]. In particular, if we consider a random process $y(n)$ and a predictive filter of the form

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

where $\hat{y}(n)$ is the predicted value and $h_n(k)$ are the filter coefficients, then the prediction error is defined by

$$e(n) = y(n) - \hat{y}(n) \tag{2}$$

and the MSE becomes

$$\epsilon = E[e^2(n)] \tag{3}$$

where E is the expectation operator. For stationary data, this quantity is a convex quadratic function of the filter coefficients $h_n(k)$ and attains its minimum at a point where the partial derivatives with respect to each of the filter coefficients are simultaneously equal to zero. Substituting (1) and (2) in (3) and simplifying, we obtain the desired expression

$$\frac{\partial \epsilon}{\partial h_n(k)} = -2E[e(n)y(n-k)] = 0 \quad \text{for } k = 1, ..., M \tag{4}$$

The dependence of the performance criterion on the filter coefficients can be interpreted in terms of a multidimensional convex surface with a unique minimum. This

surface is called the error-performance surface. The coefficients associated with the minimum mean-square error are obtained by solving the set of simultaneous equations in (4) which in this case are known as the Normal equations.

B. LEAST MEAN SQUARE (LMS) ALGORITHM

1. 1-D LMS

The Normal equations can be solved by *brute force* using matrix inversion or by computationally faster methods such as the Levinson-Durbin algorithm for Toeplitz matrices. However here, we are interested in a method called the *steepest descent*, which provides an iterative solution to the Normal equations [Ref. 5, 6, 7, 8]. We start with a initial set of filter coefficients and a corresponding point on the error performance surface. We then compute the gradient vector formed by the partial derivatives of the mean-squared error with respect to each of the filter coefficients at that point. Using (4) the gradient vector can be expressed as

$$\nabla(n) = -2E[\epsilon(n)\underline{\mathbf{y}}(n)] \tag{5}$$

where $\underline{\mathbf{y}}(n)$ is a $M \times 1$ vector that contains the data covered by the filter mask at time n

$$\underline{\mathbf{y}}(n) = [y(n-1), y(n-2), ..., y(n-M)]^T \tag{6}$$

Finally we update the coefficients by changing them in a direction opposite to that of the gradient vector using a predefined step size μ

$$\underline{\mathbf{h}}_{n+1} = \underline{\mathbf{h}}_n + \frac{1}{2}\mu[-\nabla(n)] = \underline{\mathbf{h}}_n + \mu E[\epsilon(n)\underline{\mathbf{y}}(n)] \tag{7}$$

where $\underline{\mathbf{h}}_n$ is a $M \times 1$ vector that contains the filter coefficients at time n

$$\underline{\mathbf{h}}_n = [h_n(1), h_n(2), ..., h_n(M)]^T \tag{8}$$

The inconvenience of this approach is that it requires an exact measure of the gradient vector at each iteration and the gradient involves statistical expectation. Usually the

statistics of the data are not available and must be estimated from the raw data. Thus to obtain a really accurate estimate of the gradient is quite cumbersome and computationally expensive.

A practical method to estimate the gradient vector in a simple manner directly from acquired data is used in the Least Mean Squares (LMS) algorithm of Widrow [Ref. 5]. For this, the expectation in (5) is ignored and an instantaneous estimate of the gradient vector is taken to be

$$\hat{\nabla}(n) = -2e(n)\underline{\mathbf{y}}(n) \tag{9}$$

The update for the filter coefficients then has the form

$$\underline{\mathbf{h}}_{n+1} = \underline{\mathbf{h}}_n + \frac{1}{2}\mu[-\hat{\nabla}(n)] = \underline{\mathbf{h}}_n + \mu e(n)\underline{y}(n) \tag{10}$$

This method is quite attractive for a wide range of applications since it requires no matrix inversions, correlation function estimation, or (actual) gradient computation, and hence has low computational complexity. However its convergence is relatively slow. A detailed derivation and analysis of the LMS properties can be found in [Ref. 5, 6].

2. 2-D LMS

The extension of the LMS algorithm to 2-D signals is straightforward. Reference [Ref. 9] gives a detailed derivation and shows that the analysis presented by other authors for the 1- D LMS is also applicable to the 2-D version of the algorithm.

The final form of the 2-D LMS algorithm is very similar to (10) but the vector containing the 1-D filter coefficients $\underline{\mathbf{h}}_n$ is substituted by a matrix $\mathbf{W}_j$ containing the 2-D filter coefficients. The instantaneous estimate of the 2-D gradient uses the data matrix $\mathbf{X}_j$ formed by the 2-D input samples covered by the 2-D filter mask at iteration j. For a $N \times M$ 2-D sequence, at sample $y(n, m)$; if j is the linear

scanning index

$$j = mM + n \tag{11}$$

then the algorithm takes the form:

$$\mathbf{W}_{j+1} = \mathbf{W}_j + \frac{1}{2}\mu[-\hat{\nabla}(n)] = \mathbf{W}_j + \mu e_j \mathbf{X}_j \tag{12}$$

A separate derivation of this algorithm was also presented in [Ref. 10], together with some examples of its performance through computer simulation of a noise canceler and an adaptive line enhancer applied to an image processing problem.

C. 1-D RECURSIVE LEAST SQUARES (RLS)

In the adaptive methods presented above the need to solve the Normal equations appears as a consequence of the minimization of a statistical cost function, the mean-squared error. However the implementation instead uses the actual data to compute errors and update the coefficients. This is the main cause of the performance deficiencies encountered when implementing this algorithm.

Another possible approach is to base the performance criterion upon error measures derived from the actual data. This class of techniques is known generally as Least Squares (LS) algorithms.

The LS algorithm is designed to find the set of filter coefficients that minimize the cumulative sum of squared errors.

$$\epsilon(n) = \sum_{i=1}^{n} \epsilon^2(i) \tag{13}$$

Although this seems very similar to the previous performance criterion, it results in a set of deterministic Normal equations whose solution provides filter coefficients that are exactly optimal, according to (13), for the acquired data instead of statistically optimal for a class of data as in the case of the *steepest descent* methods [Ref. 6].

To formulate this problem we once again differentiate the cost function with respect to the coefficients. However since we here use summations instead of expectations the result is

$$\frac{\partial \epsilon(n)}{\partial h_n(k)} = -2r_n(0,k) + 2\sum_{l=1}^{M} h_n(l) r_n(k,l) = 0 \quad \text{for } k = 1, ..., M \tag{14}$$

where we define the deterministic correlation function $r_n(k,l)$ as

$$r_n(k,l) = \sum_{i=1}^{n} y(n-k)y(n-l) \quad \text{for } k, m = 0, ..., M \tag{15}$$

This set of M simultaneous equations are the deterministic Normal equations. The equations are written in matrix form as

$$\mathbf{R}(n)\underline{\mathrm{h}}_n = \underline{\mathrm{r}}(n) \tag{16}$$

where $\mathbf{R}(n)$ is the $M \times M$ deterministic correlation matrix with the structure

$$\mathbf{R}(n) = \begin{bmatrix} r_n(1,1) & r_n(1,2) & \cdots & r_n(1,M) \\ r_n(2,1) & r_n(2,2) & \cdots & r_n(2,M) \\ \vdots & \vdots & \ddots & \vdots \\ r_n(M,1) & r_n(M,2) & \cdots & r_n(M,M) \end{bmatrix} \tag{17}$$

and $\underline{\mathrm{r}}(n)$ is the $M \times 1$ vector of deterministic cross-correlation terms between the desired filter response and the filter inputs.

$$\underline{\mathrm{r}}(n) = [r_n(0,1), r_n(0,2), \cdots, r_n(0,M)]^T \tag{18}$$

If $\mathbf{R}(n)$ is nonsingular then the solution to the Normal Equations is formally

$$\underline{\mathrm{h}}_n = \mathbf{R}^{-1}(n)\underline{\mathrm{r}}(n) \tag{19}$$

This *brute force* solution requires on the order of M^3 arithmetic operations. A better approach is to use a method known as the recursive least squares (RLS). This uses the Matrix Inversion Lemma [Ref. 5] to update the inverse correlation matrix and the

cross correlation vector as new data is acquired and thus to compute $\underline{\mathbf{h}}_n$ recursively. The resulting expression for $\mathbf{R}^{-1}(n)$ is

$$\mathbf{R}^{-1}(n) = \mathbf{R}^{-1}(n-1) - \frac{\mathbf{R}^{-1}(n-1)\underline{\mathbf{y}}(n)\underline{\mathbf{y}}^T(n)\mathbf{R}^{-1}(n-1)}{1 + \underline{\mathbf{y}}^T(n)\mathbf{R}^{-1}(n-1)\underline{\mathbf{y}}(n)} \tag{20}$$

By defining the a priori error $e(n|n-1)$ as

$$e(n|n-1) = y(n) - \underline{\mathbf{y}}^T(n)\underline{\mathbf{h}}_{n-1} \tag{21}$$

and the gain vector $\underline{\mathbf{k}}(n)$ as

$$\underline{\mathbf{k}}(n) = \frac{\mathbf{R}^{-1}(n-1)\underline{\mathbf{y}}(n)}{1 + \underline{\mathbf{y}}^T(n)\mathbf{R}^{-1}(n-1)\underline{\mathbf{y}}(n)} \tag{22}$$

we can rewrite (20) as

$$\mathbf{R}^{-1}(n) = \mathbf{R}^{-1}(n-1) - \underline{\mathbf{k}}(n)\underline{\mathbf{y}}^T(n)\mathbf{R}^{-1}(n-1) \tag{23}$$

If we then use $\underline{\mathbf{r}}(n) = \underline{\mathbf{r}}(n-1) + y(n)\underline{\mathbf{y}}(n)$ and substituting in (19) the desired update for coefficient vector $\underline{\mathbf{h}}_n$ is found to be

$$\underline{\mathbf{h}}_n = \underline{\mathbf{h}}_{n-1} + \underline{\mathbf{k}}(n)e(n|n-1) \tag{24}$$

To update the coefficient vector as new data is acquired all we must do is to compute the last four equations assuming that all the parameters with index $n-1$ are available at time n. Since the non-singularity of the deterministic correlation matrix is a requirement for the solution of the problem, we must start with the initial condition

$$\mathbf{R}^{-1}(0) = c^{-1}\mathbf{I}_{M\times M} \tag{25}$$

where c is a small positive constant. It also customary to initialize all the components of the coefficient filter $\underline{\mathbf{h}}_n$ to zero.

The RLS algorithm is computationally more expensive than the LMS. The RLS algorithm requires a total of $3M(3 + M)/2$ multiplications/divisions per iteration,

while the LMS algorithm requires only $2M + 1$. On the other hand the convergence performance of the RLS algorithm is much superior [Ref. 5, 6]. A detailed derivation of this algorithm and its overall performance can be found in many references such as [Ref. 5, 6, 8].

An enormous reduction of the computational complexity of the RLS method is obtained by using a geometrical formulation in its derivation. The computational complexity of the algorithm is reduced to approximately $6M$ arithmetic operations. The geometrical approach also provides an interesting interpretation of the prediction problem in terms of the concepts of vector spaces and orthogonality. Since we will use an expanded version the geometrical formulation to derive the extension of this method to 2-D signals, and the derivation is lengthy, we will not derive the 1-D method here. However a very comprehensive explanation of the geometrical approach for 1-D signals can be found in [Ref. 6].

For the case of nonstationary data it is frequently advantageous to incorporate a forgetting factor in the cost function

$$\epsilon(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i) \quad \text{for } 0 < \lambda \leq 1 \tag{26}$$

The interpretation of this forgetting factor can be understood as an exponential windowing of the data in a fashion such that the most recent data has a heavier influence in the cost function to be minimized. The fast algorithm is mathematically equivalent to the RLS, hence its stability is guaranteed in theory for any possible forgetting factor λ [Ref. 7]. However the efficiency of this class of algorithms is a result of the reduced number of variables used to represent quantities such as the inverse deterministic correlation matrix. Due to the finite precision arithmetic used, the representation is only approximate. As a result the accumulation of round-off errors can set off instability of the algorithm.The sensitivity of some quantities to round-off error are highly

dependent on the forgetting factor used. This imposes a lower bound on λ. Typical values for λ are in the range:

$$0.95 \leq \lambda < 1.0 \tag{27}$$

III. 2-D FAST RECURSIVE LEAST SQUARES

In this chapter we present the derivation of the 2-D FRLS algorithm. As mentioned before we use the geometrical formulation to obtain a fast, computationally efficient algorithm.

We start by introducing a convenient notation that closely follows the one used by Alexander [Ref. 6] for the geometric derivation of the 1-D FRLS. This is followed by a brief set of vector space considerations that are the basis of the problem solution. Next some auxiliary filters that use the same data as the 2-D filter are introduced. The key to this method is to find a relation between the parameters of these filters that permits the recursive update of all of them as soon as new data is available.

A. 2-D PREDICTION FILTERS

The method to be described applies to a general 2-D prediction filter of the form

$$\hat{y}(n_1, n_2) = \sum_i \sum_j a_{ij}\, y(n_1 - i, n_2 - j) \tag{28}$$

with (i, j) defined in a region that allows the 2-D AR model related to the prediction error process to be recursively computable (ex: quadrant or non-symmetric half plane support). The recursive computability is a requirement for applications where inverse filtering is used to recover the 2-D data sequence from the estimated error sequence as in most of the image coding schemes. To be specific and develop clear notation we will assume a first quadrant $(N + 1) \times (M + 1)$ filter of the form

$$\hat{y}(n_1, n_2) = \sum_{i=0}^{N} \sum_{j=0}^{M} a_{ij}\, y(n_1 - i, n_2 - j) \qquad (i, j) \neq (0, 0) \tag{29}$$

and a 2-D data sequence with $K \times L$ points as shown in Figure 1.

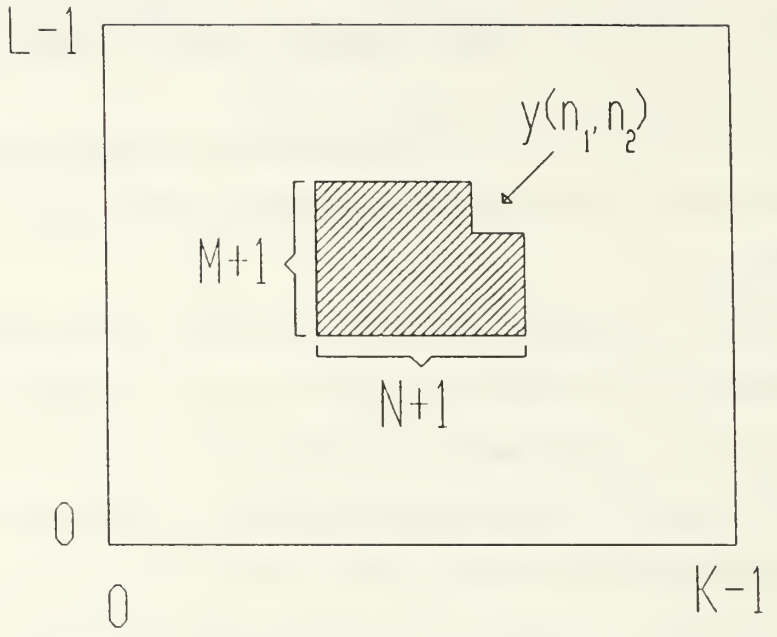

Figure 1. First Quadrant (N+1)(M+1) Filter

The prediction error $e(n_1, n_2) = y(n_1, n_2) - \hat{y}(n_1, n_2)$ can be expressed using vector notation as

$$e(n_1, n_2) = y(n_1, n_2) - \underline{\mathbf{y}}^T(n_1, n_2)\,\underline{\mathbf{a}} \tag{30}$$

where

$$\begin{aligned} \underline{\mathbf{y}}(n_1, n_2) \;=\; & [y(n_1 - 1, n_2), \cdots, y(n_1 - N, n_2), y(n_1, n_2 - 1), \cdots \\ & \cdots, y(n_1 - N, n_2 - 1), \cdots, y(n_1 - N, n_2 - 2), \cdots \\ & \cdots, y(n_1, n_2 - M), \cdots, y(n_1 - N, n_2 - M)]^T \end{aligned} \tag{31}$$

is a $(N + 1)(M + 1) - 1$ dimensional vector formed by the elements covered by the filter mask, ordered along rows, and

$$\underline{\mathbf{a}} = [a_{10}, \cdots, a_{N0}, a_{01}, \cdots, a_{N1}, \cdots, a_{0M}, \cdots, a_{NM}]^T \tag{32}$$

is the vector of 2-D filter coefficients with the same ordering. We assume for now that $y(k, l) = 0$ for $k < 0$ or $l < 0$.

When performing linear prediction along one of the possible directions of recursion (e.g., along rows), and all of the data necessary for each prediction is available, the *optimal filter* (least squares criterion) up to point (n_1, n_2), will be defined as the one having a set of coefficients $a_{ij}(n_1, n_2)$, ($0 \leq i \leq N$, $0 \leq j \leq M$, $(i,j) \neq (0,0)$) that minimizes the sum of the squared errors along that recursion direction.

$$\epsilon(n_1, n_2) = \sum_{i=0}^{n_1} [e(i, n_2)]^2 + \sum_{i=0}^{K-1} \sum_{j=0}^{n_2-1} [e(i,j)]^2 \tag{33}$$

B. 2-D DATA ORDERING

One question that arises whenever we deal with finite extent sequences is what to do when we approach the boundaries of the 2- D data sequence and the prediction mask needs to cover points that lie outside of the region where the data is defined. One approach is to set the points outside of this region (i.e., the boundary conditions) to zero. The inconvenience of this approach is that the boundary conditions depend not only on the extent of the 2-D sequence but also on the shape of the filter mask and this can lead to additional complications [Ref. 11]. In addition, when we reach the end of a row and we start a new one, the data under the mask is almost all reset to zero. This causes a strong discontinuity in the process.

An alternative approach is to assume that, although the 2-D data we process may not be stationary, the statistical properties of the data do not vary too rapidly, and so to use the data at the end of one row as the initial condition for prediction along the next row as is shown in Figure 2. This appears to be at least as reasonable as the first approach. It will be seen later that this approach also has several advantages in deriving an algorithm based on the geometrical approach. From a practical point of view, it is as if we fold the 2-D data plane and form a cylinder with perimeter equal to K, but the data rows instead of folding into themselves, are misaligned by one row. This allows the prediction to be performed along rows with the 2-D mask moving from bottom to top in a helical fashion.

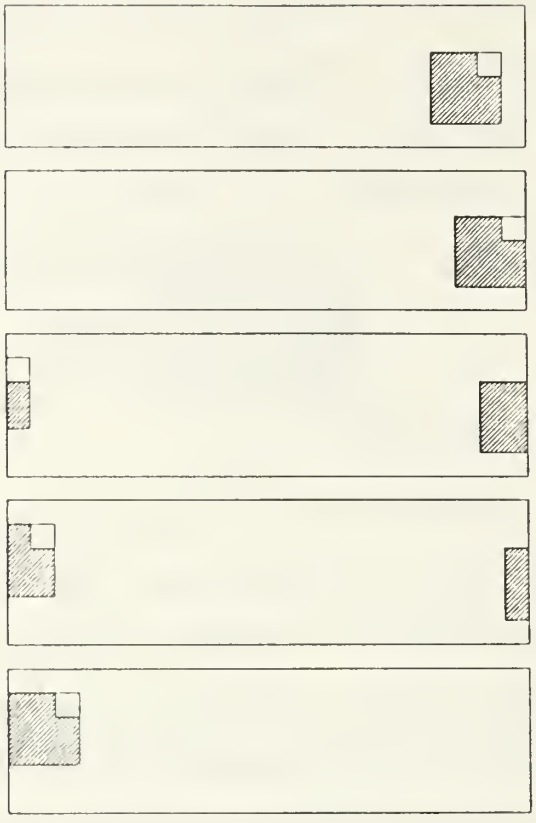

Figure 2. Data Ordering

C. PREVIOUS DATA/ PAST OBSERVATIONS

Performing linear prediction on a 2-D signal along rows implies that the new data comes only from a strip with the width of the filter mask $(M+1)$. This suggests an analogy with the $(M+1)$ channel 1-D prediction problem.

1. (M+1) Channel Analogy

The $(M+1)$ rows of the data strip can be viewed as $(M+1)$ channels of a 1-D signal. To support this idea, define a linear index n such that

$$n = n_2 \times K + n_1 \tag{34}$$

Then the 2-D data sequence $y(n_1, n_2)$ can be expressed in terms of the data in the first channel as

$$y(n_1, n_2) = y_1(n) \tag{35}$$

We can also express the data in the other channels in terms of $y_1(n)$ as

$$y_i(n) = y_1(n - (i-1)K) \qquad \text{for } i = 1, ..., M+1 \tag{36}$$

We will predict along the first channel using data from all channels defined by the 2-D filter mask. A consequence of this approach is that the length of data used from each channel depends on the shape of the 2-D mask. Figure 3 shows the particular case of a Quarter Plane mask. In order to predict $y_1(n) = y(n_1, n_2)$, the data used is formed by N samples of channel 1 (y_1) and (N+1) samples of channels 2 (y_2) to $M+1$ (y_{M+1}). This requires a total of $(N+1) \times (M+1) - 1$ coefficients as in the 2-D mask.

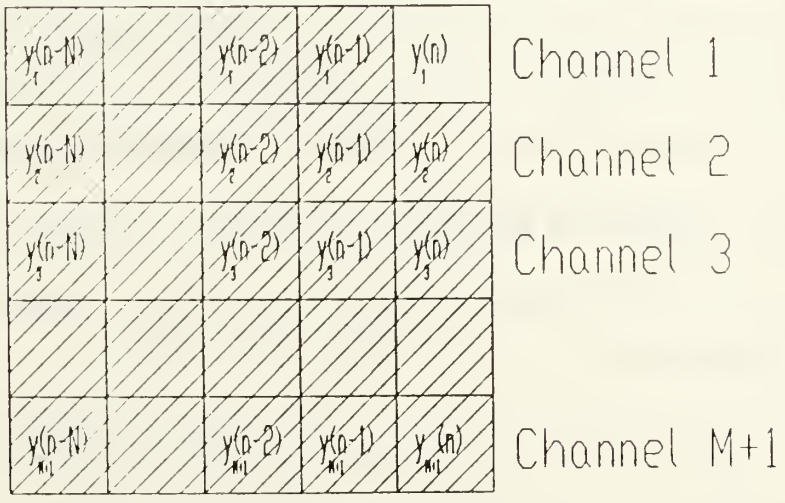

Figure 3. M+1 Channel Analogy

2. Past data

The multichannel analogy will be used to define what we call the *past data* and the *observation data* because the notion of multichannel prediction will be useful later. From now on we will drop the 2-D notation and use the index n associated with the recursion along the rows.

Begin by defining the $(n+1)$-dimensional vector

$$\underline{\mathbf{y}}_i(n) = [y_i(0), y_i(1), \cdots\cdots, y_i(n)]^T \qquad 1 \le i \le M+1 \tag{37}$$

that contains data in channel i up to the point n. Further define the delay operator z^{-k} such that

$$z^{-k}\underline{\mathbf{y}}_i(n) = [0.0, \cdots, y_i(0), y_i(1), \cdots, y_i(n-k)]^T \tag{38}$$

is a $(n+1)$-dimensional vector that contains the data of $\underline{\mathbf{y}}_i(n)$ delayed by k samples and pre-windowed. We call $\underline{\mathbf{y}}_1(n)$ the *observation* data since it contains the data sequence we desire to predict. The *past* data with respect to $y_1(n)$ is formed by all of the points $y_i(j)$ such that $(2 \le i \le M+1, 0 \le j \le n)$ or $(i = 1, 0 \le j \le n-1)$ as shown in Figure 4. Note that different channels have data in common. With the new notation defined we are ready to reformulate the problem.

D. PROBLEM REFORMULATION

We start by redefining (31), the vector that contains the data covered by the 2-D filter mask, as

$$\underline{\mathbf{y}}_{1,N}(n) = [y_1(n-1), \cdots, y_1(n-N), y_2(n), \cdots, y_{M+1}(n), \cdots, y_{M+1}(n-N)]^T \tag{39}$$

where the subscript (1,N) denotes the fact that the data in the first channel appears delayed by 1 to N samples. We want to find a prediction filter of the form

$$\hat{y}_1(n) = \underline{\mathbf{y}}_{1,N}^T(n)\,\underline{\mathbf{a}}(n) \tag{40}$$

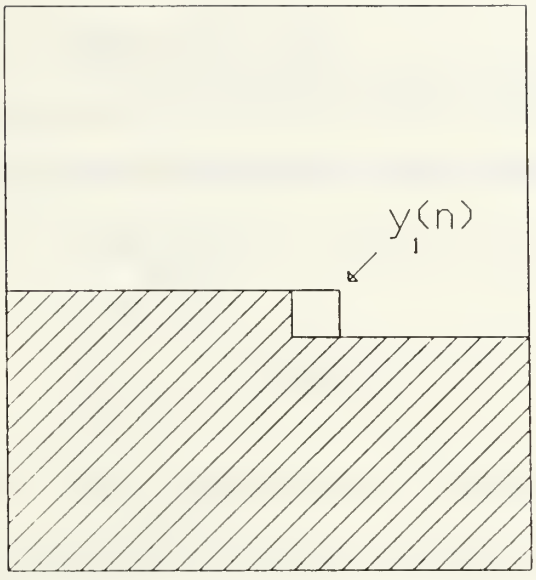

Figure 4. Past Data

with

$$\underline{\mathbf{a}}(n) = [a_{10}(n), \cdots, a_{N0}(n), a_{01}, \cdots, a_{N1}(n), \cdots, a_{0M}(n), \cdots, a_{NM}(n)]^T \tag{41}$$

that minimizes the sum of squared errors

$$\epsilon_1(n) = \sum_{i=0}^{n} [e_1(i)]^2 \tag{42}$$

where the prediction error given by

$$e_1(n) = y_1(n) - \hat{y}_1(n) \tag{43}$$

This can be written in vector notation as

$$\epsilon_1(n) = \underline{\mathbf{e}}_1^T(n)\underline{\mathbf{e}}_1(n) \tag{44}$$

where

$$\underline{\mathbf{e}}_1(n) = \underline{\mathbf{y}}_1(n) - \underline{\hat{\mathbf{y}}}_1(n) \tag{45}$$

and

$$\hat{\underline{y}}_1(n) = \mathbf{Y}_{1,N}(n)\,\underline{\mathbf{a}}(n) \tag{46}$$

where $\mathbf{Y}_{1,N}(n)$ is a data matrix formed by the data covered by the 2-D mask from the origin up to time n. The matrix $\mathbf{Y}_{1,N}(n)$ can be written as

$$\mathbf{Y}_{1,N}(n) = \begin{bmatrix} \underline{\mathbf{y}}^T_{1,N}(0) \\ \underline{\mathbf{y}}^T_{1,N}(1) \\ \vdots \\ \vdots \\ \underline{\mathbf{y}}^T_{1,N}(n) \end{bmatrix} \tag{47}$$

or using a different partition, $\mathbf{Y}_{1,N}(n)$ can alternatively be written in terms of the data in the $M+1$ channels and their delayed versions (37) and (38) as

$$\mathbf{Y}_{1,N}(n) = \left[z^{-1}\underline{\mathbf{y}}_1(n), \cdots, z^{-N}\underline{\mathbf{y}}_1(n), \underline{\mathbf{y}}_2(n), z^{-1}\underline{\mathbf{y}}_2(n), \cdots, z^{-N}\underline{\mathbf{y}}_{M+1}(n)\right] \tag{48}$$

As will be shown later, a necessary initial condition for the geometric formulation to work is that we start at a sample $y_1(0)$ such that $\underline{\mathbf{y}}_{1,N}(0) = \underline{\mathbf{0}}$. That is, the initial conditions for the data under the 2-D mask must be zero. Now let us proceed with the minimization of (42). The least squares solution for $\underline{\mathbf{a}}(n)$ is given by the pseudo-inverse:

$$\underline{\mathbf{a}}(n) \;=\; \left(\mathbf{Y}^T_{1,N}(n)\mathbf{Y}_{1,N}(n)\right)^{-1}\mathbf{Y}^T_{1,N}(n)\underline{\mathbf{y}}_1(n) \tag{49}$$

where $\left(\mathbf{Y}^T_{1,N}(n)\mathbf{Y}_{1,N}(n)\right)^{-1}$ can be interpreted as the inverse of a 2-D deterministic correlation matrix and $\mathbf{Y}^T_{1,N}(n)\underline{\mathbf{y}}_1(n)$ can be interpreted as the vector of the deterministic cross-correlations between the *observation* data and the *past* data. A new expression for $\underline{\mathbf{e}}_1(n)$ is obtained substituting the solution for $\underline{\mathbf{a}}(n)$ in (46) and using the result in (45). This yields

$$\begin{aligned} \underline{\mathbf{e}}_1(n) &= \underline{\mathbf{y}}_1(n) - \mathbf{P}_{1,N}(n)\underline{\mathbf{y}}_1(n) \\ &= \left(\mathbf{I} - \mathbf{P}_{1,N}(n)\right)\underline{\mathbf{y}}_1(n) \end{aligned} \tag{50}$$

where

$$\mathbf{P}_{1,N}(n) = \mathbf{Y}_{1,N}(n)\left(\mathbf{Y}_{1,N}^T(n)\mathbf{Y}_{1,N}(n)\right)^{-1}\mathbf{Y}_{1,N}^T(n) \tag{51}$$

is interpreted as the projection matrix that projects vectors into the subspace spanned by the columns of $\mathbf{Y}_{1,N}(n)$. (Note here that it is assumed that $\left(\mathbf{Y}_{1,N}^T(n)\mathbf{Y}_{1,N}(n)\right)^{-1}$ exists.) Also define

$$\mathbf{P}_{1,N}^{\perp}(n) = \mathbf{I} - \mathbf{P}_{1,N}(n) \tag{52}$$

as the orthogonal projection matrix associated with the same subspace.

Both the L.S. estimate of $\underline{\mathbf{y}}_1(n)$ and the prediction error can now be expressed in terms of the projections matrices. The estimate $\hat{\underline{\mathbf{y}}}_1(n)$ is the projection of $\underline{\mathbf{y}}_1(n)$ onto the subspace spanned by the *previous* data

$$\hat{\underline{\mathbf{y}}}_1(n) = \mathbf{P}_{1,N}(n)\underline{\mathbf{y}}_1(n) \tag{53}$$

The error $\underline{\mathbf{e}}_1(n)$ is orthogonal to the estimate $\hat{\underline{\mathbf{y}}}_1(n)$.

$$\underline{\mathbf{e}}_1(n) = \mathbf{P}_{1,N}^{\perp}(n)\underline{\mathbf{y}}_1(n) \tag{54}$$

Next, we define the operator $\mathbf{K}_{1,N}(n)$ as

$$\mathbf{K}_{1,N}(n) = \left(\mathbf{Y}_{1,N}^T(n)\mathbf{Y}_{1,N}(n)\right)^{-1}\mathbf{Y}_{1,N}^T(n) \tag{55}$$

hence

$$\underline{\mathbf{a}}(n) = \mathbf{K}_{1,N}(n)\underline{\mathbf{y}}_1(n) \tag{56}$$

$\mathbf{K}_{1,N}(n)$ can be interpreted as the operator that computes the *best LS* filter $\underline{\mathbf{a}}(n)$ for predicting $\underline{\mathbf{y}}_1(n)$ given the data set $\mathbf{Y}_{1,N}(n)$. Now since $\underline{\mathbf{y}}_1(n)$ can be obtained from $\underline{\mathbf{y}}_1(n-1)$ as soon as $y_1(n)$ is available, if we find an efficient way to get $\mathbf{K}_{1,N}(n)$ from $\mathbf{K}_{1,N}(n-1)$ then we will be able to update $\underline{\mathbf{a}}(n-1)$ to $\underline{\mathbf{a}}(n)$.

E. VECTOR SPACE CONSIDERATIONS

Before going any further we must present some concepts and relations associated with vector spaces that will be needed later on. To make the results generic, in this section our data matrix is called $\mathbf{U}(n)$. This generic data matrix can represent the matrix $\mathbf{Y}_{1,\mathcal{N}}(n)$ defined in (47) and (48) and other similar matrices that are defined later in this chapter. Then following the definition of (51) and (52), the projection matrix associated with $\mathbf{U}(n)$ is

$$\mathbf{P}_{\mathbf{U}}(n) = \mathbf{U}(n)\left(\mathbf{U}^T(n)\mathbf{U}(n)\right)^{-1}\mathbf{U}^T(n) \tag{57}$$

and the orthogonal projection matrix is

$$\mathbf{P}_{\mathbf{U}}^{\perp}(n) = \mathbf{I} - \mathbf{P}_{\mathbf{U}}(n) \tag{58}$$

The columns of $\mathbf{U}(n)$ are formed by the vectors that span the vector space associated with $\mathbf{P}_{\mathbf{U}}(n)$; hence when we compute $\mathbf{P}_{\mathbf{U}}(n)\mathbf{U}(n)$, we have

$$\mathbf{P}_{\mathbf{U}}(n)\mathbf{U}(n) = \mathbf{U}(n)\underbrace{\left(\mathbf{U}^T(n)\mathbf{U}(n)\right)^{-1}\mathbf{U}^T(n)\mathbf{U}(n)}_{\mathbf{I}} = \mathbf{U}(n) \tag{59}$$

It follows from this that the projection matrix is *idempotent*

$$\mathbf{P}_{\mathbf{U}}(n)\mathbf{P}_{\mathbf{U}}(n) = \underbrace{\mathbf{P}_{\mathbf{U}}(n)\mathbf{U}(n)}_{\mathbf{U}(n)}\left(\mathbf{U}^T(n)\mathbf{U}(n)\right)^{-1}\mathbf{U}^T(n) = \mathbf{P}_{\mathbf{U}}(n) \tag{60}$$

The following relations also follow from the definition of $\mathbf{P}_{\mathbf{U}}(n)$ and $\mathbf{P}_{\mathbf{U}}^{\perp}(n)$.

$$\mathbf{P}_{\mathbf{U}}^{T}(n) = \mathbf{P}_{\mathbf{U}}(n) \tag{61}$$

$$\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{P}_{\mathbf{U}}^{\perp}(n) = \mathbf{P}_{\mathbf{U}}^{\perp}(n) \tag{62}$$

$$\mathbf{P}_{\mathbf{U}}^{\perp^T}(n) = \mathbf{P}_{\mathbf{U}}^{\perp}(n) \tag{63}$$

$$\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{P}_{\mathbf{U}}(n) = \mathbf{O}_{(n+1)\times(n+1)} \tag{64}$$

All of these relations are easy to prove.

Now let us append a matrix $\mathbf{V}$ (i.e., more columns) to the matrix $\mathbf{U}(n)$. The columns of $\mathbf{V}$ do not need to be orthogonal to the subspace spanned by $\mathbf{U}(n)$. hence the subspace spanned by $[\mathbf{U}(n), \mathbf{V}]$ is the same as the one spanned by $[\mathbf{U}(n), \mathbf{W}]$, with

$$\mathbf{W} = \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V} \tag{65}$$

Then we define

$$\begin{aligned} \mathbf{P}_{\mathbf{UV}}(n) &= \mathbf{P}_{\mathbf{U}}(n) + \mathbf{P}_{\mathbf{W}}(n) \\ &= \mathbf{P}_{\mathbf{U}}(n) + \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n) \end{aligned} \tag{66}$$

If we note that $\mathbf{P}_{\mathbf{UV}}^{\perp}(n) = \mathbf{I} - \mathbf{P}_{\mathbf{UV}}(n)$; then it follows that

$$\mathbf{P}_{\mathbf{UV}}^{\perp}(n) = \mathbf{P}_{\mathbf{U}}^{\perp}(n) - \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n) \tag{67}$$

These results also are valid when $\mathbf{V}$ is a vector (i.e., a matrix with a single column).

Some other useful relations with generalized vectors or matrices $\mathbf{y}$ and $\mathbf{z}$ which follow immediately from (66) and (67), are

$$\mathbf{P}_{\mathbf{UV}}(n)\mathbf{y} = \mathbf{P}_{\mathbf{U}}(n)\mathbf{y} + \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} \tag{68}$$

$$\mathbf{P}_{\mathbf{UV}}^{\perp}(n)\mathbf{y} = \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} - \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} \tag{69}$$

$$\begin{aligned} \mathbf{z}^T\mathbf{P}_{\mathbf{UV}}(n)\mathbf{y} &= \mathbf{z}^T\mathbf{P}_{\mathbf{U}}(n)\mathbf{y} \\ &+ \mathbf{z}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} \end{aligned} \tag{70}$$

$$\begin{aligned} \mathbf{z}^T\mathbf{P}_{\mathbf{UV}}^{\perp}(n)\mathbf{y} &= \mathbf{z}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} \\ &- \mathbf{z}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{U}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{y} \end{aligned} \tag{71}$$

These relations with the appropriate choices of $\mathbf{U}(n), \mathbf{V}, \mathbf{y}, \mathbf{z}$ will be the basis of several recursions needed later.

1. Projection Matrix Time Update

Let us partition the data matrix $\mathbf{U}(n)$ as

$$\mathbf{U}(n) = \begin{bmatrix} \mathbf{U}(n-1) \\ \underline{\mathbf{u}}^T \end{bmatrix} \tag{72}$$

where $\underline{\mathbf{u}}^T$ is the last row of $\mathbf{U}(n)$. We can see from (47) that this is a valid representation when $\mathbf{U}(n)$ is taken to be $\mathbf{Y}_{1,N}(n)$. In this case $\underline{\mathbf{u}}^T$ corresponds to $\underline{\mathbf{y}}_{1,N}^T(n)$. It is seen that $\mathbf{U}(n)$ is formed by the columns of $\mathbf{U}(n-1)$ with one more dimension appended, the components of $\underline{\mathbf{u}}^T$. Now define a $(n+1)$-dimensional vector $\underline{\pi}(n)$ called the *unit time vector* [Ref. 6]

$$\underline{\pi}(n) = [0, 0, 0, \cdots, 0, 0, 0, 1]^T \tag{73}$$

and append it to the data matrix $\mathbf{U}(n)$ to form $[\mathbf{U}(n), \underline{\pi}(n)]$. It will now be shown that the subspace spanned by this new matrix contains not only $\underline{\hat{\mathbf{y}}}_1(n)$ but also $\underline{\hat{\mathbf{y}}}_1(n-1)$.

To see this, proceed as follows. We know that if $\underline{\hat{\mathbf{y}}}_1(n)$ lies in the subspace spanned by $\mathbf{U}(n)$, then appending $\underline{\pi}(n)$ will not change a thing. Using (68) with $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{y} = \underline{\hat{\mathbf{y}}}_1(n)$ we obtain

$$\begin{aligned} \mathbf{P}_{\mathbf{U}\underline{\pi}}(n)\underline{\hat{\mathbf{y}}}_1(n) &= \underbrace{\mathbf{P}_{\mathbf{U}}(n)\underline{\hat{\mathbf{y}}}_1(n)}_{\underline{\hat{\mathbf{y}}}_1(n)} \\ &+ \mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n)\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n)]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n)\right)^{-1}\underline{\pi}^T(n)\underbrace{\mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\hat{\mathbf{y}}}_1(n)}_{\underline{\mathbf{0}}} \end{aligned} \tag{74}$$

To see that $\underline{\hat{\mathbf{y}}}_1(n-1)$ also lies in this subspace, note that since $\underline{\pi}(n)$ has only its last component non-zero, a linear combination of the vectors in $[\mathbf{U}(n), \underline{\pi}(n)]$ can be used to obtain a matrix whose columns span the same subspace as the columns of $\mathbf{U}(n-1)$.

It can be shown that $\mathbf{P}_{\mathbf{U}\underline{\pi}}(n)$ has the particular form

$$\mathbf{P}_{\mathbf{U}\underline{\pi}}(n) = \begin{bmatrix} \mathbf{P}_{\mathbf{U}}(n-1) & \underline{\mathbf{0}} \\ \underline{\mathbf{0}}^T & 1 \end{bmatrix} \tag{75}$$

(a detailed derivation of this result is presented in the Appendix). Further recall that $\underline{y}_1(n)$ is a vector formed by all the data from the origin up to point n. Thus (75) provides a way to decompose a projection matrix into *past* and *present* components.

$$\begin{aligned} \mathbf{P}_{\mathbf{U}\underline{\pi}}(n)\underline{y}_1(n) &= \begin{bmatrix} \mathbf{P}_{\mathbf{U}}(n-1) & \underline{0} \\ \underline{0}^T & 0 \end{bmatrix} \begin{bmatrix} \underline{y}_1(n-1) \\ y_1(n) \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{O} & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} \underline{y}_1(n-1) \\ y_1(n) \end{bmatrix} = \begin{bmatrix} \underline{\hat{y}}_1(n-1) \\ y_1(n) \end{bmatrix} \end{aligned} \tag{76}$$

2. Angle Parameter

The vector $\underline{\pi}(n)$ also provides a way to quantify the change of subspaces when we update $\mathbf{U}(n-1)$ to $\mathbf{U}(n)$. First note that the inner product of two vectors gives the cosine of the angle between them multiplied by the product of their lengths, and also that the length of a vector resulting from the projection of any vector into a subspace is given by the length of the original vector multiplied by the cosine of the angle between the vector and the subspace (this angle has always magnitude $\leq \frac{\pi}{2}$). Now observe that $\underline{\pi}(n)$ is a unit vector orthogonal to the subspace spanned by $\mathbf{U}(n-1)$, and $\mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n)$ is a vector orthogonal to the subspace spanned by $\mathbf{U}(n)$ with length equal to the cosine of the angle between $\underline{\pi}(n)$ and this subspace. Then defining the inner-product as

$$\langle \underline{a}, \underline{b} \rangle = \underline{a}^T \underline{b} \tag{77}$$

we obtain

$$\gamma(n) = \left\langle \underline{\pi}(n), \mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n) \right\rangle = 1 \times \cos\theta \times \cos\theta = \cos^2\theta \tag{78}$$

where θ is the angle between the components of $\underline{\pi}(n)$ that are perpendicular to both subspaces $\{\mathbf{U}(n)\}$ and $\{\mathbf{U}(n-1)\}$. The variable $\gamma(n)$ will be used to update $\mathbf{K}_{1,N}(n)$.

In order to begin the derivation of the recursive procedure we now introduce three auxiliary filters:

- Forward Multichannel Prediction Filter
- Backward Multichannel Prediction Filter
- Gain Transversal Filter

These are discussed next.

F. AUXILIARY FILTERS

The reason for the auxiliary filters will become clear later when it is shown that the update of $\underline{\mathbf{a}}(n-1)$ can be expressed in terms of the auxiliary filters parameters.

1. (M+1) Channel Forward Prediction Filter

We begin by defining a (M+1)-channel signal $\mathbf{x}_F(n)$ formed by the data acquired by the 2-D mask when it is moved from n to $n+1$:

$$\mathbf{x}_F(n) = [y_1(n), y_2(n+1), \cdots, y_{M+1}(n+1)]^T \tag{79}$$

We want to find the *best LS* filter that predicts $\mathbf{x}_F(n)$ based on the data $\underline{\mathbf{y}}_{1,N}(n)$ covered by the 2-D mask (see Figure 5). Let the coefficients of this filter be defined by a $((N+1)(M+1)-1) \times (M+1)$ matrix of the form

$$\underline{\mathbf{F}}(n) = [\underline{\mathbf{f}}_1(n), \underline{\mathbf{f}}_2(n), \cdots, \underline{\mathbf{f}}_{M+1}(n)] \tag{80}$$

where each of the $((N+1)(M+1)-1)$-dimensional vectors $\underline{\mathbf{f}}_i(n)$ for $(1 \leq i \leq M+1)$ is comprised of the multichannel prediction coefficients for channel i with the same support as $\underline{\mathbf{a}}(n)$. The prediction of $\mathbf{x}_F(n)$ is given by

$$\mathbf{x}_F(n) = \mathbf{F}^T(n)\underline{\mathbf{y}}_{1,N}(n) \tag{81}$$

and the prediction error is

$$\mathbf{e}_F(n) = \mathbf{x}_F(n) - \hat{\mathbf{x}}_F(n) \tag{82}$$

If we define

$$\begin{aligned} \mathbf{X}_F(n) &= \left[\underline{\mathbf{y}}_1(n), \underline{\mathbf{y}}_2(n+1), \cdots, \underline{\mathbf{y}}_{M+1}(n+1)\right] \\ &= [\mathbf{x}_F(0), \mathbf{x}_F(1), \cdots, \mathbf{x}_F(n)]^T \end{aligned} \tag{83}$$

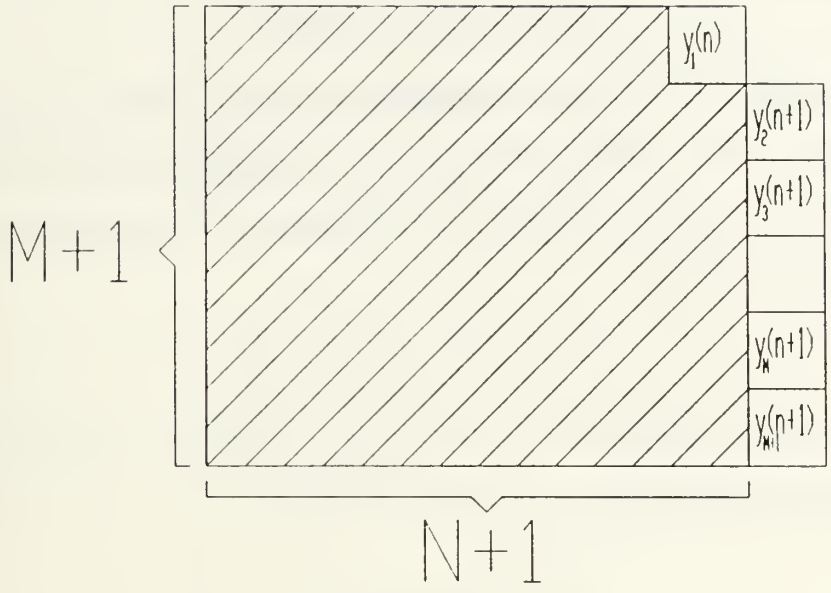

Figure 5. Forward Filter Mask

then $\mathbf{X}_F(n)$ is a $(n+1) \times (M+1)$ matrix that contains all the multichannel data from the origin up to current value of the index n. The estimate of $\mathbf{X}_F(n)$ is thus

$$\hat{\mathbf{X}}_F(n) = \mathbf{Y}_{1,N}(n)\mathbf{F}(n) \tag{84}$$

and the prediction error, also a $(n+1) \times (M+1)$ matrix, is

$$\mathbf{E}_F(n) = \mathbf{X}_F(n) - \hat{\mathbf{X}}_F(n) \tag{85}$$

The error covariance for the multichannel Forward Filter is:

$$\Sigma_F(n) = \mathbf{E}_F^T(n)\mathbf{E}_F(n) \tag{86}$$

Since we desire $\mathbf{F}(n)$ to minimize the error energy

$$tr\left[\Sigma_F(n)\right] = tr\left[\mathbf{E}_F^T(n)\mathbf{E}_F(n)\right] \tag{87}$$

the *optimal LS* filter is again obtained using the pseudo-inverse of $\mathbf{Y}_{1,N}(n)$

$$\mathbf{F}(n) \;=\; \left(\mathbf{Y}_{1,N}^T(n)\mathbf{Y}_{1,N}(n)\right)^{-1}\mathbf{Y}_{1,N}^T(n)\mathbf{X}_F(n) \tag{88}$$

This can be expressed using the operator defined in (55) as

$$\mathbf{F}(n) = \mathbf{K}_{1,N}(n)\mathbf{X}_F(n) \tag{89}$$

The orthogonality principles mentioned before also apply. That is, the estimate $\hat{\mathbf{X}}_F(n)$ is the projection of the columns of $\mathbf{X}_F(n)$ onto the subspace spanned by the columns of the matrix containing the *previous* multichannel data (which in this case is the same as the 2-D data).

$$\hat{\mathbf{X}}_F(n) = \mathbf{P}_{1,N}(n)\mathbf{X}_F(n) \tag{90}$$

The error $\mathbf{E}_F(n)$ is orthogonal to the estimate $\hat{\mathbf{E}}_F(n)$, i.e., the columns of these matrices span subspaces that are orthogonal to each other.

$$\mathbf{E}_F(n) = \mathbf{P}^{\perp}_{1,N}(n)\mathbf{X}_F(n) \tag{91}$$

If $\mathbf{e}_F^T(n)$ is defined as the last row of $\mathbf{E}_F(n)$, it can be obtained using $\underline{\pi}(n)$ as

$$\mathbf{e}_F^T(n) = \underline{\pi}^T(n)\mathbf{E}_F(n) = \underline{\pi}(n)^T\mathbf{P}^{\perp}_{1,N}(n)\mathbf{X}_F(n) \tag{92}$$

2. (M+1) Channel Backward Prediction Filter

For the backward prediction problem we define a (M+1)-channel signal $\mathbf{x}_B(n)$ formed by the data left out by the 2-D mask when it is moved from time n to $n+1$. This is given by (see Figure 6)

$$\mathbf{x}_B(n) = [y_1(n-N), y_2(n-N), \cdots, y_{M+1}(n-N)]^T \tag{93}$$

Now define a $(n+1)\times((N+1)(M+1)-1)$ data matrix $\mathbf{Y}_{0,N-1}(n)$ that has a structure similar to $\mathbf{Y}_{1,N}(n)$ (47,48)

$$\mathbf{Y}_{0,N-1}(n) = \begin{bmatrix} \underline{\mathbf{y}}^T_{0,N-1}(0) \\ \underline{\mathbf{y}}^T_{0,N-1}(1) \\ \vdots \\ \vdots \\ \underline{\mathbf{y}}^T_{0,N-1}(n) \end{bmatrix} \tag{94}$$

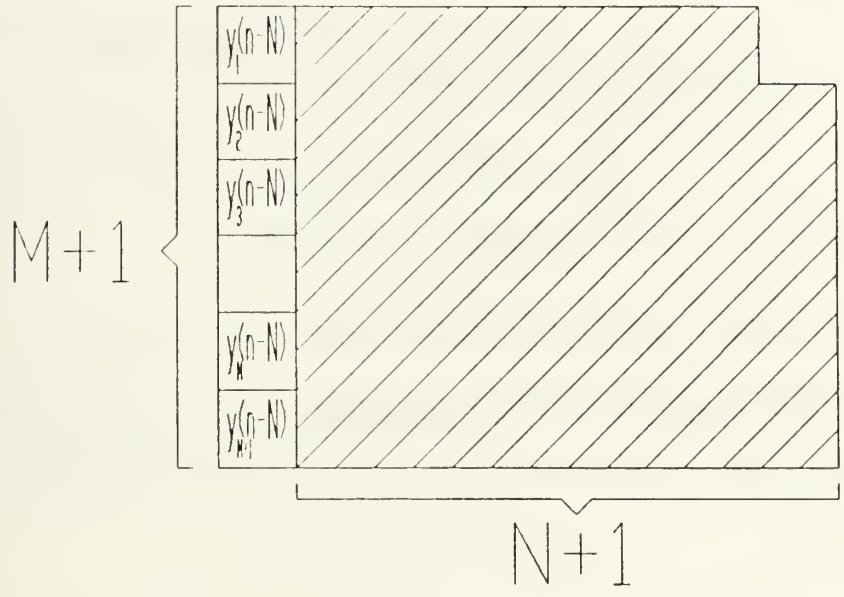

Figure 6. Backward Filter Mask

or

$$\mathbf{Y}_{0,N-1}(n) = [\underline{y}_1(n), \cdots, z^{-N+1}\underline{y}_1(n), z\underline{y}_2(n), \underline{y}_2(n), \cdots, z^{-N+1}\underline{y}_{M+1}(n)] \quad (95)$$

We note that

$$\underline{y}_{1,N}(n) = \underline{y}_{0,N-1}(n-1) \quad (96)$$

and

$$\mathbf{Y}_{1,N}(n) = z^{-1}\mathbf{Y}_{0,N-1}(n) \quad (97)$$

The problem is now to find the *best LS* filter that predicts $\mathbf{x}_B(n)$ based on the data matrix $\mathbf{Y}_{0,N-1}(n)$. Let the coefficients of this filter be defined by the $((N+1)(M+1)-1) \times (M+1)$ matrix

$$\underline{\mathbf{B}}(n) = [\underline{b}_1(n), \underline{b}_2(n), \cdots, \underline{b}_{M+1}(n)] \quad (98)$$

where each of the $((N+1)(M+1)-1)$-dimensional vectors $\underline{b}_i(n)$ for $(1 \leq i \leq M+1)$ is comprised of the backward prediction coefficients for channel i. The support of

this filter is the same as the support of $\underline{\mathbf{a}}(n)$ but shifted one sample to the right (see Figure 6).

The prediction of $\mathbf{x}_B(n)$ is given by

$$\mathbf{x}_B(n) = \mathbf{B}(n)^T \underline{\mathbf{y}}_{0,N-1}(n) \tag{99}$$

and the prediction error is

$$\mathbf{e}_B(n) = \mathbf{x}_B(n) - \hat{\mathbf{x}}_B(n) \tag{100}$$

We continue to proceed as we did for the forward multichannel filter. The $(n+1) \times (M+1)$ matrix that contains the backward multichannel data from the origin up to n is defined as

$$\begin{aligned} \mathbf{X}_B(n) &= \left[z^{-N}\underline{\mathbf{y}}_1(n), z^{-N}\underline{\mathbf{y}}_2(n), \cdots, z^{-N}\underline{\mathbf{y}}_{M+1}(n)\right] \\ &= [\mathbf{x}_B(0), \mathbf{x}_B(1), \cdots, \mathbf{x}_B(n)]^T \end{aligned} \tag{101}$$

Then the estimate of $\mathbf{X}_B(n)$ is

$$\hat{\mathbf{X}}_B(n) = \mathbf{Y}_{0,N-1}(n)\mathbf{B}(n) \tag{102}$$

and the prediction error (also a $(n+1) \times (M+1)$ matrix) is

$$\mathbf{E}_B(n) = \mathbf{X}_B(n) - \hat{\mathbf{X}}_B(n) \tag{103}$$

The error covariance matrix for the multichannel backward filter is then given by

$$\Sigma_B(n) = \mathbf{E}_B^T(n)\mathbf{E}_B(n) = \mathbf{X}_B(n)^T \mathbf{P}_{0,N-1}^{\perp}(n)\mathbf{X}_B(n) \tag{104}$$

To minimize the error energy

$$tr\left[\Sigma_B(n)\right] = tr\left[\mathbf{E}_B(n)^T\mathbf{E}_B(n)\right] \tag{105}$$

our *optimal LS* filter is, once more, obtained using a pseudo-inverse, but this time for the data matrix $\mathbf{Y}_{0,N-1}(n)$.

$$\mathbf{B}(n) = \left(\mathbf{Y}_{0,N-1}^T(n)\mathbf{Y}_{0,N-1}(n)\right)^{-1} \mathbf{Y}_{0,N-1}^T(n)\mathbf{X}_B(n) \tag{106}$$

Finally defining a new operator $\mathbf{K}_{0,N-1}(n)$ in terms of the new data matrix $\mathbf{Y}_{0,N-1}(n)$

$$\mathbf{K}_{0,N-1}(n) = \left(\mathbf{Y}_{0,N-1}^T(n)\mathbf{Y}_{0,N-1}(n)\right)^{-1}\mathbf{Y}_{0,N-1}^T(n) \tag{107}$$

we can rewrite (106) as

$$\mathbf{B}(n) = \mathbf{K}_{0,N-1}(n)\mathbf{X}_B(n) \tag{108}$$

The orthogonality principles apply once more. The estimate $\hat{\mathbf{X}}_B(n)$ is the projection of the columns of $\mathbf{X}_B(n)$ onto the subspace spanned by the *previous* backward multichannel data

$$\hat{\mathbf{X}}_B(n) = \mathbf{P}_{0,N-1}(n)\mathbf{X}_B(n) \tag{109}$$

where $\mathbf{P}_{0,N-1}(n)$ is the projection matrix associated with the vector space spanned by the columns of the new data set $\mathbf{Y}_{0,N-1}(n)$.

$$\mathbf{P}_{0,N-1}(n) = \mathbf{Y}_{0,N-1}(n)\left(\mathbf{Y}_{0,N-1}^T(n)\mathbf{Y}_{0,N-1}(n)\right)^{-1}\mathbf{Y}_{0,N-1}^T(n) \tag{110}$$

The error $\mathbf{E}_B(n)$ is orthogonal to the estimate $\hat{\mathbf{X}}_B(n)$, i.e., the columns of these two matrices span subspaces that are orthogonal to each other.

$$\mathbf{E}_B(n) = \mathbf{P}_{0,N-1}^{\perp}(n)\mathbf{X}_B(n) \tag{111}$$

Defining $\mathbf{e}_B^T(n)$ to be the last row of $\mathbf{E}_B(n)$ we have

$$\mathbf{e}_B^T(n) = \underline{\pi}(n)^T\mathbf{E}_B(n) = \underline{\pi}(n)^T\mathbf{P}_{0,N-1}^{\perp}(n)\mathbf{X}_B(n) \tag{112}$$

3. Gain Transversal Filter

The gain transversal filter does not relate to specific prediction operations for the data but rather provides another way of quantifying the angular change $\gamma(n)$ between the subspaces associated with data matrices at times n and $n-1$. To begin, consider the projection of the vector $\underline{\pi}(n)$ onto the subspace spanned by $\mathbf{Y}_{0,N-1}(n)$.

Since this projection $\mathbf{P}_{0,N-1}(n)\underline{\pi}(n)$ is contained in the subspace of $\mathbf{Y}_{0,N-1}(n)$, we can express it as a linear combination of the columns of $\mathbf{Y}_{0,N-1}(n)$. We write this as

$$\mathbf{P}_{0,N-1}(n)\underline{\pi}(n) = \mathbf{Y}_{0,N-1}(n)\underline{\mathbf{g}}(n) \tag{113}$$

where $\underline{\mathbf{g}}(n)$ is a $((N+1)(M+1)-1)$-dimensional vector of weights. Note that (113) can be interpreted as the LS prediction of $\underline{\pi}(n)$ based on the data matrix $\mathbf{Y}_{0,N-1}(n)$ where $\underline{\mathbf{g}}(n)$ is the $((N+1)(M+1)-1)$-dimensional vector of filter prediction coefficients. The estimate of $\underline{\pi}(n)$ is thus given by

$$\underline{\hat{\pi}}(n) = \mathbf{P}_{0,N-1}(n)\underline{\pi}(n) = \mathbf{Y}_{0,N-1}(n)\underline{\mathbf{g}}(n) \tag{114}$$

and the prediction error is

$$\underline{\mathbf{e}}_{\underline{\pi}}(n) = \underline{\pi}(n) - \mathbf{P}_{0,N-1}(n)\underline{\pi}(n) = \mathbf{P}^{\perp}_{0,N-1}(n)\underline{\pi}(n) \tag{115}$$

The last component of $\underline{\mathbf{e}}_{\underline{\pi}}(n)$ can be obtained using $\underline{\pi}(n)$ and turns out to be equal to $\gamma(n)$ as in (78)

$$\epsilon_{\underline{\pi}}(n) = \underline{\pi}(n)^T\mathbf{P}^{\perp}_{0,N-1}(n)\underline{\pi}(n) = \gamma(n) = \cos^2\theta \tag{116}$$

Then substituting the middle part of (115) in (116) we obtain

$$\gamma(n) = \langle\underline{\pi}(n), \underline{\pi}(n) - \mathbf{P}_{0,N-1}(n)\underline{\pi}(n)\rangle = 1 - \langle\underline{\pi}(n), \mathbf{P}_{0,N-1}(n)\underline{\pi}(n)\rangle \tag{117}$$

and using (113) in (117) we find

$$\gamma(n) = 1 - \left\langle\underline{\pi}(n), \mathbf{Y}_{0,N-1}(n)\underline{\mathbf{g}}(n)\right\rangle = 1 - \underline{\mathbf{y}}^T_{0,N-1}(n)\underline{\mathbf{g}}(n) = \cos^2\theta \tag{118}$$

where $\underline{\mathbf{y}}^T_{0,N-1}(n)$ is the last row of the data matrix $\mathbf{Y}_{0,N-1}(n)$. If we now recognize that $\cos^2\theta = 1 - \sin^2\theta$ we see that

$$\underline{\mathbf{y}}^T_{0,N-1}(n)\underline{\mathbf{g}}(n) = \sin^2\theta \tag{119}$$

If we use the LS criteria to get $\underline{\mathbf{g}}(n)$, the solution is again, given in terms of a pseudo-inverse, or more conveniently in terms of the operator $\mathbf{K}_{0,N-1}(n)$ of (107)

$$\underline{\mathbf{g}}(n) = \mathbf{K}_{0,N-1}(n)\underline{\pi}(n) \tag{120}$$

This is in the same form that we have for the other filters.

G. FILTER UPDATE PROCEDURES

In this section we determine the time update for the four filters defined so far. We begin by showing how to update $\mathbf{K}_{1,N}(n)$ and $\mathbf{K}_{0,N-1}(n)$. This result will be used for updating each of the four filters.

1. Transversal Operator Update

Since the operations to be described now are common to all four filters, we develop the formulas in this subsection in terms of the generic matrices $\mathbf{U}$ and $\mathbf{V}$ introduced in section E of this chapter. Beginning with (66) we have

$$\mathbf{P}_{\mathbf{UV}}(n) = \mathbf{P}_{\mathbf{U}}(n) + \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{P}_{\mathbf{U}}^{\perp}(n) \tag{121}$$

where $\mathbf{U}$ is a $(n+1)\times((N+1)(M+1)-1)$ matrix and $\mathbf{V}$ is a $(n+1)\times(M+1)$ matrix. By definition

$$\mathbf{P}_{\mathbf{UV}}(n) = [\mathbf{U},\mathbf{V}]\left([\mathbf{U},\mathbf{V}]^T[\mathbf{U},\mathbf{V}]\right)^{-1}[\mathbf{U},\mathbf{V}]^T \tag{122}$$

and

$$\mathbf{K}_{\mathbf{UV}}(n) = \left([\mathbf{U},\mathbf{V}]^T[\mathbf{U},\mathbf{V}]\right)^{-1}[\mathbf{U},\mathbf{V}]^T \tag{123}$$

It follows that

$$\mathbf{K}_{\mathbf{UV}}(n) = \mathbf{K}_{\mathbf{UV}}(n)\mathbf{P}_{\mathbf{UV}}(n) \tag{124}$$

hence from (123)

$$\begin{aligned}\mathbf{K}_{\mathbf{UV}}(n)[\mathbf{U},\mathbf{V}] &= \mathbf{I}_{[(N+1)(M+1)+M]\times[(N+1)(M+1)+M]} \\ &= \begin{bmatrix} \mathbf{I}_{N'\times N'} & \mathbf{O}_{N'\times M'} \\ \mathbf{O}_{M'\times N'} & \mathbf{I}_{M'\times M'} \end{bmatrix}\end{aligned} \tag{125}$$

with $N' = ((N+1)(M+1)-1)$ and $M' = (M+1)$. Now we can partition (125) and write it as two equations, namely

$$\mathbf{K}_{\mathbf{UV}}(n)\mathbf{U} = \begin{bmatrix} \mathbf{I}_{N'\times N'} \\ \mathbf{O}_{M'\times N'} \end{bmatrix} \tag{126}$$

and

$$\mathbf{K}_{\mathbf{UV}}(n)\mathbf{V} = \begin{bmatrix} \mathbf{O}_{N'\times M'} \\ \mathbf{I}_{M'\times M'} \end{bmatrix} \tag{127}$$

The first equation can be used in conjunction with $\mathbf{P}_{\mathbf{U}} = \mathbf{U}(n)\mathbf{K}_{\mathbf{U}}(n)$ to obtain

$$\mathbf{K}_{\mathbf{UV}}(n)\mathbf{P}_{\mathbf{U}}(n) = \begin{bmatrix} \mathbf{I}_{N'\times N'} \\ \mathbf{O}_{M'\times N'} \end{bmatrix} \mathbf{K}_{\mathbf{U}}(n) = \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n) \\ \mathbf{O}_{M'\times(n+1)} \end{bmatrix} \tag{128}$$

Now substituting (121) in (124) and using (125)-(128) to simplify the resulting expression we have

$$\begin{aligned} \mathbf{K}_{\mathbf{UV}}(n) &= \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n) \\ \mathbf{O}_{M'\times(n+1)} \end{bmatrix} + \left\{ \begin{bmatrix} \mathbf{O}_{N'\times M'} \\ \mathbf{I}_{M'\times M'} \end{bmatrix} - \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n) \\ \mathbf{O}_{M'\times(n+1)} \end{bmatrix} \mathbf{V} \right\} \\ &\times \left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V} \right)^{-1} \mathbf{V}^T \mathbf{P}_{\mathbf{U}}^{\perp}(n) \end{aligned} \tag{129}$$

By forming $[\mathbf{V}, \mathbf{U}]$ and following similar procedures, it is straightforward to show that

$$\begin{aligned} \mathbf{K}_{\mathbf{VU}}(n) &= \begin{bmatrix} \mathbf{O}_{M'\times(n+1)} \\ \mathbf{K}_{\mathbf{U}}(n) \end{bmatrix} + \left\{ \begin{bmatrix} \mathbf{I}_{M'\times M'} \\ \mathbf{O}_{N'\times M'} \end{bmatrix} - \begin{bmatrix} \mathbf{O}_{M'\times(n+1)} \\ \mathbf{K}_{\mathbf{U}}(n) \end{bmatrix} \mathbf{V} \right\} \\ &\times \left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V}]^T \mathbf{P}_{\mathbf{U}}^{\perp}(n)\mathbf{V} \right)^{-1} \mathbf{V}^T \mathbf{P}_{\mathbf{U}}^{\perp}(n) \end{aligned} \tag{130}$$

For the particular case when $\mathbf{V} = \underline{\pi}(n)$ these relations are also valid but we can obtain them from the derivation given in the Appendix. The result is

$$\mathbf{K}_{\mathbf{U}\underline{\pi}}(n) = \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n-1) & \underline{0} \\ -\underline{\mathbf{u}}^T \mathbf{K}_{\mathbf{U}}(n-1) & 1 \end{bmatrix} \tag{131}$$

To check this, note that for $\mathbf{V} = \underline{\pi}(n)$ (124) can be written as

$$\mathbf{K}_{\mathbf{U}\underline{\pi}}(n) = \mathbf{K}_{\mathbf{U}\underline{\pi}}(n)\mathbf{P}_{\mathbf{U}\underline{\pi}}(n) \tag{132}$$

Then equating these two ways of obtaining $\mathbf{K}_{\mathbf{U},\underline{\pi}}(n)$ we can update $\mathbf{K}_{\mathbf{U}}(n-1)$ to $\mathbf{K}_{\mathbf{U}}(n)$

$$\begin{aligned} \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n-1) & \underline{0} \\ -\underline{\mathbf{u}}^T \mathbf{K}_{\mathbf{U}}(n-1) & 1 \end{bmatrix} &= \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n) \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n)\underline{\pi}(n) \\ -1 \end{bmatrix} \\ &\times \left([\mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n)]^T \mathbf{P}_{\mathbf{U}}^{\perp}(n)\underline{\pi}(n) \right)^{-1} \underline{\pi}^T(n)\mathbf{P}_{\mathbf{U}}^{\perp}(n) \end{aligned} \tag{133}$$

2. 2-D Filter Update

We now develop update formulas for the 2-D filter. From (56) we have

$$\underline{\mathbf{a}}(n) = \mathbf{K}_{1,N}(n)\underline{\mathbf{y}}_1(n) \tag{134}$$

To find $\underline{\mathbf{a}}(n)$ in terms of $\underline{\mathbf{a}}(n-1)$ we use (134). Post-multiplying by $\underline{\mathbf{y}}_1(n)$, and taking $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{U} = \mathbf{Y}_{1,N}(n)$ yields after simplification

$$\begin{bmatrix} \mathbf{K}_{1,N}(n-1) & \underline{0} \\ -\underline{\mathbf{y}}_{1,N}^T\mathbf{K}_{1,N}(n-1) & 1 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{y}}_1(n-1) \\ y_1(n) \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{1,N}(n) \\ 0 \end{bmatrix} \underline{\mathbf{y}}_1(n) - \begin{bmatrix} \mathbf{K}_{1,N}(n)\underline{\pi}(n) \\ -1 \end{bmatrix} \frac{\underline{\pi}^T(n)\mathbf{P}_{1,N}^{\perp}(n)\underline{\mathbf{y}}_1(n)}{\underline{\pi}^T(n)\mathbf{P}_{1,N}^{\perp}(n)\pi(n)} \tag{135}$$

At this point we note from (97) that since

$$\mathbf{Y}_{1,N}(n) = z^{-1}\mathbf{Y}_{0,N-1}(n) \tag{136}$$

with $\underline{\mathbf{y}}_{1,N}(0) = \underline{0}$ as defined initially, then

$$\mathbf{Y}_{1,N}(n) = \begin{bmatrix} \underline{0}^T \\ \mathbf{Y}_{0,N-1}(n-1) \end{bmatrix} \tag{137}$$

From this it follows, using the respective definitions, that

$$\mathbf{K}_{1,N}(n) = [\underline{0}, \mathbf{K}_{0,N-1}(n-1)] \tag{138}$$

and

$$\mathbf{P}_{1,N}(n) = \begin{bmatrix} 0 & \underline{0}^T \\ \underline{0} & \mathbf{P}_{0,N-1}(n-1) \end{bmatrix} \tag{139}$$

These results in conjunction with (118) and (120) allow us to write

$$\underline{\mathbf{g}}(n-1) = \mathbf{K}_{0,N-1}(n-1)\underline{\pi}(n-1) = \mathbf{K}_{1,N}(n)\underline{\pi}(n) \tag{140}$$

and

$$\begin{aligned}\gamma(n-1) &= 1-\left\langle \underline{\pi}(n-1), \mathbf{Y}_{0,N-1}(n-1)\underline{\mathbf{g}}(n-1)\right\rangle = 1-\underline{\mathbf{y}}^T_{0,N-1}(n-1)\underline{\mathbf{g}}(n-1)\\ &= 1-\left\langle \underline{\pi}(n), \mathbf{Y}_{1,N}(n)\underline{\mathbf{g}}(n-1)\right\rangle = 1-\underline{\mathbf{y}}^T_{1,N}(n)\underline{\mathbf{g}}(n-1) \qquad (141)\end{aligned}$$

and also

$$\gamma(n-1) = \underline{\pi}^T(n)\mathbf{P}^{\perp}_{1,N}(n)\underline{\pi}(n) \qquad (142)$$

The upper part of (135) then becomes

$$\underline{\mathbf{a}}(n-1) = \underline{\mathbf{a}}(n) - \underline{\mathbf{g}}(n-1)\frac{e_1(n)}{\gamma(n-1)} \qquad (143)$$

or

$$\underline{\mathbf{a}}(n) = \underline{\mathbf{a}}(n-1) + \underline{\mathbf{g}}(n-1)\frac{e_1(n)}{\gamma(n-1)} \qquad (144)$$

To get $\epsilon_1(n)$ before updating $\underline{\mathbf{a}}(n-1)$ we substitute (144) in

$$\epsilon_1(n) = y_1(n) - \underline{\mathbf{y}}^T_{1,N}(n)\underline{\mathbf{a}}(n) \qquad (145)$$

to obtain

$$\epsilon_1(n) = \epsilon_1(n|n-1) - \underline{\mathbf{y}}^T_{1,N}(n)\underline{\mathbf{g}}(n-1)\frac{e_1(n)}{\gamma(n-1)} \qquad (146)$$

where we define

$$\epsilon_1(n|n-1) = y_1(n) - \underline{\mathbf{y}}^T_{1,N}(n)\underline{\mathbf{a}}(n-1) \qquad (147)$$

Now (146) can be simplified using (141)

$$\epsilon_1(n) = \epsilon_1(n|n-1) + (\gamma(n-1)-1)\frac{e_1(n)}{\gamma(n-1)} = \epsilon_1(n|n-1)\gamma(n-1) \qquad (148)$$

and finally (144) can be written as

$$\underline{\mathbf{a}}(n) = \underline{\mathbf{a}}(n-1) + \underline{\mathbf{g}}(n-1)e_1(n|n-1) \qquad (149)$$

At this moment we have all we need to update $\underline{\mathbf{a}}(n-1)$ to $\underline{\mathbf{a}}(n)$ assuming that all variables with index $n-1$ are available. However we want to find $\mathbf{g}(n)$ and $\gamma(n)$ in order to proceed to update $\underline{\mathbf{a}}(n)$ to $\underline{\mathbf{a}}(n+1)$ as soon as $\mathbf{Y}_{1,N}(n+1)$ is available. This is discussed next.

3. Gain Filter Update

Begin by forming a new data matrix $\mathbf{Y}_{0,N}(n)$ that can be expressed in terms of $\mathbf{Y}_{1,N}(n)$ or $\mathbf{Y}_{0,N-1}(n)$ and the other matrices $\mathbf{X}_F(n)$ or $\mathbf{X}_B(n)$ after a multiplication by suitable permutation matrices. Each row and column of the permutation matrix contains a single 1 with all the other entries equal to zero. The positions of the 1's are chosen so that when this matrix premultiplies one of the data matrices it rearranges its columns to conform with the desired order. We write this as

$$\mathbf{Y}_{0,N}(n) = [\mathbf{X}_F(n), \mathbf{Y}_{1,N}(n)]\,\Psi_F = [\mathbf{Y}_{0,N-1}(n), \mathbf{X}_B(n)]\,\Psi_B \tag{150}$$

Now form $\mathbf{K}_{0,N}(n)$ and post-multiply the result by $\underline{\pi}(n)$ using the first part of (150) and (130) with $\mathbf{V} = \mathbf{X}_F(n)$ and $\mathbf{U} = \mathbf{Y}_{1,N}(n)$ to obtain

$$\begin{aligned}\mathbf{K}_{0,N}(n)\underline{\pi}(n) &= \Psi_F^T \begin{bmatrix} \mathbf{O}_{M'\times(n+1)} \\ \mathbf{K}_{1,N}(n) \end{bmatrix} \underline{\pi}(n) + \Psi_F^T \begin{bmatrix} \mathbf{I}_{M'\times M'} \\ \underbrace{-\mathbf{K}_{1,N}(n)\mathbf{X}_F(n)}_{-\mathbf{F}(n)} \end{bmatrix} \\ &\quad \times \left(\underbrace{\mathbf{X}_F^T(n)\mathbf{P}_{1,N}^{\perp}(n)\mathbf{X}_F(n)}_{\Sigma_F(n)} \right)^{-1} \underbrace{\mathbf{X}_F^T(n)\mathbf{P}_{1,N}^{\perp}(n)\underline{\pi}(n)}_{\mathbf{e}_F(n)}\end{aligned} \tag{151}$$

Since the permutation matrices are orthogonal ($\Psi^T = \Psi^{-1}$) we have

$$\Psi_F\mathbf{K}_{0,N}(n)\pi(n) = \begin{bmatrix} \mathbf{O}_{M'\times(n+1)} \\ \mathbf{K}_{1,N}(n) \end{bmatrix} \underline{\pi}(n) + \begin{bmatrix} \mathbf{I}_{M'\times M'} \\ -\mathbf{F}(n) \end{bmatrix} \Sigma_F^{-1}(n)\mathbf{e}_F(n) \tag{152}$$

By analogy with the definition of gain filter (120) we can define

$$\underline{\mathbf{g}}'(n) = \Psi_F\mathbf{K}_{0,N}(n)\underline{\pi}(n) \tag{153}$$

or

$$\Psi_F^T\underline{\mathbf{g}}'(n) = \mathbf{K}_{0,N}(n)\underline{\pi}(n) \tag{154}$$

as the $[(N+1)(M+1)-1]+(M+1)$-order LS predictor of $\underline{\pi}(n)$ using the permuted data matrix $\mathbf{Y}_{0,N}(n)\Psi_F^T$. Equation (152) can be rewritten using (140) as

$$\underline{\mathbf{g}}'(n) = \begin{bmatrix} \mathbf{O}_{M'} \\ \underline{\mathbf{g}}(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{I}_{M'\times M'} \\ -\mathbf{F}(n) \end{bmatrix} \Sigma_F^{-1}(n)\mathbf{e}_F(n) \tag{155}$$

An alternative way to find $\mathbf{K}_{0,N}(n)\underline{\pi}(n)$ is to use the second part of (150) in (129) with $\mathbf{V} = \mathbf{X}_B(n)$ and $\mathbf{U} = \mathbf{Y}_{0,N-1}(n)$ to obtain

$$\begin{aligned} \mathbf{K}_{0,N}(n)\underline{\pi}(n) &= \Psi_B^T \begin{bmatrix} \mathbf{K}_{0,N-1}(n) \\ \mathbf{O}_{M'\times(n+1)} \end{bmatrix} \underline{\pi}(n) \\ &+ \Psi_B^T \begin{bmatrix} -\mathbf{K}_{0,N-1}(n)\mathbf{X}_B(n) \\ \mathbf{I}_{M'\times M'} \end{bmatrix} {\Sigma_B}^{-1}(n)\mathbf{e}_B(n) \end{aligned} \tag{156}$$

Now similarly define

$$\underline{\mathbf{g}}''(n) = \Psi_B\mathbf{K}_{0,N}(n)\underline{\pi}(n) = \begin{bmatrix} \underline{\mathbf{g}}(n) \\ \mathbf{O}_{M'} \end{bmatrix} + \begin{bmatrix} -\mathbf{B}(n) \\ \mathbf{I}_{M'\times M'} \end{bmatrix} \Sigma_B^{-1}(n)\mathbf{e}_B(n) \tag{157}$$

as the $[(N+1)(M+1)-1]+(M+1)$-order LS predictor of $\underline{\pi}(n)$ using the permuted data matrix $\mathbf{Y}_{1,N-1}(n)\Psi_B^T$. Since

$$\Psi_B^T\underline{\mathbf{g}}''(n) = \Psi_F^T\underline{\mathbf{g}}'(n) \tag{158}$$

it follows that

$$\underline{\mathbf{g}}''(n) = \Psi_B\Psi_F^T\underline{\mathbf{g}}'(n) \tag{159}$$

It is useful to partition $\underline{\mathbf{g}}''(n)$ as

$$\underline{\mathbf{g}}''(n) = \begin{bmatrix} \mathbf{M}(n) \\ \mathbf{m}(n) \end{bmatrix} \tag{160}$$

where $\mathbf{M}(n)$ is a $(N+1)(M+1)-1$ vector and $\mathbf{m}(n)$ is a $(M+1)$ vector. The lower partition of (157) is then

$$\mathbf{m}(n) = \Sigma_B^{-1}(n)\mathbf{e}_B(n) \tag{161}$$

This can be substituted in the upper partition of (157) to give

$$\underline{\mathbf{g}}(n) = \mathbf{M}(n) + \mathbf{B}(n)\mathbf{m}(n) \tag{162}$$

Thus it is seen that if $\underline{\mathbf{g}}'(n)$ is available, we can obtain $\underline{\mathbf{g}}(n)$, provided that we have $\mathbf{F}(n)$, $\mathbf{B}(n)$, and all the associated parameters already available.

The inversions of $\Sigma_F(n)$ and $\Sigma_B(n)$ can also be carried out recursively using the matrix inversion lemma (details are given in subsection 7 of this chapter).

4. Forward Filter Update

To update the forward filter, proceed as follows. From (89) we have

$$\mathbf{F}(n) = \mathbf{K}_{1,N}(n)\mathbf{X}_F(n) \tag{163}$$

To find $\mathbf{F}(n)$ in terms of $\mathbf{F}(n-1)$ we use (134) post-multiplied by $\mathbf{X}_F(n)$ with $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{U}(n) = \mathbf{Y}_{1,N}(n)$ to obtain

$$\begin{aligned} &\begin{bmatrix} \mathbf{K}_{1,N}(n-1) & \underline{0} \\ -\underline{\mathbf{y}}_{1,N}^T\mathbf{K}_{1,N}(n-1) & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_F(n-1) \\ \mathbf{x}_F^T(n) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_{1,N}(n) \\ 0 \end{bmatrix} \mathbf{X}_F(n) - \begin{bmatrix} \mathbf{K}_{1,N}(n)\underline{\pi}(n) \\ -1 \end{bmatrix} \frac{\underline{\pi}^T(n)\mathbf{P}_{1,N}^{\perp}(n)\mathbf{X}_F(n)}{\underline{\pi}^T(n)\mathbf{P}_{1,N}^{\perp}(n)\underline{\pi}(n)} \end{aligned} \tag{164}$$

Then using only the upper partition we find

$$\mathbf{F}(n-1) = \mathbf{F}(n) - \underline{\mathbf{g}}(n-1)\frac{\mathbf{e}_F^T(n)}{\gamma(n-1)} \tag{165}$$

or

$$\mathbf{F}(n) = \mathbf{F}(n-1) + \underline{\mathbf{g}}(n-1)\frac{\mathbf{e}_F^T(n)}{\gamma(n-1)} \tag{166}$$

To compute $\mathbf{e}_F(n)$ before having $\mathbf{F}(n-1)$ we note from (81) and (82) that

$$\mathbf{e}_F(n) = \mathbf{x}_F(n) - \mathbf{F}^T(n)\underline{\mathbf{y}}_{1,N}(n) \tag{167}$$

and substitute (166) to obtain

$$\mathbf{e}_F^T(n) = \mathbf{e}_F^T(n|n-1) - \underline{\mathbf{y}}_{1,N}^T(n)\underline{\mathbf{g}}(n-1)\frac{\mathbf{e}_F^T(n)}{\gamma(n-1)} \tag{168}$$

where we define

$$\mathbf{e}_F(n|n-1) = \mathbf{x}_F(n) - \mathbf{F}^T(n-1)\underline{\mathbf{y}}_{1,N}(n) \tag{169}$$

Now (168) can be simplified to

$$\mathbf{e}_F^T(n) = \mathbf{e}_F^T(n|n-1) + (\gamma(n-1)-1)\frac{\mathbf{e}_F^T(n)}{\gamma(n-1)} = \mathbf{e}_F^T(n|n-1)\gamma(n-1) \tag{170}$$

Hence from (166) we have

$$\mathbf{F}(n) = \mathbf{F}(n-1) + \underline{\mathbf{g}}(n-1)\mathbf{e}_F^T(n|n-1) \tag{171}$$

This is the desired update formula for $\mathbf{F}(n)$.

To compute $\Sigma_F(n)$ we use (71) with $\mathbf{U} = \mathbf{Y}_{1,N}(n)$, $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{z} = \mathbf{y} = \mathbf{X}_F(n)$ and, also (75) to obtain

$$\mathbf{X}_F(n)^T\mathbf{P}^{\perp}_{1,N,\underline{\pi}}(n)\mathbf{X}_F(n) = \Sigma_F(n) - \frac{\mathbf{e}_F(n)\mathbf{e}_F^T(n)}{\gamma(n-1)} \tag{172}$$

Then using (170) we obtain

$$\mathbf{X}_F^T(n)\begin{bmatrix} \mathbf{P}^{\perp}_{1,N}(n-1) & \underline{0} \\ \underline{0}^T & 0 \end{bmatrix}\mathbf{X}_F(n) = \Sigma_F(n) - \mathbf{e}_F(n)\mathbf{e}_F^T(n|n-1) \tag{173}$$

Finally, simplifying (173) we find

$$\Sigma_F(n-1) = \Sigma_F(n) - \mathbf{e}_F(n)\mathbf{e}_F^T(n|n-1) \tag{174}$$

or

$$\Sigma_F(n) = \Sigma_F(n-1) + \mathbf{e}_F(n)\mathbf{e}_F^T(n|n-1) \tag{175}$$

Thus all the parameters for the forward multichannel prediction problem can be updated from the values obtained at the end of the $(n-1)$ recursion.

5. Backward Filter Update

The update procedure for the backward filter is similar to that for the forward filter. From (108) we have

$$\mathbf{B}(n) = \mathbf{K}_{0,N-1}(n)\mathbf{X}_B(n) \tag{176}$$

To find $\mathbf{B}(n)$ in terms of $\mathbf{B}(n-1)$ we use (134) with $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{U} = \mathbf{Y}_{0,N-1}(n)$ and postmultiply by $\mathbf{X}_B(n)$ to obtain

$$\begin{aligned} &\begin{bmatrix} \mathbf{K}_{0,N-1}(n-1) & \underline{\mathbf{0}} \\ -\underline{\mathbf{y}}^T_{0,N-1}\mathbf{K}_{0,N-1}(n-1) & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_B(n-1) \\ \mathbf{x}^T_B(n) \end{bmatrix} \\ = &\begin{bmatrix} \mathbf{K}_{0,N-1}(n) \\ 0 \end{bmatrix} \mathbf{X}_B(n) - \begin{bmatrix} \mathbf{K}_{0,N-1}(n)\underline{\pi}(n) \\ -1 \end{bmatrix} \frac{\underline{\pi}^T(n)\mathbf{P}^{\perp}_{0,N-1}(n)\mathbf{X}_B(n)}{\underline{\pi}^T(n)\mathbf{P}^{\perp}_{0,N-1}(n)\underline{\pi}(n)} \end{aligned} \tag{177}$$

Then using only the upper partition of this equation we obtain

$$\mathbf{B}(n-1) = \mathbf{B}(n) - \underline{\mathbf{g}}(n)\frac{\mathbf{e}^T_B(n)}{\gamma(n)} \tag{178}$$

or

$$\mathbf{B}(n) = \mathbf{B}(n-1) + \underline{\mathbf{g}}(n)\frac{\mathbf{e}^T_B(n)}{\gamma(n)} \tag{179}$$

To compute $\mathbf{e}_B(n)$ before having $\mathbf{B}(n-1)$ we substitute (179) in

$$\mathbf{e}_B(n) = \mathbf{x}_B(n) - \mathbf{B}^T(n)\underline{\mathbf{y}}_{0,N-1}(n) \tag{180}$$

to find

$$\mathbf{e}^T_B(n) = \mathbf{e}^T_B(n|n-1) - \underline{\mathbf{y}}^T_{0,N-1}(n)\underline{\mathbf{g}}(n)\frac{\mathbf{e}^T_B(n)}{\gamma(n)} \tag{181}$$

where we define

$$\mathbf{e}_B(n|n-1) = \mathbf{x}_B(n) - \mathbf{B}^T(n-1)\underline{\mathbf{y}}_{0,N-1}(n) \tag{182}$$

Now (181) can be simplified to

$$\mathbf{e}_B^T(n) = \mathbf{e}_B^T(n|n-1) + (\gamma(n) - 1)\frac{\mathbf{e}_B^T(n)}{\gamma(n)} = \mathbf{e}_B^T(n|n-1)\gamma(n) \tag{183}$$

Therefore

$$\mathbf{B}(n) = \mathbf{B}(n-1) + \underline{\mathbf{g}}(n)\mathbf{e}_B^T(n|n-1) \tag{184}$$

We note here that the update of $\underline{\mathbf{g}}(n)$ depends on $\mathbf{B}(n)$ and vice-versa, but we will address that problem shortly.

To compute $\Sigma_B(n)$ we use (71) with $\mathbf{U} = \mathbf{Y}_{0,N-1}(n)$, $\mathbf{V} = \underline{\pi}(n)$ and $\mathbf{z} = \mathbf{y} = \mathbf{X}_B(n)$, and also (75) to obtain

$$\mathbf{X}_B^T(n)\mathbf{P}^{\perp}_{0,N-1,\underline{\pi}}(n)\mathbf{X}_B(n) = \Sigma_B(n) - \frac{\mathbf{e}_B(n)\mathbf{e}_B^T(n)}{\gamma(n)} \tag{185}$$

Then using (183)

$$\mathbf{X}_B^T(n)\begin{bmatrix} \mathbf{P}^{\perp}_{0,N-1}(n-1) & \underline{0} \\ \underline{0}^T & 0 \end{bmatrix}\mathbf{X}_B(n) = \Sigma_B(n) - \mathbf{e}_B(n)\mathbf{e}_B^T(n|n-1) \tag{186}$$

and simplifying we find

$$\Sigma_B(n-1) = \Sigma_B(n) - \mathbf{e}_B(n)\mathbf{e}_B^T(n|n-1) \tag{187}$$

or

$$\Sigma_B(n) = \Sigma_B(n-1) + \mathbf{e}_B(n)\mathbf{e}_B^T(n|n-1) \tag{188}$$

We are now ready to solve the problem of mutual dependence between $\underline{\mathbf{g}}(n)$ and $\mathbf{B}(n)$. For this refer to (161) and (162). Substituting (184) in (162) we obtain

$$\underline{\mathbf{g}}(n) = \mathbf{M}(n) + \mathbf{B}(n-1)\mathbf{m}(n) + \underline{\mathbf{g}}(n)\mathbf{e}_B^T(n|n-1)\mathbf{m}(n) \tag{189}$$

Now note that since

$$\mathbf{e}_B^T(n|n-1)\mathbf{m}(n) = \mathbf{e}_B^T(n|n-1)\Sigma_B^{-1}(n)\mathbf{e}_B(n) \tag{190}$$

is a scalar, (189) becomes

$$\underline{\mathbf{g}}(n) = [\mathbf{M}(n) + \mathbf{B}(n-1)\mathbf{m}(n)](1 - \mathbf{e}_B^T(n|n-1)\mathbf{m}(n))^{-1} \tag{191}$$

and the problem is solved.

6. Angle Parameter Update

The final relation need to complete the recursion is an update formula for $\gamma(n)$. This update is performed in a manner similar to the update of $\underline{\mathbf{g}}(n)$; that is we compute the angle parameter related to the data matrix $\mathbf{Y}_{0,N}(n)$ using two different approaches and then equate them. By the same procedure that lead to (118) we define

$$\gamma'(n) = 1 - \underline{\mathbf{y}}_{0,N}^T(n)\mathbf{K}_{0,N}(n)\underline{\pi}(n) \tag{192}$$

Then using (154) for $\mathbf{K}_{0,N}(n)\underline{\pi}(n)$ and partitioning $\underline{\mathbf{y}}_{0,N}(n)$ into

$$\underline{\mathbf{y}}_{0,N}(n) = \left[\mathbf{x}_F^T(n), \underline{\mathbf{y}}_{1,N}^T(n)\right] \Psi_F \tag{193}$$

we obtain

$$\gamma'(n) = 1 - \left[\mathbf{x}_F^T(n), \underline{\mathbf{y}}_{1,N}^T(n)\right] \underbrace{\Psi_F \Psi_F^T}_{\mathbf{I}} \underline{\mathbf{g}}'(n) \tag{194}$$

and using (152) and simplifying we find

$$\gamma'(n) = \gamma(n-1) - \mathbf{e}_F^T(n)\Sigma_F^{-1}(n)\mathbf{e}_F(n) \tag{195}$$

If we now use (157) for $\mathbf{K}_{0,N}(n)\underline{\pi}(n)$ and partition $\underline{\mathbf{y}}_{0,N}(n)$ as

$$\underline{\mathbf{y}}_{0,N}(n) = \left[\underline{\mathbf{y}}_{0,N-1}^T(n), \mathbf{x}_B^T(n)\right] \Psi_B \tag{196}$$

we obtain

$$\gamma'(n) = 1 - \left[\underline{\mathbf{y}}_{0,N-1}^T(n), \mathbf{x}_B^T(n)\right] \underbrace{\Psi_B \Psi_B^T}_{\mathbf{I}} \underline{\mathbf{g}}''(n) \tag{197}$$

This can be simplified to

$$\gamma'(n) = \gamma(n) - \mathbf{e}_B^T(n)\Sigma_B^{-1}(n)\mathbf{e}_B(n) \tag{198}$$

but using (183) and (161) we obtain

$$\gamma(n) = \gamma'(n)[1 - \mathbf{e}_B^T(n|n-1)\mathbf{m}(n)]^{-1} \tag{199}$$

where everything in the right side of (199) is available.

Since $\Sigma_B^{-1}(n)$ does not appear explicitly in any of the updates we need only to update $\Sigma_F^{-1}(n)$. This can be easily done using the matrix inversion lemma. The specific procedure is outlined below.

7. Inverse Matrix Update

Although all the quantities have now been derived that permit the recursion to continue, we note that the inverse matrix Σ_F^{-1}, and not the matrix itself occurs in the recursions. This is a fairly small matrix and could be inverted directly. However it is more efficient to also compute the inverses recursively. This is easily accomplished using the matrix inversion lemma [Ref. 5, 7]. We have from (175)

$$\Sigma_F(n) = \Sigma_F(n-1) + \mathbf{e}_F(n)\mathbf{e}_F^T(n|n-1) \tag{200}$$

or using (183) we have

$$\Sigma_F(n) = \Sigma_F(n-1) + \frac{\mathbf{e}_F^T(n)\mathbf{e}_F(n)}{\gamma(n-1)} \tag{201}$$

The matrix inversion lemma states that if

$$\mathbf{A} = \mathbf{E} + \mathbf{F}\mathbf{G}^{-1}\mathbf{F}^T \tag{202}$$

then

$$\mathbf{A}^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{F}[\mathbf{G} + \mathbf{F}^T\mathbf{E}^{-1}\mathbf{F}]^{-1}\mathbf{F}^T\mathbf{E}^{-1} \tag{203}$$

For

$$\mathbf{A} = \Sigma_F(n) \tag{204}$$

$$\mathbf{E} = \Sigma_F(n-1) \tag{205}$$

$$\mathbf{F} = \mathbf{e}_F(n) \tag{206}$$

$$\mathbf{G} = \gamma(n-1) \tag{207}$$

we obtain

$$\Sigma_F^{-1}(n) = \Sigma_F^{-1}(n-1) - \frac{\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)\mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)}{\gamma(n-1) + \mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)} \tag{208}$$

The computation of $\Sigma_F^{-1}(n)$ using the matrix lemma amounts to $1.5(M+1)^2+2.5(M+1)$ multiplications and one division.

Although the matrix $\Sigma_B^{-1}(n)$ could be computed in a similar way, the inverse of $\Sigma_B(n)$ is not needed for the 2-D filter.

H. ALGORITHM SUMMARY

1. Computational Complexity

The computational cost of the algorithm depends on the shape and size of the filter mask. For each mask we must define, as mentioned before, both a forward and a backward multichannel signal a number of channels equal to the number of new points acquired or dropped off by the mask. Call this number K_1. For the quarter plane filter we used $K_1 = M + 1$. Now further define K_2 as the number of coefficients in the 2-D filter mask (for the case used in the derivation $K_2 = (M+1)(N+1) - 1$). The computational cost of the algorithm depends only on these two numbers. Note that the use of the permutation matrices only changes the ordering of the elements in the matrices affected by them, allowing the procedure to be used for any shape and size of filter. The permutation can be obtained without any multiplications, hence it will not be considered in this analysis. We also mentioned before the use of a forgetting factor λ in the cost function to handle nonstationary signals. The effect of this constant in the final algorithm shows up only in the computation of the inverse error covariance matrix for the forward multichannel filter:

$$\Sigma_F^{-1}(n) = \lambda^{-1}\left(\Sigma_F^{-1}(n-1) - \frac{\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)\mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)}{\lambda\gamma(n-1) + \mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)}\right) \tag{209}$$

This increases the computational cost by K1 multiplications. A detailed count of the number of operations required by the algorithm is given in subsection 3 below. The

method used for the 1-D RLS can be applied to 2-D signals by forming the $K_2 \times K_2$ 2-D deterministic correlation matrix and the $K_2 \times 1$ vector of deterministic cross-correlation terms between the desired filter response and the filter inputs. This form of 2-D RLS algorithm requires $1.5K_2^2 + 4.5K_2$ operations per iteration which increases quadratically with K_2.

2. Initial Conditions

The recursive implementation of the algorithm requires some initialization for the variables used. The assumption that the signal is zero before iteration $n = 0$ is reasonable and suggests that all the filter parameters including those of the gain filter, should all be set to zero. This choice of initial conditions implies that the angle parameter $\gamma(0)$ must be set to 1.0 since all of the subspaces associated with previous data are the null space. However, a positive forward prediction error energy is necessary for the algorithm to start. The initial conditions used are therefore

$$\underline{\mathbf{a}}(0) = \underline{0} \tag{210}$$

$$\mathbf{F}(0) = \mathbf{O} \tag{211}$$

$$\mathbf{B}(0) = \mathbf{O} \tag{212}$$

$$\underline{\mathbf{g}}(0) = \underline{0} \tag{213}$$

$$\gamma(0) = 1 \tag{214}$$

$$\Sigma_F^{-1} = \frac{1}{\delta}\mathbf{I}_{M' \times M'} \tag{215}$$

$$\delta = \textit{small positive constant} \tag{216}$$

3. Iteration at time n and Required Arithmetic Operations

The terms to be computed at each iteration, and their formulas are given below.

– A priori 2-D prediction error (K_2 operations)

$$\epsilon_1(n|n-1) = y_1(n) - \underline{\mathbf{y}}_{1,N}^T(n)\underline{\mathbf{a}}(n-1) \tag{217}$$

- 2-D filter update (K_2 operations)

$$\underline{\mathbf{a}}(n) = \underline{\mathbf{a}}(n-1) + \underline{\mathbf{g}}(n-1)e_1(n|n-1) \tag{218}$$

- A priori multichannel forward prediction error (K_1K_2 operations)

$$\mathbf{e}_F(n|n-1) = \mathbf{x}_F(n) - \mathbf{F}^T(n-1)\underline{\mathbf{y}}_{1,N}(n) \tag{219}$$

- Multichannel forward prediction error (K_1 operations)

$$\mathbf{e}_F(n) = \mathbf{e}_F(n|n-1)\gamma(n-1) \tag{220}$$

- Inverse error covariance matrix for the multichannel forward filter ($1.5K_1^2 + 2.5K_1$ operations)

$$\Sigma_F^{-1}(n) = \Sigma_F^{-1}(n-1) - \frac{\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)\mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)}{\gamma(n-1) + \mathbf{e}_F^T(n)\Sigma_F^{-1}(n-1)\mathbf{e}_F(n)} \tag{221}$$

- Multichannel forward filter update (K_1K_2 operations)

$$\mathbf{F}(n) = \mathbf{F}(n-1) + \underline{\mathbf{g}}(n-1)\mathbf{e}_F^T(n|n-1) \tag{222}$$

- Extended gain transversal filter ($K_1^2 + K_1K_2$ operations)

$$\underline{\mathbf{g}}''(n) = \begin{bmatrix} \mathbf{M}(n) \\ \mathbf{m}(n) \end{bmatrix} = \Psi_B\Psi_F^T \left(\begin{bmatrix} \mathbf{O}_{M'} \\ \underline{\mathbf{g}}(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{I}_{M'\times M'} \\ -\mathbf{F}(n) \end{bmatrix} \Sigma_F^{-1}\mathbf{e}_F(n) \right) \tag{223}$$

- Extended angle parameter (K_1 operations using previous results)

$$\gamma'(n) = \gamma(n-1) - \mathbf{e}_F^T(n)\Sigma_F^{-1}(n)\mathbf{e}_F(n) \tag{224}$$

- A priori multichannel backward prediction error (K_1K_2 operations)

$$\mathbf{e}_B(n|n-1) = \mathbf{x}_B^T(n) - \mathbf{B}^T(n-1)\underline{\mathbf{y}}_{0,N-1}^T(n) \tag{225}$$

- Angle parameter ($K_1 + 1$ operations)

$$\gamma(n) = \gamma'(n)[1 - \mathbf{e}_B^T(n|n-1)\mathbf{m}(n)]^{-1} \tag{226}$$

– Gain transversal filter ($K_1K_2 + K_2$ operations using previous results)

$$\underline{\mathbf{g}}(n) = [\mathbf{M}(n) + \mathbf{B}(n-1)\mathbf{m}(n)]\,(1 - \mathbf{e}_B^T(n|n-1)\mathbf{m}(n))^{-1} \tag{227}$$

– Multichannel backward filter update (K_1K_2 operations)

$$\mathbf{B}(n) = \mathbf{B}(n-1) + \underline{\mathbf{g}}(n)\mathbf{e}_B^T(n|n-1) \tag{228}$$

The total number of operations (multiplications or divisions) required per iteration by the algorithm is

$$2.5K_1^2 + 6K_1K_2 + 4.5K_1 + 3K_2 + 1 \tag{229}$$

IV. EXTENSIONS, APPLICATIONS AND RESULTS

The 2-D FRLS algorithm was developed in the previous chapter for a 2-D prediction filter whose *observed* signal was the same as the sequence under the filter mask. In this particular case the coefficients of $\underline{\mathbf{a}}(n)$ for the 2-D filter, are identical to the coefficients of the first column $\mathbf{f}_1(n)$ of the multichannel filter $\underline{\mathbf{F}}(n)$. To see this, note that by definition, the error energy for the multichannel forward prediction filter (87) can be rewritten as the summation of the error covariance associated with each of the (M+1) channels of the forward prediction filter, where the first term is $\epsilon_1(n)$, i.e., the sum of squared errors for the first channel (44). We can rewrite (87) as

$$tr[\Sigma_F(n)] = tr[\mathbf{E}_F^T(n)\mathbf{E}_F(n)] = \epsilon_1(n) + \Upsilon_F(n) \tag{230}$$

where $\Upsilon_F(n)$ is independent of the coefficients in $\underline{\mathbf{f}}_1(n)$. The 2-D filter coefficients and the coefficients in the first column of the forward multichannel filter are the result of minimizing the same cost function and are thus identical.

We will now consider the case when the data sequence under the prediction mask is distinct from the *observed* sequence (general FIR Wiener filter) and also the case when the filter mask covers not only observation data, but also other input data sequence (ARMA model). Following that we will present the results of computer simulations to illustrate the applications of the adaptive algorithms for 2-D signal processing.

A. GENERAL FIR WIENER FILTER

The extension of the 2-D FRLS to the general FIR Wiener filtering problem is straightforward to obtain by following the same concepts presented in chapter 3. To be specific we again consider a first quadrant $(N+1) \times (M+1)$ filter here. However, the procedure applies more generally to nonsymmetric half plane and other filters as

discussed in section H of chapter 3. The multichannel notation developed in chapter 3 is used here to define two $K \times L$ 2-D sequences $d_1(n)$ and $y_1(n)$, where $d_1(n)$ is the sequence we want to estimate based upon the input sequence $y_1(n)$. Our goal is to find a prediction filter of the form

$$\hat{d}_1(n) = \underline{\mathbf{y}}_{1,N}^T(n)\,\underline{\mathbf{a}}(n) \tag{231}$$

with $\underline{\mathbf{y}}_{1,N}$ defined as in (39) and $\underline{\mathbf{a}}(n)$ defined as in (41) that minimizes the sum of squared errors

$$\epsilon_1(n) = \sum_{i=0}^{n} [\epsilon_1(i)]^2 \tag{232}$$

where the prediction error $\epsilon_1(n)$ is given by

$$\epsilon_1(n) = d_1(n) - \hat{d}_1(n) \tag{233}$$

This can be written in vector notation as

$$\epsilon_1(n) = \underline{\mathbf{e}}_1^T(n)\underline{\mathbf{e}}_1(n) \tag{234}$$

where

$$\underline{\mathbf{e}}_1(n) \;=\; \underline{\mathbf{d}}_1(n) - \hat{\underline{\mathbf{d}}}_1(n) \tag{235}$$

with $\underline{\mathbf{d}}_1(n)$ defined as $(n+1)$-dimensional vector that contains the *observation* data from the origin up to point n. Then the estimate $\hat{\underline{\mathbf{d}}}_1(n)$ is given by

$$\hat{\underline{\mathbf{d}}}_1(n) = \mathbf{Y}_{1,N}(n)\,\underline{\mathbf{a}}(n) \tag{236}$$

where $\mathbf{Y}_{1,N}(n)$ is the same data matrix as in (47) and (48). The least squares solution for $\underline{\mathbf{a}}(n)$ is once more given by the pseudo-inverse

$$\underline{\mathbf{a}}(n) \;=\; \left(\mathbf{Y}_{1,N}^T(n)\mathbf{Y}_{1,N}(n)\right)^{-1} \mathbf{Y}_{1,N}^T(n)\underline{\mathbf{d}}_1(n) \tag{237}$$

and the estimate of $\underline{\mathbf{d}}_1(n)$ and the prediction error can also be expressed in terms of the projection matrices defined in chapter 3. The estimate $\hat{\underline{\mathbf{d}}}_1(n)$ is the projection of

$\underline{\mathbf{d}}_1(n)$ onto the subspace spanned by the input data

$$\hat{\underline{\mathbf{d}}}_1(n) = \mathbf{P}_{1,N}(n)\underline{\mathbf{d}}_1(n) \tag{238}$$

The error $\underline{\mathbf{e}}_1(n)$ is orthogonal to the estimate $\hat{\underline{\mathbf{d}}}_1(n)$ and is given by

$$\underline{\mathbf{e}}_1(n) = \mathbf{P}^{\perp}_{1,N}(n)\underline{\mathbf{d}}_1(n) \tag{239}$$

The operator $\mathbf{K}_{1,N}(n)$ defined in (55) can be used to rewrite $\underline{\mathbf{a}}(n)$ as

$$\underline{\mathbf{a}}(n) = \mathbf{K}_{1,N}(n)\underline{\mathbf{y}}_1(n) \tag{240}$$

We already know how to obtain $\mathbf{K}_{1,N}(n)$ from $\mathbf{K}_{1,N}(n-1)$ in a efficient way, thus we are able update $\underline{\mathbf{a}}(n-1)$ to $\underline{\mathbf{a}}(n)$ as soon as $d_1(n)$ is available. The complete algorithm is the same as the one summarized in section H of chapter 3 with $y_1(n)$ replaced by $d_1(n)$ in (217).

B. ARMA MODEL

The ARMA version of the 2-D FRLS can be viewed as follows. Let us call the output or *observed* data $y_1(n)$ and the input data $w_1(n)$. For the present let us assume that this latter sequence is also known or observed. Let us separate the coefficients that operate on the two different sequences and call $\underline{\mathbf{a}}(n)$ the vector of AR coefficients of the filter, and $\underline{\mathbf{b}}(n)$ the vector of MA coefficients of the filter. As before, we develop this extension of the 2-D FRLS to ARMA models assuming a first quadrant $(N+1)\times(M+1)$ quarter plane mask for both the AR and MA components of the filter, noting that more general forms are possible. Using the scanning index n defined before, we proceed by defining an ARMA prediction filter of the form

$$\hat{y}_1(n) = \underline{\mathbf{y}}^T_{1,N}(n)\,\underline{\mathbf{a}}(n) + \underline{\mathbf{w}}^T_{1,N}(n)\,\underline{\mathbf{b}}(n) \tag{241}$$

with $\underline{\mathbf{y}}_{1,N}(n)$ and $\underline{\mathbf{w}}_{1,N}(n)$ defined using the same concepts as in (39). We want to find $\underline{\mathbf{a}}(n)$ and $\underline{\mathbf{b}}(n)$, the filter coefficients defined respectively for each mask, to minimize

the sum of squared errors

$$\epsilon_1(n) = \sum_{i=0}^{n} [\epsilon_1(i)]^2 \tag{242}$$

where the prediction error is given by

$$\epsilon_1(n) = y_1(n) - \hat{y}_1(n) \tag{243}$$

This can be written in vector notation as

$$\epsilon_1(n) = \underline{\mathbf{e}}_1^T(n)\underline{\mathbf{e}}_1(n) \tag{244}$$

with

$$\underline{\mathbf{e}}_1(n) \;=\; \underline{\mathbf{y}}_1(n) - \underline{\hat{\mathbf{y}}}_1(n) \tag{245}$$

We can combine the AR and MA coefficients in one single vector $\underline{\mathbf{c}}(n)$ as

$$\underline{\mathbf{c}}(n) \;=\; [\underline{\mathbf{a}}^T(n), \underline{\mathbf{b}}^T(n)]^T \tag{246}$$

The data under the mask can then be expressed as

$$\underline{\mathbf{z}}_{1,N}(n) \;=\; [\underline{\mathbf{y}}_{1,N}^T(n), \underline{\mathbf{w}}_{1,N}^T(n)]^T \tag{247}$$

Now we have for the estimate of $\underline{\mathbf{y}}_1(n)$

$$\underline{\hat{\mathbf{y}}}_1(n) = \mathbf{Z}_{1,N}(n)\,\underline{\mathbf{c}}(n) \tag{248}$$

where $\mathbf{Z}_{1,N}(n)$ is the data matrix

$$\mathbf{Z}_{1,N}(n) = [\mathbf{Y}_{1,N}(n), \mathbf{W}_{1,N}(n)] \tag{249}$$

formed by $\mathbf{Y}_{1,N}(n)$ and $\mathbf{W}_{1,N}(n)$, the data matrices associated respectively with the AR and the MA masks with the same structure as (47) and (48). The least squares solution for $\underline{\mathbf{c}}(n)$ is given by the pseudo-inverse of $\mathbf{Z}_{1,N}(n)$

$$\underline{\mathbf{c}}(n) \;=\; \left(\mathbf{Z}_{1,N}^T(n)\mathbf{Z}_{1,N}(n)\right)^{-1} \mathbf{Z}_{1,N}^T(n)\underline{\mathbf{y}}_1(n) \tag{250}$$

After defining new projection matrices and transversal filter operators associated with the new data matrices, the algorithm to recursively update $\underline{\mathbf{c}}(n)$ closely follows the procedures developed in chapter 3.

C. ARMA MODELING WITH UNKNOWN INPUT

In the previous section on 2-D ARMA modeling, it was assumed that the sequence $w_1(n)$ was known and available. This is an ideal situation which could sometimes exist for example in a system identification problem (see Figure 7). Given both

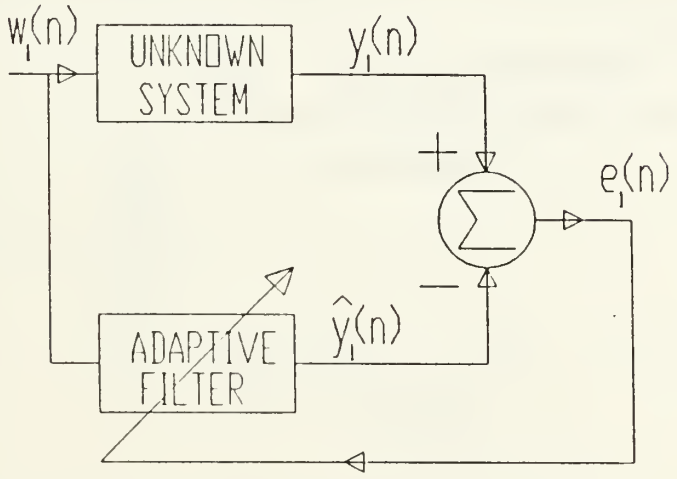

Figure 7. Modeling with Known Input

the input sequence $w_1(n)$, and the output sequence $y_1(n)$ and an assumed linearity and order for the model, we have all the information necessary to identify the unknown system under analysis. Knowledge of the input sequence is not always available, however this is the case, for example, in the problem of estimating the parameters of an ARMA model where $w_1(n)$ is a 2-D white noise sequence. The ARMA parameter estimation problem for unknown input was addressed using recursive algorithms for 1-D signals by embedding the AR and MA estimation in a 2-Channel AR modeling problem [Ref. 2]. The difficulties with the procedures suggested are similar here; the 2-D white noise process $w_1(n)$ is not known and needs to be estimated from the data.

Assume that at some moment n all the data under the MA mask is known except the most *recent* sample of the noise sequence $w_1(n)$. Further assume that the ARMA filter coefficients estimated so far are fairly close to the actual coefficients that characterize the system. The natural choice for an estimate for the unknown noise sample is the error $e_1(n)$, i.e., we expect the error to be zero if the noise sample was known. This procedure is known as "bootstrapping" [Ref. 2] (see Figure 8). It

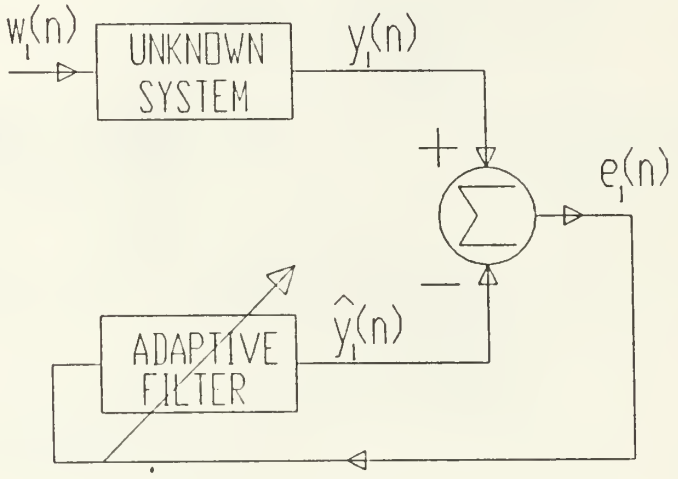

Figure 8. Modeling with Unknown Input

involves two steps. First an estimate $y_1(n)$ is obtained assuming $w_1(n) = 0$ and using the old parameter estimates. Secondly $w_1(n)$ is set equal to $e_1(n)$ and we proceed as in the case of a known input sequence. This method is highly nonlinear and hence very difficult to analyze. However it was found to give reasonable results in practice.

D. SIMULATION RESULTS

The 2-D FRLS algorithm was tested both on computer-generated data and on digitized images. For a baseline reference the 2-D LMS algorithm was also implemented. The synthetic data for the system identification and parameter estimation results was obtained by driving different 2-D transfer functions with computer gen-

erated white Gaussian noise. Most of the tests were performed on 32×32 point 2-D data sequences. Image coding was performed for 256×256 pixel images using the VICOM system for display purposes and a VAX-785 computer for the algorithm implementation. The algorithms were coded in Fortran.

1. 2-D System Identification

The first computer simulation in system identification was performed using a (2×2) MA model. A computer-generated white Gaussian noise sequence was applied both to a filter with known coefficients and to the adaptive filter in the manner of Figure 7. The error between the output of the two filters was used to adjust the coefficients of the adaptive filter. The MA filter had the form

$$\begin{aligned} d(n_1, n_2) &= b(0,0)y(n_1, n_2) + b(0,1)y(n_1, n_2 - 1) \\ &+ b(1,0)y(n_1 - 1, n_2) + b(1,1)y(n_1 - 1, n_2 - 1) \end{aligned} \tag{251}$$

with

$$b(0,0) = 1.0 \tag{252}$$

$$b(0,1) = 0.6 \tag{253}$$

$$b(1,0) = -0.3 \tag{254}$$

$$b(1,1) = 0.3 \tag{255}$$

The rate of convergence is shown in Figure 9. As can be seen, each of the coefficients converged very rapidly to the actual value. The 2-D LMS algorithm was also implemented for this case, but as can be seen in Figure 10 the convergence rate is very slow.

The next simulation was performed using an ARMA model where both the AR and the MA masks were first order nonsymmetrical half plane filters. A computer-generated white Gaussian noise sequence was applied both to a filter with known coefficients and to the adaptive filter. The error between the output of the

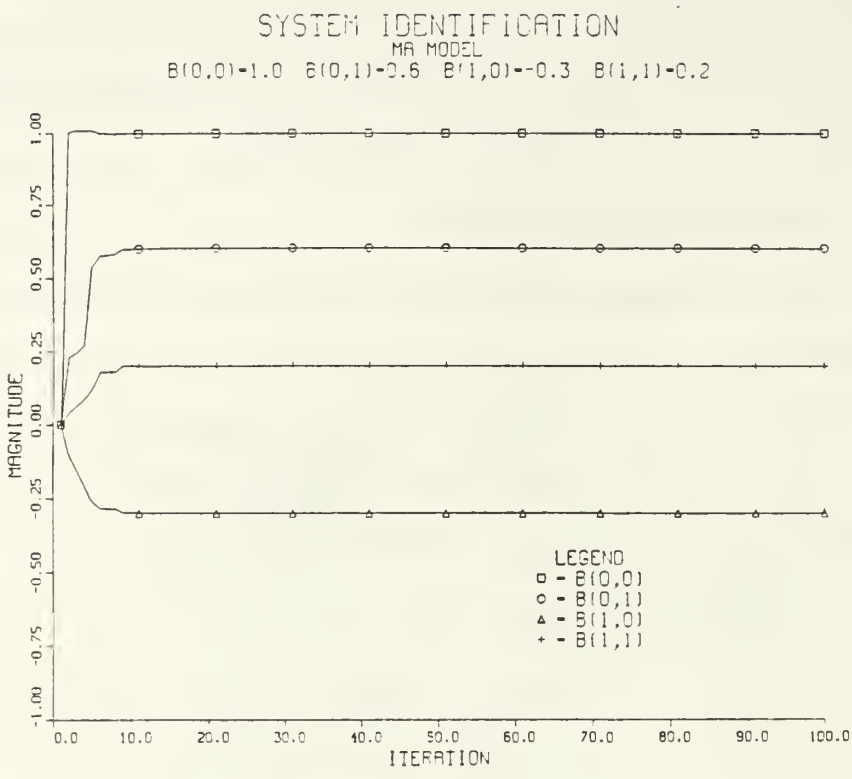

Figure 9. System Identification MA Model

two filters was used to adjust the coefficients of the adaptive filter. The ARMA filter had the form

$$\begin{aligned} y(n_1,n_2) &= a(1,0)y(n_1-1,n_2)+a(-1,1)y(n_1+1,n_2-1) \\ &+ a(0,1)y(n_1,n_2-1)+a(1,1)y(n_1-1,n_2-1) \\ &+ b(1,0)w(n_1-1,n_2)+b(-1,1)w(n_1+1,n_2-1) \\ &+ b(0,1)w(n_1,n_2-1)+b(1,1)w(n_1-1,n_2-1) \end{aligned} \tag{256}$$

with

$$a(1,0) = 0.8 \tag{257}$$

$$a(-1,1) = -0.1 \tag{258}$$

$$a(0,1) = 0.4 \tag{259}$$

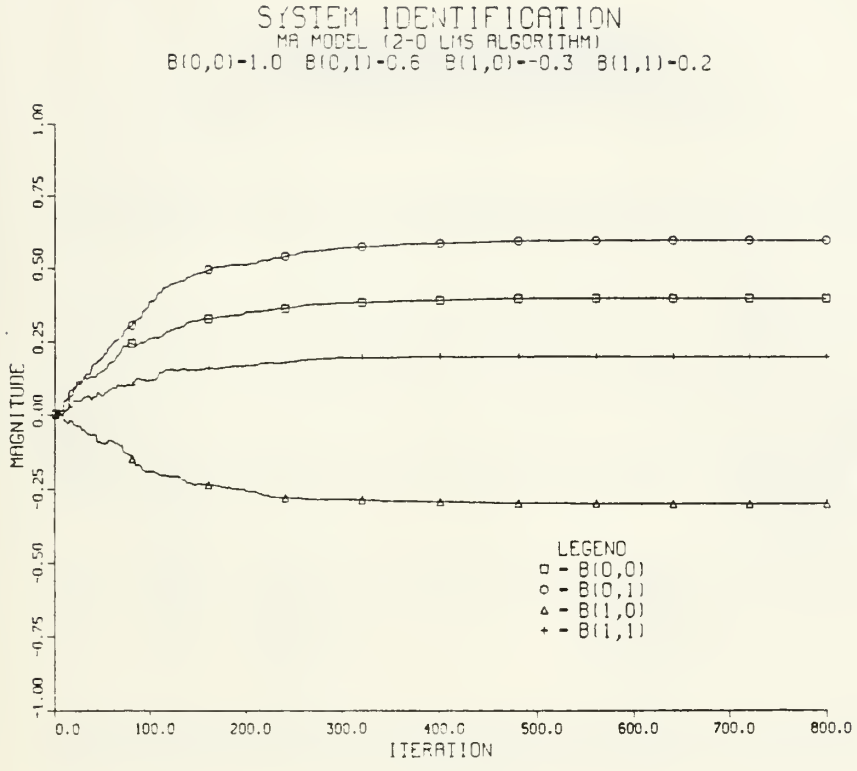

Figure 10. System Identification MA Model (2-D LMS)

$$a(1,1) = -0.5 \tag{260}$$

$$b(1,0) = -0.2 \tag{261}$$

$$b(-1,1) = 0.1 \tag{262}$$

$$b(0,1) = 0.8 \tag{263}$$

$$b(1,1) = -0.5 \tag{264}$$

The rate of convergence is shown in Figure 11 and Figure 12 for both the AR and the MA coefficients. As can be seen there, each of the coefficients converged in about 80 iterations to the true value.

To test the behavior of the algorithm with non-stationary data the same model was run with data obtained by changing some of the coefficients at iteration

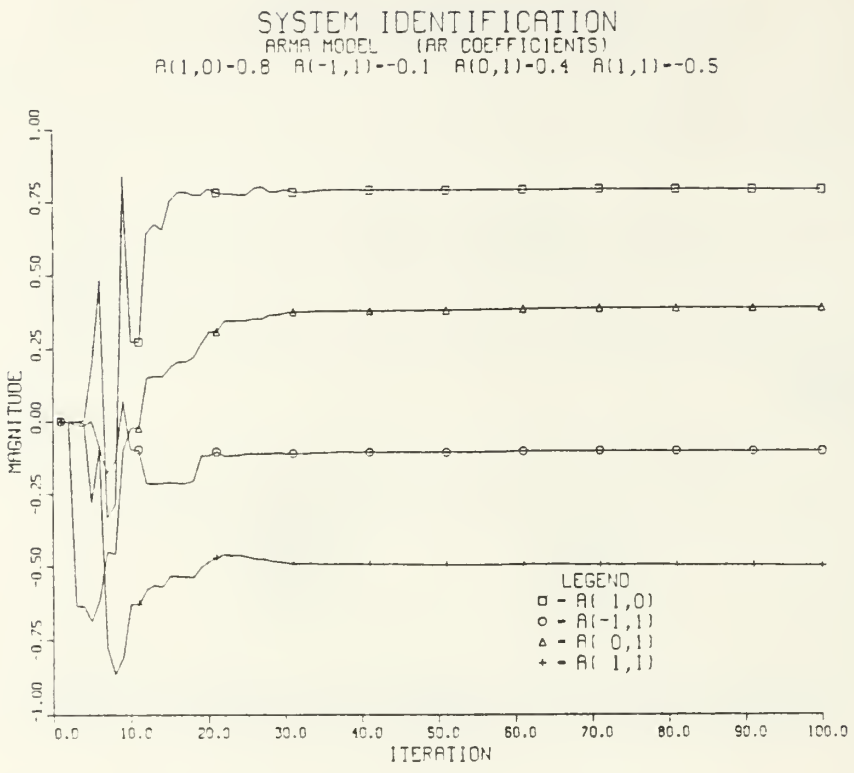

Figure 11. System Identification ARMA Model (AR coeff. stationary data)

120 to

$$a(1,0) = 0.1 \tag{265}$$

$$a(-1,1) = 0.6 \tag{266}$$

$$b(1,0) = 0.4 \tag{267}$$

$$b(0,1) = 0.2 \tag{268}$$

As shown in Figure 13 and Figure 14, the AR and the MA coefficients converged to the true initial values, but at iteration 120 when some of the coefficients were changed the algorithm started slowly tracking the new coefficients. Since no forgetting factor was used here, the algorithm does not forget the initial data and the convergence is very slow. The estimated coefficients remain biased. By using a forgetting factor

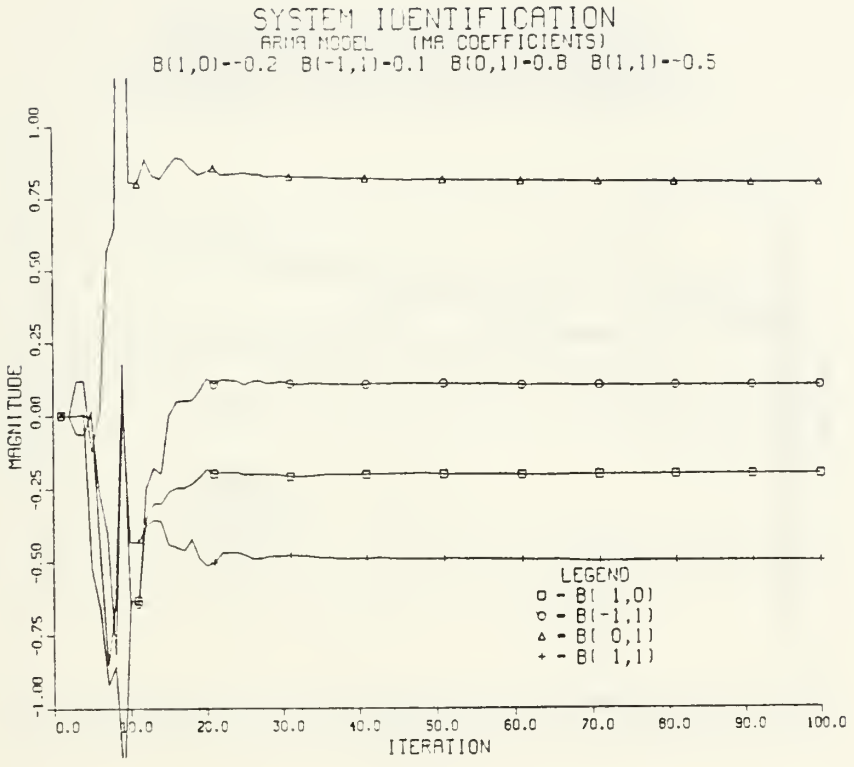

Figure 12. System Identification ARMA Model (MA coeff. stationary data)

of $\lambda = 0.95$ we obtained the results shown in Figure 15 and Figure 16. With the introduction of this forgetting factor the filter coefficients were able to lock on the new coefficients in about 150 more iterations.

2. 2-D Parameter Estimation

The first computer simulation in parameter estimation was performed using a first order nonsymmetric half plane AR model. A computer- generated white Gaussian noise sequence was applied to a 2-D AR filter with known coefficients to obtain the data. The adaptive filter has access only to the AR (output) sequence as shown in Figure 8. The error between the output of the two filters was used to adjust

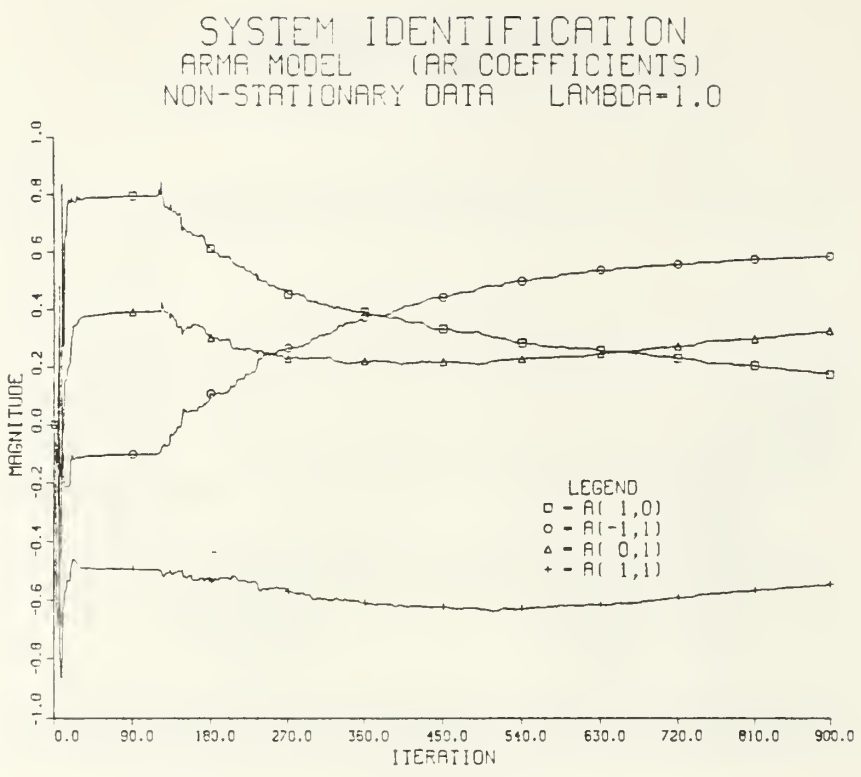

Figure 13. System Identification ARMA Model (AR coeff. nonstationary data) $\lambda = 1.0$

the coefficients of the adaptive filter. The AR filter had the form

$$\begin{aligned} y(n_1, n_2) &= a(1,0)y(n_1 - 1, n_2) + a(-1,1)y(n_1 + 1, n_2 - 1) \\ &+ a(0,1)y(n_1, n_2 - 1) + a(1,1)y(n_1 - 1, n_2 - 1) + w(n_1, n_2) \end{aligned} \tag{269}$$

with

$$a(1,0) = 0.8 \tag{270}$$

$$a(-1,1) = -0.1 \tag{271}$$

$$a(0,1) = 0.4 \tag{272}$$

$$a(1,1) = -0.5 \tag{273}$$

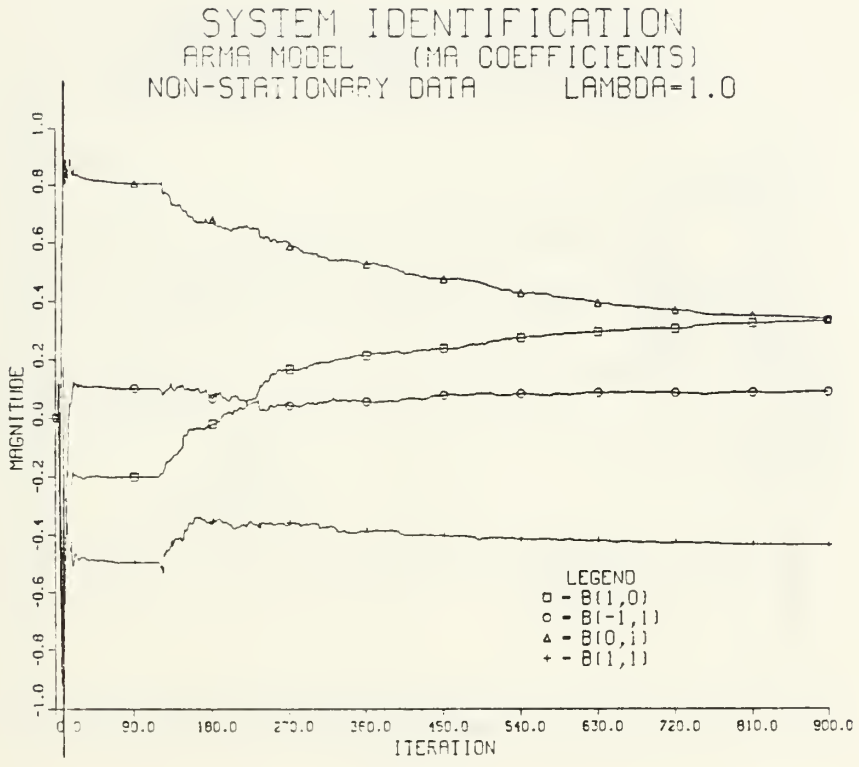

Figure 14. System Identification ARMA Model (MA coeff. nonstationary data) $\lambda = 1.0$

The rate of convergence is shown in Figure 17 and as can be seen each of the coefficients converged to values close to the true values. Since the algorithm does not know the input sequence and has to estimate it, there is a slight variation of the estimated coefficient around the true values. Here again the 2-D LMS algorithm was implemented, but as can be seen from Figure 18 some of the coefficients did not converge even after 900 iterations.

Next the modeling procedure with unknown input was tested for the ARMA case. A first order nonsymmetric half plane mask was used for both the AR and MA coefficients. A computer-generated white Gaussian noise sequence was applied to a filter with known coefficients to obtain the ARMA data. The adaptive filter does not have access to the driving sequence. The error between the output of the two filters

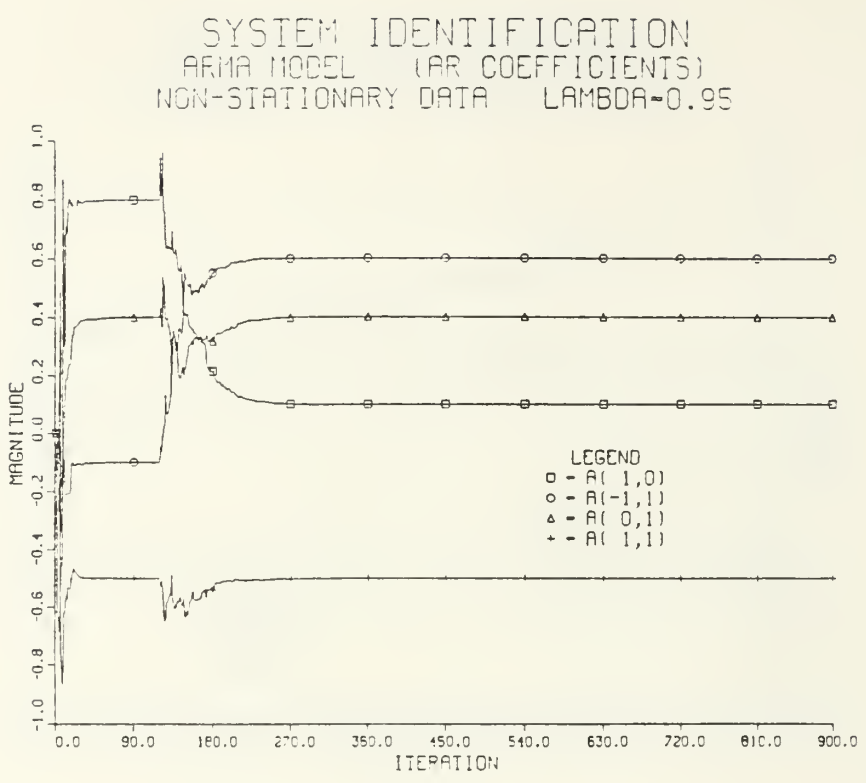

Figure 15. System Identification ARMA Model (AR coeff. nonstationary data) $\lambda = 0.95$

was used to adjust the coefficients of the adaptive filter. The ARMA filter had the form

$$\begin{aligned} y(n_1,n_2) &= a(1,0)y(n_1-1,n_2) + a(-1,1)y(n_1+1,n_2-1) \\ &+ a(0,1)y(n_1,n_2-1) + a(1,1)y(n_1-1,n_2-1) \\ &+ w(n_1,n_2) + b(1,0)w(n_1-1,n_2) + b(-1,1)w(n_1+1,n_2-1) \\ &+ b(0,1)w(n_1,n_2-1) + b(1,1)w(n_1-1,n_2-1) \end{aligned} \tag{274}$$

with

$$a(1,0) = 0.8 \tag{275}$$

$$a(-1,1) = -0.1 \tag{276}$$

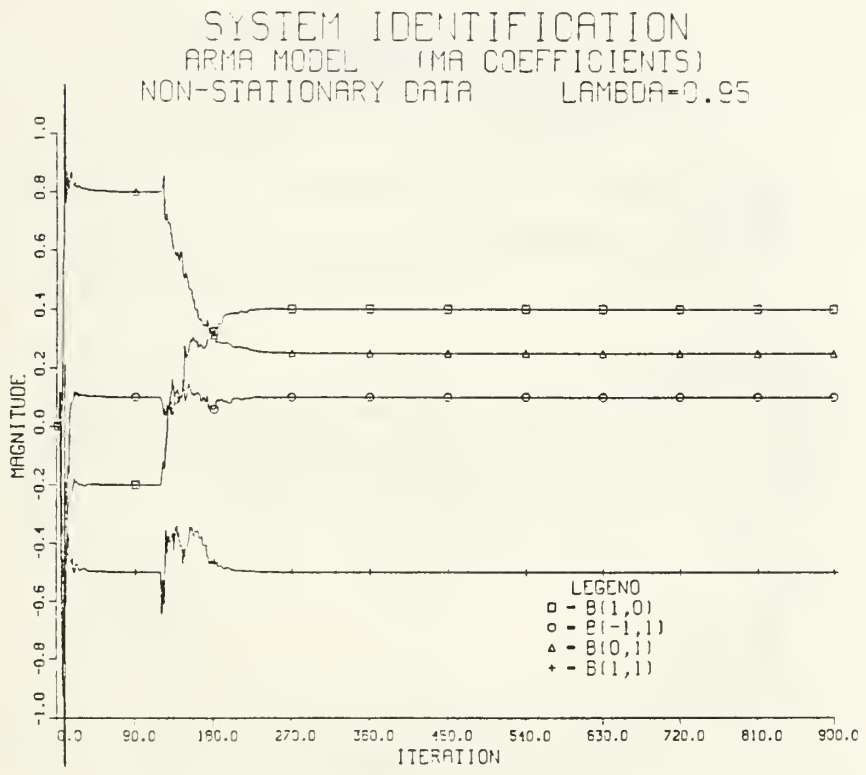

Figure 16. System Identification ARMA Model (MA coeff. nonstationary data) $\lambda = 0.95$

$$a(0,1) = 0.4 \tag{277}$$

$$a(1,1) = -0.5 \tag{278}$$

$$b(1,0) = -0.2 \tag{279}$$

$$b(-1,1) = 0.1 \tag{280}$$

$$b(0,1) = 0.8 \tag{281}$$

$$b(1,1) = -0.5 \tag{282}$$

The rate of convergence is shown in Figure 19 and Figure 20 and as can be seen each of the AR coefficients converged to values close to the true values. This is similar to what happened for the AR parameter estimation problem. The MA coefficients also converged to values acceptably close to the actual MA values, but they show some

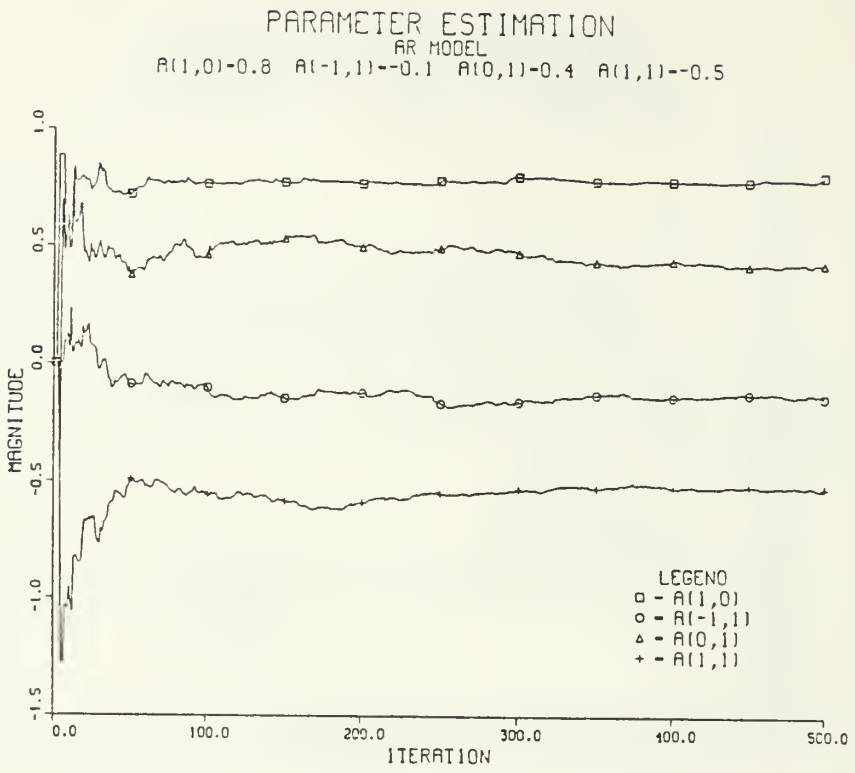

Figure 17. Parameter Estimation AR Model

bias. As in the previous case the algorithm does not have knowledge of the input sequence, hence estimates it using the *bootstrapping* scheme. Although this method is difficult to analyze due to its non-linear nature, the results obtained are encouraging.

3. Image Coding

An image coding problem was also used to test the adaptive algorithm. Two black and white images, Figure 21 and Figure 22, with 256×256 pixels were the 2-D sequences used to perform AR and ARMA parameter estimation as described above. The 2-D LMS algorithm was also applied to the images to estimate the AR parameters. These parameters were then used to form the linear predictor used in the coding and decoding scheme shown in Figure 23. A two level quantizer with the step size value taken from the Max table was used, assuming that the error sequence

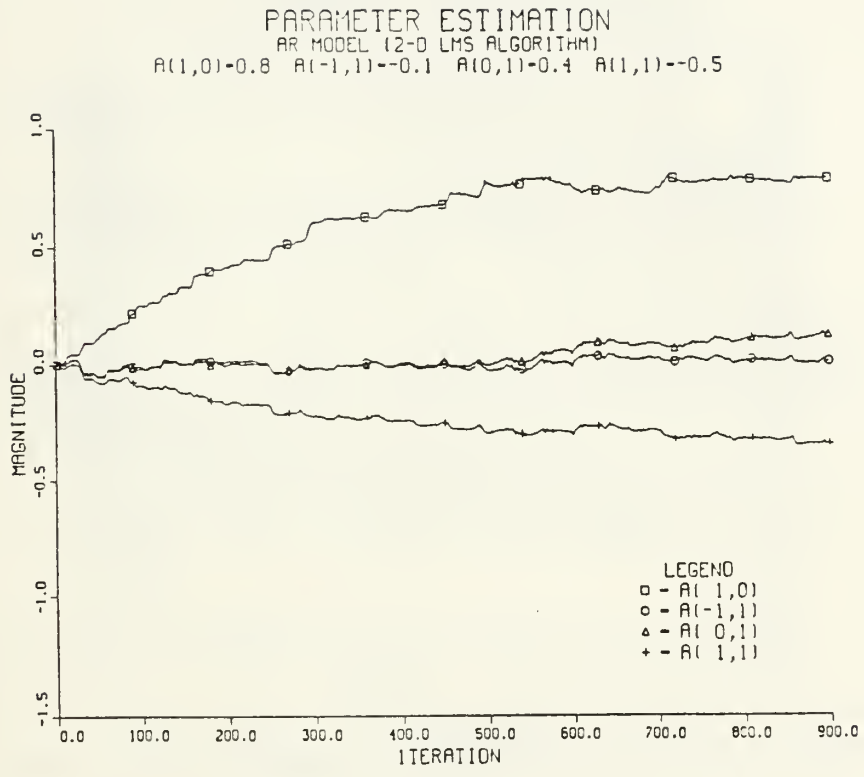

Figure 18. Parameter Estimation AR Model (2-D LMS)

obtained had a Gaussian distribution [Ref. 12]. This resulted in a quantized error sequence corresponding to one bit per pixel. The image reconstruction was performed by driving the inverse filter with the quantized error sequence.

The first image was reconstructed after being encoded using the three different linear predictors and the results are shown in Figures 24, 25 and 26. The error images between the original image and the reconstructed images are shown in Figures 27, 28 and 29. One of the most widely used measures for the performance of a predictive coder is the signal-to-noise ratio (SNR). For a $K \times L$ 2-D sequence $y(n_1, n_2)$, it can be defined as follows:

$$SNR(dB) = 10 \log \frac{\sum_{n_1=0}^{K-1} \sum_{n_2=0}^{L-1} y^2(n_1, n_2)}{\sum_{n_1=0}^{K-1} \sum_{n_2=0}^{L-1} e^2(n_1, n_2)} \tag{283}$$

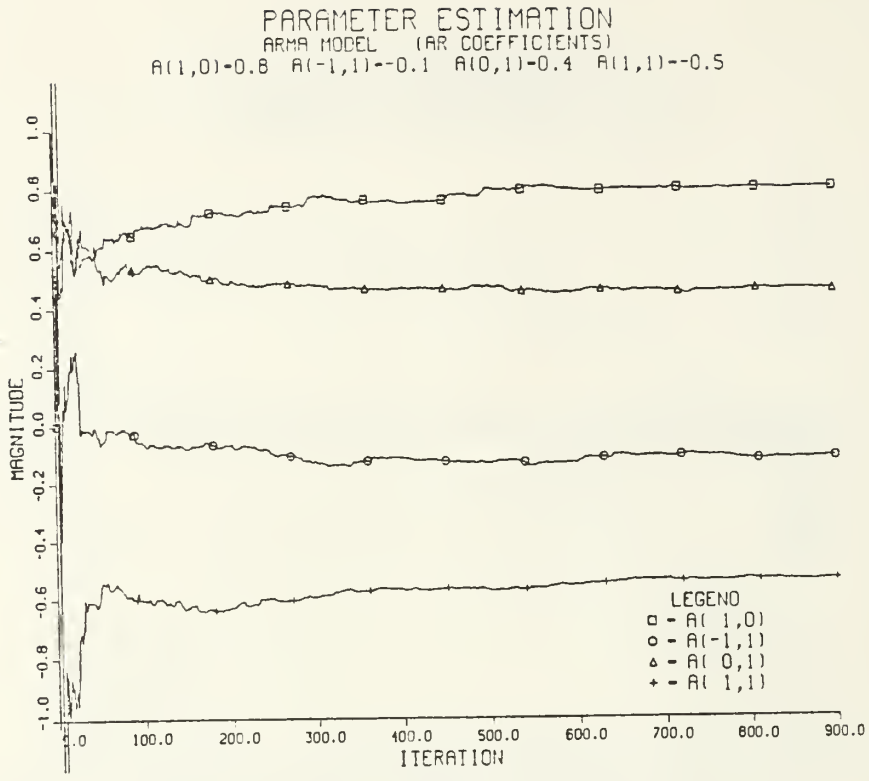

Figure 19. Parameter Estimation ARMA Model (AR coeff.)

The subjective quality of the reconstructed images agrees with the signal-to-noise ratio (SNR) obtained for each case.

$$\text{2-D LMS (AR)} \quad SNR = 15.96\,dB \tag{284}$$

$$\text{2-D FRLS (AR)} \quad SNR = 17.24\,dB \tag{285}$$

$$\text{2-D FRLS (ARMA)} \quad SNR = 18.29\,dB \tag{286}$$

The better performance of the 2-D FRLS when compared with the 2-D LMS is apparent in the results. The improvements obtained for this image with the ARMA model imply that the model was able to fit the image better than the AR model and, thus produce an error sequence that was more nearly white.

The algorithm was also tested on the second image (Figure 22). The results are shown in Figures 30, 31 and 32. The error images between the original image and the reconstructed images are shown in Figures 33, 34 and 35. The subjective

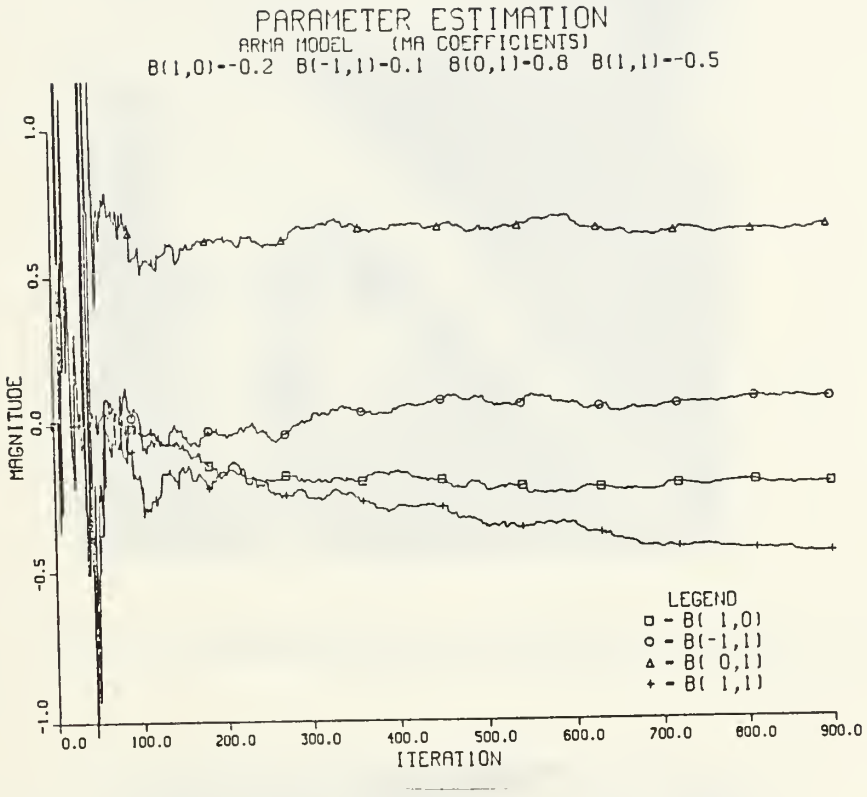

Figure 20. Parameter Estimation ARMA Model (MA coeff.)

quality of the reconstructed images once again agrees with the signal-to-noise ratio (SNR) obtained for each case, but the quality difference between the reconstructed images obtained using different methods is smaller.

$$\text{2-D LMS (AR)} \quad SNR = 17.25\,dB \tag{287}$$

$$\text{2-D FRLS (AR)} \quad SNR = 18.16\,dB \tag{288}$$

$$\text{2-D FRLS (ARMA)} \quad SNR = 18.62\,dB \tag{289}$$

In particular, the improvements obtained with the ARMA model for this case are not as large. This is probably because the second image has a large number of sharp edges, and these are quite difficult to model with any finite order linear model.

Figure 21. Image 1 Original

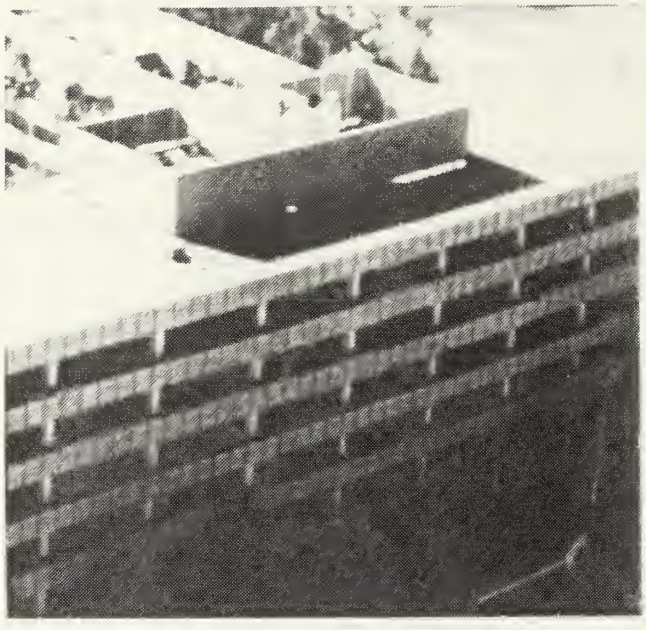

Figure 22. Image 2 Original

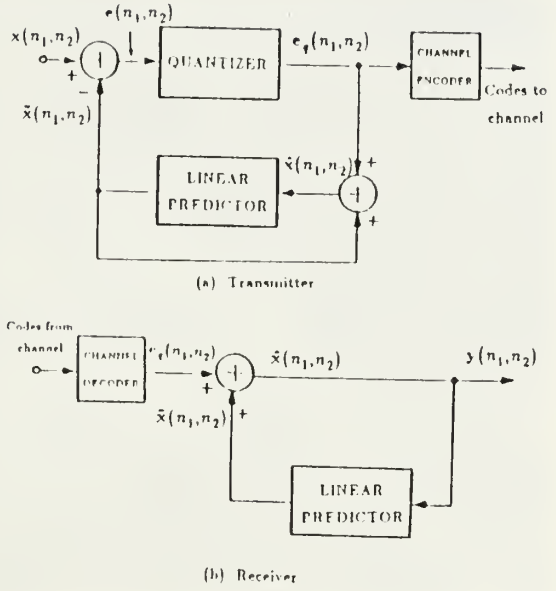

Figure 23. Predictive coding. (taken from [Ref. 6])

Figure 24. Image 1 Reconstructed 2-D LMS (AR)

Figure 25. Image 1 Reconstructed 2-D FRLS (AR)

Figure 26. Image 1 Reconstructed 2-D FRLS (ARMA)

Figure 27. Image 1 Error 2-D LMS (AR)

Figure 28. Image 1 Error 2-D FRLS (AR)

Figure 29. Image 1 Error 2-D FRLS (ARMA)

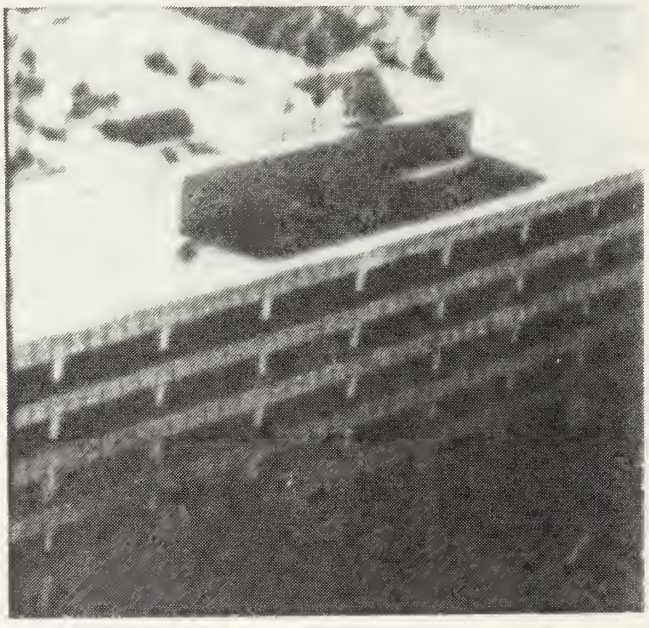

Figure 30. Image 2 Reconstructed 2-D LMS (AR)

Figure 31. Image 2 Reconstructed 2-D FRLS (AR)

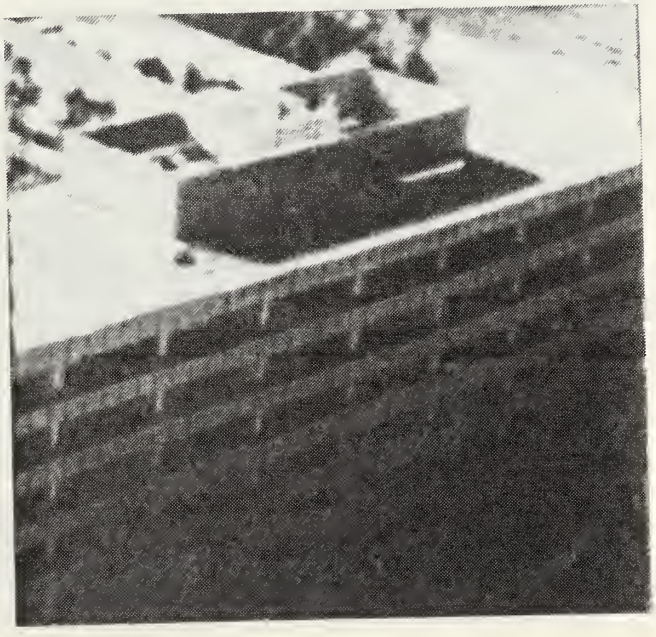

Figure 32. Image 2 Reconstructed 2-D FRLS (ARMA)

Figure 33. Image 2 Error 2-D LMS (AR)

Figure 34. Image 2 Error 2-D FRLS (AR)

Figure 35. Image 2 Error 2-D FRLS (ARMA)

V. CONCLUSIONS AND RECOMMENDATIONS

A Two-Dimensional Fast Recursive Least Squares (2-D FRLS) algorithm was developed using a geometric formulation. The derivation is based on the relation between least squares prediction and the concepts of orthogonality associated with vector spaces. The ordering necessary to develop the recursive algorithm was imposed on the data by using a linear scanning index.

A substantial reduction in computational cost is obtained when compared with the basic 2-D RLS algorithm. The 2-D FRLS algorithm requires on the order of $6K_1K_2$ arithmetic operations per iteration compared with $1.5K_2^2$ for the basic RLS, where K_1 is the number of channels defined for the 2-D FRLS algorithm and K_2 is the total number of coefficients in the 2-D filter. The 2-D LMS algorithm, due to its simplicity, is still more economical than our algorithm in terms of computational cost, but lacks the excellent convergence performance experienced for the 2-D FRLS.

The work described here could be extended in several different ways. First a thorough investigation of the behavior of the algorithm when using finite word length implementation as well as different forgetting factors could be developed. Secondly, techniques used for the 1-D fast RLS to obtain further reductions of the computational cost could be investigated. In particular a variant called the *gain normalized* Fast Transversal Filter [Ref. 6] seems to be applicable to the 2-D FRLS. Its derivation however does not follow directly from the geometrical approach presented here. Finally, the algorithm could be tested in other areas including 2-D parametric spectral estimation.

APPENDIX PROJECTION MATRIX UPDATE

This appendix derives the results

$$\mathbf{P}_{\mathbf{U}\underline{\pi}}(n) = \begin{bmatrix} \mathbf{P}_{\mathbf{U}}(n-1) & \underline{\mathbf{0}} \\ \underline{\mathbf{0}}^T & 1 \end{bmatrix} \tag{290}$$

and

$$\mathbf{K}_{\mathbf{U}\underline{\pi}}(n) = \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n-1) & \underline{\mathbf{0}} \\ -\underline{\mathbf{u}}^T\mathbf{K}_{\mathbf{U}}(n-1) & 1 \end{bmatrix} \tag{291}$$

used in Chapter 3.

Begin by noting that the data matrix $[\mathbf{U}(n), \underline{\pi}(n)]$ can be partitioned as

$$[\mathbf{U}(n), \underline{\pi}(n)] = \begin{bmatrix} \mathbf{U}(n) & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathbf{U}(n-1) & \underline{\mathbf{0}} \\ \underline{\mathbf{u}}^T & 1 \end{bmatrix} \tag{292}$$

where $\underline{\mathbf{u}}$ is the last row of $\mathbf{U}(n)$. $\mathbf{P}_{U\underline{\pi}}(n)$ is defined by

$$\mathbf{P}_{U\underline{\pi}}(n) = [\mathbf{U}(n), \underline{\pi}(n)] \left([\mathbf{U}(n), \underline{\pi}(n)]^T [\mathbf{U}(n), \underline{\pi}(n)] \right)^{-1} [\mathbf{U}(n), \underline{\pi}(n)]^T \tag{293}$$

Then using (292) we have

$$\left([\mathbf{U}(n), \underline{\pi}(n)]^T [\mathbf{U}(n), \underline{\pi}(n)] \right)^{-1} = \begin{bmatrix} \mathbf{U}^T(n-1)\mathbf{U}(n-1) + \underline{\mathbf{u}}\,\underline{\mathbf{u}}^T & \underline{\mathbf{u}} \\ \underline{\mathbf{u}}^T & 1 \end{bmatrix}^{-1} \tag{294}$$

Here we will use the relation for the inverse of a symmetrical matrix by partitioning [Ref. 13]. If:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{295}$$

with $\mathbf{C} = \mathbf{B}^T$ and $\mathbf{A}$ and $\mathbf{D}$ square matrices, then

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^T & -\mathbf{P}\mathbf{Q}^{-1} \\ -\mathbf{Q}^{-1}\mathbf{P}^T & \mathbf{Q}^{-1} \end{bmatrix} \tag{296}$$

with

$$\mathbf{P} = \mathbf{A}^{-1}\mathbf{B}$$

and

$$\mathbf{Q} = \mathbf{D} - \mathbf{C}\mathbf{P}$$

For our case we have

$$\mathbf{A} = \mathbf{U}^T(n-1)\mathbf{U}(n-1) + \underline{\mathbf{u}}\,\underline{\mathbf{u}}^T \tag{297}$$

$$\mathbf{B} = \underline{\mathbf{u}} \tag{298}$$

$$\mathbf{C} = \underline{\mathbf{u}}^T \tag{299}$$

$$\mathbf{D} = 1 \tag{300}$$

The matrix inversion lemma [Ref. 5, 7] is used first to obtain $\mathbf{A}^{-1}$ as follows.

The matrix inversion lemma states that if

$$\mathbf{A} = \mathbf{E} + \mathbf{F}\mathbf{G}^{-1}\mathbf{F}^T \tag{301}$$

then

$$\mathbf{A}^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{F}\left[\mathbf{G} + \mathbf{F}^T\mathbf{E}^{-1}\mathbf{F}\right]^{-1}\mathbf{F}^T\mathbf{E}^{-1} \tag{302}$$

Now taking

$$\mathbf{E} = \left[\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right] \tag{303}$$

$$\mathbf{F} = \underline{\mathbf{u}} \tag{304}$$

$$\mathbf{G} = 1 \tag{305}$$

we have

$$\mathbf{A}^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\underline{\mathbf{u}}\left[1 + \underbrace{\underline{\mathbf{u}}^T\mathbf{E}^{-1}\underline{\mathbf{u}}}_{K=scalar}\right]^{-1}\underline{\mathbf{u}}^T\mathbf{E}^{-1} \tag{306}$$

$$= \mathbf{E}^{-1} - \frac{\mathbf{E}^{-1}\,\underline{\mathbf{u}}\,\underline{\mathbf{u}}^T\mathbf{E}^{-1}}{1+K} \tag{307}$$

where $K = \underline{\mathbf{u}}^T\mathbf{E}^{-1}\underline{\mathbf{u}}$. The upper left partition of (296) becomes

$$\mathbf{A}^{-1} + \mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^T = \mathbf{A}^{-1} + \mathbf{A}^{-1}\underline{\mathbf{u}}\left[1 - \underline{\mathbf{u}}^T\mathbf{A}^{-1}\underline{\mathbf{u}}\right]^{-1}\underline{\mathbf{u}}^T\mathbf{A}^{-1} \tag{308}$$

$$= \mathbf{E}^{-1} - \frac{\mathbf{E}^{-1}\,\underline{\mathbf{u}}\,\underline{\mathbf{u}}^T\mathbf{E}^{-1}}{1+K} \tag{309}$$

$$+ \left[\mathbf{E}^{-1}\underline{\mathbf{u}} - \frac{\mathbf{E}^{-1}\,\underline{\mathbf{u}}K}{1+K}\right](K+1)\left[\underline{\mathbf{u}}^T\mathbf{E}^{-1} - \frac{K\underline{\mathbf{u}}^T\mathbf{E}^{-1}}{1+K}\right]$$

$$= \mathbf{E}^{-1} - \frac{\mathbf{E}^{-1}\,\underline{\mathbf{u}}\,\underline{\mathbf{u}}^T\mathbf{E}^{-1}}{1+K} + \frac{\mathbf{E}^{-1}\,\underline{\mathbf{u}}\,\underline{\mathbf{u}}^T\mathbf{E}^{-1}}{1+K} \tag{310}$$

$$= \mathbf{E}^{-1} = \left[\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right]^{-1} \tag{311}$$

To find the other partitions we write

$$\mathbf{P} = \mathbf{A}^{-1}\mathbf{B} = \mathbf{E}^{-1}\underline{\mathbf{u}} - \frac{\mathbf{E}^{-1}\underline{\mathbf{u}}K}{1+K} \tag{312}$$

$$= \frac{\mathbf{E}^{-1}\underline{\mathbf{u}}}{1+K} = \frac{\left(\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right)^{-1}\underline{\mathbf{u}}}{1+K}$$

and

$$\mathbf{Q} = \mathbf{D} - \mathbf{C}\mathbf{P} = 1 - \frac{\underline{\mathbf{u}}^T\mathbf{E}^{-1}\underline{\mathbf{u}}}{1+K} = \frac{1}{1+K} \tag{313}$$

thus we have

$$-\mathbf{P}\mathbf{Q}^{-1} = \left(\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right)^{-1}\underline{\mathbf{u}} \tag{314}$$

Now substituting (311), (313) and (314) in (296) we obtain

$$\mathbf{M}^{-1} = \begin{bmatrix} \left(\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right)^{-1} & -\left(\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right)^{-1}\underline{\mathbf{u}} \\ -\underline{\mathbf{u}}^T\left(\mathbf{U}^T(n-1)\mathbf{U}(n-1)\right)^{-1} & K+1 \end{bmatrix} \tag{315}$$

Postmultiplying this result by the transpose of (292) we obtain

$$\mathbf{K}_{\mathbf{U}\underline{x}}(n) = \begin{bmatrix} \mathbf{K}_{\mathbf{U}}(n-1) & \underline{\mathbf{0}} \\ -\underline{\mathbf{u}}^T\mathbf{K}_{\mathbf{U}}(n-1) & 1 \end{bmatrix} \tag{316}$$

Finally, premultiplying by (292), (293) becomes

$$\mathbf{P}_{\mathbf{U}\underline{x}}(n) = \begin{bmatrix} \mathbf{P}_{\mathbf{U}}(n-1) & \underline{\mathbf{0}} \\ \underline{\mathbf{0}}^T & 1 \end{bmatrix} \tag{317}$$

Q.E.D.

LIST OF REFERENCES

1. Alexander, S. T., Fast adaptive filters: a geometrical approach. *IEEE ASSP Magazine*, p.18-28, October 1986.

2. Friedlander, Benjamin and Lee, Daniel T. L. and Morf, Martin. Recursive least squares ladder estimation algorithms. *IEEE Transactions on ASSP*, ASSP-29, p.627–641, June 1981.

3. Kailath, Thomas and Lev-Ari, Hanoch and Cioffi, John. Least squares adaptive lattice and transversal filters: a unified geometric theory. *IEEE Transactions on Information Theory*, IT-30, p.222–236, March 1984.

4. Friedlander, Benjamin. Recursive lattice forms for spectral estimation. *IEEE Transactions on ASSP*. ASSP-30, p.920–930, December 1982.

5. Haykin, Simon. *Introduction to Adaptive Filters.* Macmillan, New York, 1986.

6. Alexander, S. T. *Adaptive Signal Processing: Theory and Applications.* Springer-Verlag, New York, 1986.

7. Bellanger, Maurice G. *Adaptive Digital Filters and Signal Analysis.* Marcel Dekker, New York, 1987.

8. Messerschmitt, David G. and Honig, Michael L. *Adaptive Filters: Structures.Algorithms and Applications.* Kluwer, Boston, 1984.

9. Hadhoud, Mohiy M. and Thomas, David. The two-dimensional adaptive LMS (TDLMS) algorithm. *IEEE Transactions on Circuits and Systems*, Vol.35, No.5, p.485–494, May 1988.

10. Wilstrup, Steven L. *Adaptive Algorithms for Two Dimensional Filtering.* Master's thesis, Naval Postgraduate School, Monterey, California, September 1988.

11. Dudgeon, Dan E and Mersereau, Russel M. *Multidimensional Digital Signal Processing.* Volume New-Jersey, Prentice-Hall, 1984.

12. El-Shaer, Hamdy T.M. *Multichannel 2-D Power Spectral Estimation and Applications.* Master's thesis, Naval Postgraduate School, Monterey, California, September 1987.

13. Allan D. Kraus. *Matrices for Engineers.* Hemisphere Publisher Corporation, Washington, 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Professor Charles W. Therrien, Code 62Th Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5004	2
4.	Professor Murali Tummala, Code 62Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
5.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor Ralph D. Hippenstiel, Code 62Hi Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
7.	Professor Roberto Cristi, Code 62Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
8.	Direcção do Serviço de Instrução e Treino Edificio da Administração Central de Marinha Praça do Comércio 1100 Lisboa Portugal	1
9.	Lt. Paulo M. D. Mónica de Oliveira Direcção do Serviço de Instrução e Treino Edificio da Administração Central de Marinha Praça do Comércio 1100 Lisboa Portugal	1

10. Lt. Armando Miguel Perez de Jesus Sequeira 2
Direcção do Serviço de Instrução e Treino
Edificio da Administração Central de Marinha
Praça do Comércio
1100 Lisboa
Portugal