# Adaptive Lattice Bilinear Filters

Heung Ki Baik, *Member, IEEE*, and V. John Mathews, *Senior Member, IEEE*

*Abstract*—**This paper presents two fast least squares lattice algorithms for adaptive nonlinear filters equipped with bilinear system models. Bilinear models are attractive for adaptive filtering applications because they can approximate a large class of nonlinear systems adequately, and usually with considerable parsimony in the number of coefficients required. The lattice filter formulation transforms the nonlinear filtering problem into an equivalent multichannel linear filtering problem and then uses multichannel lattice filtering algorithms to solve the nonlinear filtering problem. The lattice filters perform a Gram–Schmidt orthogonalization of the input data and have very good numerical properties. Furthermore, the computational complexity of the algorithms is an order of magnitude smaller than previously available methods. The first of the two approaches is an equation error algorithm that uses the measured desired response signal directly to compute the adaptive filter outputs. This method is conceptually very simple; however, it will result in biased system models in the presence of measurement noise. The second approach is an approximate least squares output error solution. In this case, the past samples of the output of the adaptive system itself are used to produce the filter output at the current time. Results of several experiments that demonstrate and compare the properties of the adaptive bilinear filters are also presented in this paper. These results indicate that the output error algorithm is less sensitive to output measurement noise than the equation error method.**

## I. Introduction

WHILE linear filters and system models have been very useful in a large variety of applications and are conceptually and implementationally very simple, there are several applications in which they will not perform well at all. A simple but pervasive situation where linear filters will not perform adequately involves saturation nonlinearities. Another situation where linear filters will seriously fail involves trying to relate two signals with nonoverlapping spectral components. This paper is concerned with developing adaptive filtering algorithms for nonlinear systems modeled as bilinear systems. System analysis using nonlinear structures has several applications, including those in channel equalization [1], echo cancellation [2], [3], noise cancellation [4], [5], characterizing semiconductor devices [6], modeling biological phenomenon [4], [7], and several others.

A very common system model that has been employed with relatively good success in nonlinear filtering applications is the Volterra system model [8], [9]. Such models express the relationship between the output signal $y(n)$ of the causal discrete-time nonlinear system and its input $x(n)$ using a Volterra series expansion in the input signal as

$$\begin{aligned} y(n) = h_0 &+ \sum_{m_1=0}^{\infty} h_1(m_1)x(n-m_1) \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2(m_1, m_2)x(n-m_1)x(n-m_2) \\ &+ \cdots + \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\cdots\sum_{m_p=0}^{\infty} h_p \\ &\cdot (m_1, m_2, \cdots, m_p)x(n-m_1)x(n-m_2) \\ &\cdots x(n-m_p) + \cdots \end{aligned} \tag{1.1}$$

where $h_p(\cdots)$ is known as the $p$th order Volterra kernel of the nonlinear system [8], [9]. Several researchers have developed adaptive filters based on truncated Volterra series expansion [2]–[4], [10]–[15]. The main problem associated with such filters is the extremely large number of coefficients (and the correspondingly large computational complexity) that is usually required to adequately model the nonlinear system under consideration. An alternate approach pursued in this paper is to use system models involving nonlinear feedback, in which the input–output relationship is governed by a recursive nonlinear difference equation of the type [16]

$$\begin{aligned} y(n) = \sum_{i=1}^{M} P_i[&x(n), \cdots, x(n-N+1), y(n-1), \\ &\cdots, y(n-N+1)] \end{aligned} \tag{1.2}$$

where $P_i[\cdots]$ is an $i$th order polynomial in the variables within the square brackets. Just as linear IIR filters can model many linear systems with more parsimony than FIR filters, there are a large number of nonlinear systems that can be approximated by nonlinear feedback models using a relatively small number of parameters. In such situations, one can expect that the corresponding adaptive filters can be implemented with good computational effi-

Manuscript received November 27, 1990; revised May 18, 1992. The associate editor coordinating the review of this paper and approving it for publication was Dr. J. J. Shynk. This work was supported in part by a Fellowship from the Korean Science Foundation, NSF Grants MIP 8708970 and MIP 8922146, and a University of Utah Faculty Fellow Award. Parts of this paper were presented at the SPIE Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations, San Diego, CA, July 8–13, 1990 and at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Toronto, Canada, May 14–17, 1991.

H. K. Baik was with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112, on leave from the Department of Electronic Engineering, Chonbuk National University, Chonju, Chonbuk, Korea 560-756, while this work was being done.

V. J. Mathews is with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112.

IEEE Log Number 9208244.

ciency. Perhaps the simplest among the nonlinear feedback models is the bilinear system model. The input–output relationship is given by the nonlinear difference equation

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij} x(n-i) \cdot y(n-j) + \sum_{j=1}^{N-1} c_j y(n-j). \quad (1.3)$$

(It should be noted that the upper limits on all the summations are the same only for pedagogical convenience. The extension to the case of an arbitrary number of nonzero coefficients in each summation is straightforward, as will be seen later.)

The simplicity of the bilinear system model when compared with the Volterra system model in (1.1) is obvious. Another attractive feature of the bilinear system models is that they can be used to approximate any Volterra system with arbitrary precision under fairly general conditions [17]. Because of these advantages, bilinear system models have found various applications, including those in control systems, population models, biological systems, economics, etc. An overview of continuous-time bilinear system models and their applications can be found in [18], [19]. In this paper, we present two adaptive lattice filtering algorithms for bilinear filtering. In spite of the potential benefits of such system models, very little work has been done on adaptive filters employing nonlinear feedback models. Among the very few published works are [20]–[24]. The results in [20], [21], [23] involve direct-form structures and employ the conventional recursive least squares adaptation algorithm or its variations, which are computationally very complex. Fast versions of such algorithms will almost certainly suffer from numerical problems. Reference [24] contains a Kalman filter type algorithm for adaptive bilinear filtering when the only unknown parameters are the noise statistics. The approach in [25] performs a Gram–Schmidt orthogonalization of the data. However, implementing an adaptive filter using this method for the structure shown in (1.3) requires $O(N^4)$ operations per time instant. Reference [21] discusses an algorithm involving the simpler least mean square (LMS) adaptive filter. Such algorithms are known for their slow and input-dependent convergence rates. Lattice structures are attractive because of the existence of fast and numerically stable adaptive algorithms. The method presented in [22] involves a lattice structure, but is useful only for a very special class of nonlinear models. We believe that ours is the first successful attempt at deriving adaptive lattice filters that is applicable for the general bilinear system model. Furthermore, we will show in the next section that the methods presented in this paper can be very easily extended to more general nonlinear output feedback structures.

The two algorithms presented in this paper differ only in the way in which the output feedback is accomplished in the adaptive filter. The first approach, which is an equation error adaptive filter, uses the desired response signal directly to compute the adaptive filter output. The other method is an output error algorithm and uses the past samples of the output of the adaptive filter to obtain the current output and results in a suboptimal least squares output error adaptive filter. The former technique results in biased system estimates in the presence of observation noise and is therefore useful when the measurement noise level is small. The output error algorithm has the potential to reduce the effects of the observation noise. It is important to note that our algorithm provides a lattice parameterization of the nonlinear relationship between the input and desired response signals of the adaptive filter and that their computational efficiency will not be useful in applications where direct form parameters are needed. However, there are a large number of applications where the estimate of the desired response signal or the estimation error are of interest (for example, noise cancellation, echo cancellation, etc.) and our algorithms are most useful in such cases.

The rest of the paper is organized as follows. The next section contains the derivation of a lattice structure for bilinear filters. In Section III, we will present the recursive least squares algorithms for adaptive lattice bilinear filtering. Experimental results demonstrating and comparing the properties of the algorithms are presented in Section IV. Finally, the concluding remarks are made in Section V. The equations for converting a set of lattice parameters to those of the direct-form bilinear filters are presented in the Appendix.

## II. A Lattice Structure for Bilinear Filtering

Consider the problem of adaptively estimating the desired response signal $y(n)$ as the response of a bilinear system to its input signal $x(n)$. We will recursively estimate the bilinear system parameters such that error function

$$\xi_N(n) = \sum_{k=1}^{n} \lambda^{n-k}[\, y(k) - \hat{y}_n(k)]^2 \quad (2.1)$$

is minimized at each instant. In this expression $\lambda$, $0 < \lambda \leq 1$, is a constant that controls the memory of the adaptive filter, and $\hat{y}_n(k)$ is an estimate of $y(k)$ based on the parameters of the adaptive filter at time $n$.

We will consider two different approaches to solving the problem, which differ in the way in which $\hat{y}_n(k)$ is evaluated. The first approach is the equation error formulation, in which the adaptive filter output is estimated as

$$\hat{y}_n(n) = \sum_{i=0}^{N-1} a_i(n) x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij}(n) x(n-i) \cdot y(n-j) + \sum_{j=1}^{N-1} c_j(n) y(n-j) \quad (2.2)$$

where $a_i(n)$, $b_{ij}(n)$, and $c_j(n)$ are the coefficients of the adaptive bilinear filter at time $n$. The second approach is

an output error formulation, in which the adaptive filter output is estimated as

$$\hat{y}_n(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij}(n)x(n-i) \cdot \hat{y}_{n-j}(n-j) + \sum_{j=1}^{N-1} c_j(n)\hat{y}_{n-j}(n-j). \quad (2.3)$$

This formulation results in a suboptimal least squares solution in the sense that the adaptive filter coefficients at time $n$ depend on those at the previous times through the estimates $\hat{y}_{n-j}(n-j); j = 1, 2, \cdots, N-1$, as can be seen from the above equation. This approximation is a reasonable one when $\lambda$ is very close to one. In such cases, the coefficients will change very slowly and, therefore, the following approximation is valid:

$$\hat{y}_n(n-j) \simeq \hat{y}_{n-j}(n-j). \quad (2.4)$$

Note that the spirit of this approximation is very similar to that employed in many gradient adaptive output error algorithms for linear IIR filtering [26], [27].

A direct solution to the equation error formulation of the problem may be found using conventional techniques. Let us define the input signal vector, $U_N(n)$ (of size $N^2 + N - 1$), and the coefficient vector, $W_N(n)$, at each time instant $n$ to be

$$U_N(n) = [x(n), \cdots, x(n-N+1), x(n)y(n-1), \cdots, x(n-N+1)y(n-N+1), y(n-1), \cdots, y(n-N+1)]^T \quad (2.5)$$

and

$$W_N(n) = [a_0(n), \cdots, a_{N-1}(n), b_{0,1}(n), \cdots, b_{N-1,N-1}(n), c_1(n), \cdots, c_{N-1}(n)]^T \quad (2.6)$$

respectively. In the above, $(\cdot)^T$ denotes the matrix transpose of $(\cdot)$. Equation (2.1) can now be written in a compact form using vector notation as

$$\xi_N(n) = \sum_{k=1}^{n} \lambda^{n-k}[y(k) - W_N^T(k)U_N(k)]^2. \quad (2.7)$$

The optimal coefficient vector $W_N^*(n)$ which minimizes $\xi_N(n)$ is immediately seen to be [28]

$$W_N^*(n) = R_N^{-1}(n)P_N(n) \quad (2.8)$$

where

$$R_N(n) = \sum_{k=1}^{n} \lambda^{n-k}U_N(k)U_N^T(k) \quad (2.9)$$

and

$$P_N(n) = \sum_{k=1}^{n} \lambda^{n-k}U_N(k)y(k). \quad (2.10)$$

The solution to the output error formulation of the problem can be solved only by iterative techniques, i.e., one needs to solve for $W_N(n)$ at each time. The solution involves simply substituting $\hat{y}_k(k)$ in place of $y(k)$ in (2.5) and iterating equations (2.9), (2.10), and (2.8) at each time.

Both the solutions involve matrix inversion or equivalent techniques and require a large number of numerical operations. The approaches followed in [20], [23] are all variations of this direct approach and require at least $O(N^4)$ arithmetic operations per time instant for recursive implementation. It is conceivable that one can use fast algorithms for solving for $W_N^*(n)$; however, such solutions will almost invariably be prone to numerical problems.

Our approach for developing a lattice structure for bilinear filtering is to transform the nonlinear filtering problem into an equivalent multichannel, but linear, filtering problem. This approach is similar to the development of the adaptive lattice Volterra filter in [15]. The basic idea is to partition the input vector $U_N(n)$ into the following set of $2N$ smaller vectors. (Here $d(n)$ corresponds to $y(n)$ or $\hat{y}_n(n)$, depending on whether the equation error or the output error approach is employed.)

$$\begin{aligned}
&\text{CH 1: } [x(n), x(n-1), \cdots, x(n-N+1)]\\
&\text{CH 2: } [d(n-1), d(n-2), \cdots, d(n-N+1)]\\
&\text{CH 3: } [x(n-1)d(n-1), x(n-2)d(n-2), \cdots, x(n-N+1)d(n-N+1)]\\
&\text{CH4: } [x(n)d(n-1), x(n-1)d(n-2), \cdots, x(n-N+2)d(n-N+1)]\\
&\text{CH 5: } [x(n-2)d(n-1), x(n-3)d(n-2), \cdots, x(n-N+1)d(n-N+2)]\\
&\text{CH 6: } [x(n)d(n-2), x(n-1)d(n-3), \cdots, x(n-N+3)d(n-N+1)]\\
&\cdots \qquad \cdots\\
&\text{CH } 2N-1\text{: } [x(n-N+1)d(n-1)]\\
&\text{CH } 2N\text{: } [x(n)d(n-N+1)].
\end{aligned} \quad (2.11)$$

Now, we can consider each of the above vectors as being formed from successive samples of signals from a different input channel. Thus we can translate the bilinear filtering problem into an equivalent multichannel linear filtering problem with $2N$ channels.

The most important manner in which this multichannel characterization differs from traditional multichannel, adaptive linear filters is that the number of coefficients associated with each channel varies from channel to channel. Channel 1 has $N$ coefficients associated with it while channels 2, 3, and 4 have $N-1$ coefficients. The number of coefficients for the $(2k-1)$th and $2k$th channels is $N-k+1$ for $k \geq 3$.

To simplify the notations, let $x_i(n)$; $i = 0, 1, \cdots, N-1$ be defined as

$$\begin{aligned} x_0(n) &= [x(n)] \\ x_1(n) &= [d(n-1), x(n-1)d(n-1), \\ &\quad \cdot x(n)d(n-1)]^T \\ x_2(n) &= [x(n-2)d(n-1), x(n)d(n-2)]^T \\ &\cdots \\ x_{N-1}(n) &= [x(n-N+1)d(n-1), x(n) \\ &\quad \cdot d(n-N+1)]^T. \end{aligned} \tag{2.12}$$

The above representation is useful since all the channels belonging to each subset have the same number of delays. In particular, the channels in $x_i(n)$ have $N-1-i$ delays associated with them. We can now express the input vector in terms of the multichannel representation as

$$\begin{aligned} X_N(n) = [x_0^T(n), \cdots, x_0^T(n-N+1), x_1^T(n), \cdots, \\ x_1^T(n-N+2), \cdots, x_{N-1}^T(n)]^T. \end{aligned} \tag{2.13}$$

All the elements of the input vector $X_N(n)$ are the same as those of $U_N(n)$. The two vectors differ from each other only in the sense that their elements are ordered in different ways, i.e.,

$$X_N(n) = L_N U_N(n) \tag{2.14}$$

where $L_N$ is an appropriate permutation matrix. Let

$$\begin{aligned} C_N(n) = [c_{0,0}(n), \cdots, c_{0,N-1}(n), \\ \cdots, c_{1,N-2}(n), \cdots, c_{N-1,0}(n)]^T \end{aligned} \tag{2.15}$$

denote the coefficient vector of the adaptive filter at time $n$ in the present multichannel formulation. Then, the output signal $\hat{y}_n(n)$ can be written as

$$\hat{y}_n(n) = C_N^T(n) X_N(n). \tag{2.16}$$

Once again, note that

$$C_N(n) = L_N W_N(n). \tag{2.17}$$

The first task involved in developing a lattice structure for bilinear filters is to obtain a Gram–Schmidt orthogonalization of the input data. The orthogonalization can be carried out in an appropriately defined inner product space. In our case, this space will be defined by the input signal corresponding to each coefficient, and the exponentially weighted least squares optimization criterion that we are employing throughout the paper. While there are several approaches for performing the Gram–Schmidt orthogonalization, our method follows the technique in [29] very closely. The development of the lattice structure depends critically on two different partitions of the input vector $X_N(n)$. The first partition divides the elements of $X_N(n)$ into $N$ "backward" input vectors as follows:

$$\begin{aligned} x_0^b(n) &= [x_0^T(n)] \\ x_1^b(n) &= [x_0^T(n-1), x_1^T(n)]^T \\ x_2^b(n) &= [x_0^T(n-2), x_1^T(n-1), x_2^T(n)]^T \\ &\cdots \\ x_{N-1}^b(n) &= [x_0^T(n-N+1), x_1^T(n-N+2), \\ &\quad \cdots, x_{N-1}^T(n)]^T. \end{aligned} \tag{2.18}$$

Let $b_m(n)$ be the optimal estimation error vector (in the least squares sense) when $x_m^b(n)$ is estimated using $x_0^b(n), x_1^b(n), \cdots, x_{m-1}^b(n)$ (Note that $b_0(n) = x_0^b(n)$ by definition). It is well known that the "backward prediction" residuals $\{b_m(n); m = 0, 1, \cdots, N-1\}$ form an orthogonal decomposition of $x_i^b(n)$; $i = 0, 1, \cdots, m-1$. Once this decomposition has been achieved, we can estimate $y(n)$ as a linear combination of the elements of the "backward prediction" error vectors.

Just as in any lattice filter formulation, efficient computation of "backward prediction" residual vectors requires knowledge of the "forward prediction" error vectors. For understanding the notion of "forward prediction" in our context, we have to introduce the second type of partitioning of the input data. For defining the $m$th order "forward prediction" error, we partition the elements of the first $m+1$ "backward" input vectors in (2.18) (which is the same as the elements of $X_m(n)$) as

$$\begin{aligned} x_{m:0}^f(n) &= [x_0^T(n-m)] \\ x_{m:1}^f(n) &= [x_0^T(n-m+1), x_1^T(n-m+1)]^T \\ x_{m:2}^f(n) &= [x_0^T(n-m+2), x_1^T(n-m+2), \\ &\quad x_2^T(n-m+2)]^T \\ &\cdots \\ x_{m:m}^f(n) &= [x_0^T(n), x_1^T(n), \cdots, x_m^T(n)]^T. \end{aligned} \tag{2.19}$$

The $m$th order "forward prediction" error vector $f_m(n)$ is defined as the estimation error when $x_{m:m}^f(n)$ is estimated as a linear combination of all the elements of $x_{m:0}^f(n), x_{m:1}^f(n), \cdots, x_{m:m-1}^f(n)$. As usual we will define $f_0(n)$ to be the same as $x_{0:0}(n)$.

We can now proceed to develop the order update equations for the "forward" and "backward" predictors of the lattice. Notice that

$$x_{m:m}^f(n) = \begin{bmatrix} x_{m-1:m-1}^f(n) \\ x_m(n) \end{bmatrix} \tag{2.20}$$

and

$$x_m^b(n) = \begin{bmatrix} x_{m-1}^b(n-1) \\ x_m(n) \end{bmatrix}. \tag{2.21}$$

Similar to the partitioning in (2.20) and (2.21), let

$$f_m(n) = \begin{bmatrix} \bar{f}_m(n) \\ f_m^{(m)}(n) \end{bmatrix} \tag{2.22}$$

and

$$b_m(n) = \begin{bmatrix} \bar{b}_m(n) \\ b_m^{(m)}(n) \end{bmatrix} \tag{2.23}$$

where $\bar{f}_m(n)$ and $\bar{b}_m(n)$ correspond to the prediction error vectors in the estimation of $x^f_{m-1:m-1}(n)$ and $x^b_{m-1}(n-1)$, respectively. Note that $b_i(n-1)$; $i = 0, 1, \cdots, m-1$ form an orthogonal decomposition for $x^b_i(n-1)$; $i = 0, 1, \cdots, m-1$ (or, equivalently, the elements of the input vector $X_m(n-1)$). Furthermore, $f_{m-1}(n)$ is the optimal "forward prediction" error when $x^f_{m-1:m-1}(n)$ is estimated using $b_i(n-1)$; $i = 0, 1, \cdots, m-2$ (which is an orthogonal decomposition of the elements of $X_{m-1}(n-1)$). It follows immediately that any possible improvement in the forward prediction of $x^f_{m-1:m-1}(n)$ using $X_m(n)$ can be obtained by making use of the additional information contained in $b_{m-1}(n-1)$. Thus the $m$th order "forward prediction" error can be obtained recursively as

$$\bar{f}_m(n) = f_{m-1}(n) - K^{fT}_m(n) b_{m-1}(n-1) \quad (2.24)$$

where $K^f_m(n)$ is the optimal coefficient matrix that estimates $x^f_{m-1:m-1}(n)$ using $b_{m-1}(n-1)$. For $m > 2$, $x^f_{m-1:m-1}(n)$ and $b_{m-1}(n-1)$ have $2(m-1)$ elements each and, therefore, $K^f_m(n)$ is a matrix with $2(m-1) \times 2(m-1)$ elements whenever $m > 2$. For the first stage, the corresponding coefficients are scalars. Let $\hat{x}^f_{m-1:m-1}(n)$ denote the optimal prediction of $x^f_{m-1:m-1}(n)$ using the elements of $b_i(n-1)$; $i = 0, 1, \cdots, m-2$. Then,

$$x^f_{m-1:m-1}(n) = \hat{x}^f_{m-1:m-1}(n) + f_{m-1}(n). \quad (2.25)$$

Since $b_m(n-1)$ is orthogonal to $\hat{x}^f_{m-1:m-1}(n)$, $K^f_m(n)$ is also the optimal coefficient matrix that estimates $f_{m-1}(n)$ using $b_{m-1}(n-1)$. Consequently,

$$K^f_m(n) = R^{-b}_{m-1}(n-1)\Delta_m(n) \quad (2.26)$$

where $R^b_{m-1}(n)$ is the least squares autocorrelation matrix of $b_{m-1}(n)$ and $\Delta_m(n)$ is the corresponding crosscorrelation matrix for $f_{m-1}(n)$ and $b_{m-1}(n-1)$. These quantities as well as methods for efficiently updating them recursively are discussed in Section III.

Similar to the above development, we can also show that

$$\bar{b}_m(n) = b_{m-1}(n-1) - K^{bT}_m(n) f_{m-1}(n) \quad (2.27)$$

where $K^b_m(n)$ is the appropriate predictor coefficient matrix. In analogy with usual lattice terminology, we will refer to the coefficient matrices $K^f_m(n)$ and $K^b_m(n)$ as reflection coefficient matrices.

It is also not very difficult to develop an order recursion for $f^{(m)}_m(n)$ and $b^{(m)}_m(n)$ as follows:

$$f^{(m)}_m(n) = f^{(m)}_{m-1}(n) - K^{f(m)T}_m(n) b_{m-1}(n-1) \quad (2.28)$$

$$b^{(m)}_m(n) = f^{(m)}_{m-1}(n) - K^{b(m)T}_m(n) f_{m-1}(n) \quad (2.29)$$

where $f^{(m)}_k(n)$; $k = 0, 1, \cdots, m-1$ are auxiliary prediction error vectors in estimating $x_m(n)$ using the vectors $x^f_{k:0}(n), \cdots, x^f_{k:k-1}(n)$, and $K^{f(m)}_m(n)$ and $K^{b(m)}_m(n)$ are the auxiliary reflection coefficient matrices at $m$th stage. Similar to the discussion surrounding the reflection coefficients, we can see that the auxiliary reflection coefficients are matrices with $2 \times 2m$ elements at the $m$th stage for $m \geq 2$. $f^{(m)}_k(n)$ can be obtained recursively as

$$f^{(m)}_k(n) = f^{(m)}_{k-1}(n) - K^{f(m)T}_k b_{k-1}(n-1);$$
$$k = 1, 2, \cdots, m-1 \quad (2.30)$$

with $f^{(m)}_0(n) = x_m(n)$.

The joint process estimation error $e_N(n)$ is the error in estimating $y(n)$ using the input vector $X_N(n)$ and is given by

$$e_N(n) = y(n) - C^T_N(n) X_N(n). \quad (2.31)$$

As discussed earlier, since the elements of the "backward prediction" error vectors $b_0(n), \cdots, b_{N-1}(n)$ span the same space that is spanned by the elements of input vector $X_N(n)$ and since $b_m(n)$ is orthogonal to the space spanned by $b_0(n), \cdots, b_{m-1}(n)$, we can compute the joint process estimation error in an order recursive manner as

$$e_m(n) = y(n) - \sum_{k=1}^{m} k^{yT}_k(n) b_{k-1}(n)$$
$$= e_{m-1}(n) - k^{yT}_m(n) b_{m-1}(n) \quad (2.32)$$

where $k^y_m(n)$ is the appropriate coefficient vector and $e_0(n) = y(n)$. Note that $k^y_m(n)$ has $2m$ elements for $m \geq 2$ and that it is a scalar when $m = 1$.

The lattice equations corresponding to the structure developed are given in Table I, and the structure is depicted in Fig. 1. The procedure for converting from lattice parameter to direct-form bilinear filter parameters is derived in the Appendix.

*Remark:* The extension of the lattice structure to general recursive polynomial systems described in (1.2) is a straightforward task. In fact, the only difference between the general case and the bilinear case is how the input vectors $x_0(n), x_1(n), \cdots, x_{N-1}(n)$ are defined. Suppose that we can partition all the terms in the polynomial system model as belonging to $M$ channels. Let $N_m$ be the delay associated with the $m$th channel. Note that $N_m$ belongs to the set $0, 1, \cdots, N-1$ since we have restricted the number of delay elements in (1.2). We will now define $x_i(n)$ as the vector formed by the most recent elements of all the channels with $N-1-i$ delay elements. Once these vectors have been defined, the rest of the derivations will apply to general recursive polynomial models also. Note that the above discussion will apply to bilinear system models where the number of coefficients associated with each summation in the definition of the system model is not restricted to be the same as in (1.3). In the rest of the paper, we will concentrate on the model in (1.3) because of the simplicity of presentation it provides.

## III. Least Squares Lattice Adaptive Bilinear Filtering Algorithm

Once the lattice structure has been developed, deriving the adaptation algorithm is fairly simple. The development is closely related to that in [29]. The key idea is that

TABLE I
LATTICE BILINEAR FILTER

*Initialization*

$$f_0(n) = b_0(n) = x(n) \tag{T-1.1}$$

$$f_0^{(p)}(n) = \begin{cases} [d(n-1), x(n-1)d(n-1), x(n)d(n-1)]^{T\dagger}, & p = 1 \\ [x(n-p)d(n-1), x(n)d(n-p)]^{T\dagger}, & p = 2, 3, \cdots, N-1 \end{cases} \tag{T-1.2}$$

$$e_0(n) = y(n). \tag{T-1.3}$$

*Iteration procedure*

Do (T-1.4)–(T-1.7), for $m = 1, 2, \cdots, N-1$

$$f_m(n) = \begin{bmatrix} f_{m-1}(n) - K_m^{fT}(n) b_{m-1}(n-1) \\ f_{m-1}^{(m)}(n) - K_m^{f(m)T}(n) b_{m-1}(n-1) \end{bmatrix} \tag{T-1.4}$$

$$b_m(n) = \begin{bmatrix} b_{m-1}(n-1) - K_m^{bT}(n) f_{m-1}(n) \\ f_{m-1}^{(m)}(n) - K_m^{b(m)T}(n) f_{m-1}(n) \end{bmatrix} \tag{T-1.5}$$

Do (T-1.6), for $p = m+1, m+2, \cdots, N-1$

$$f_m^{(p)}(n) = f_{m-1}^{(p)}(n) - K_m^{f(p)T}(n) b_{m-1}(n-1) \tag{T-1.6}$$

$$e_m(n) = e_{m-1}(n) - k_m^{yT}(n) b_{m-1}(n) \tag{T-1.7}$$

$$e_N(n) = e_{N-1}(n) - k_N^{yT}(n) b_{N-1}(n). \tag{T-1.8}$$

†For equation error formulation, $d(n) = y(n)$.
For output error formulation, $d(n) = y(n) - e_N(n)$.

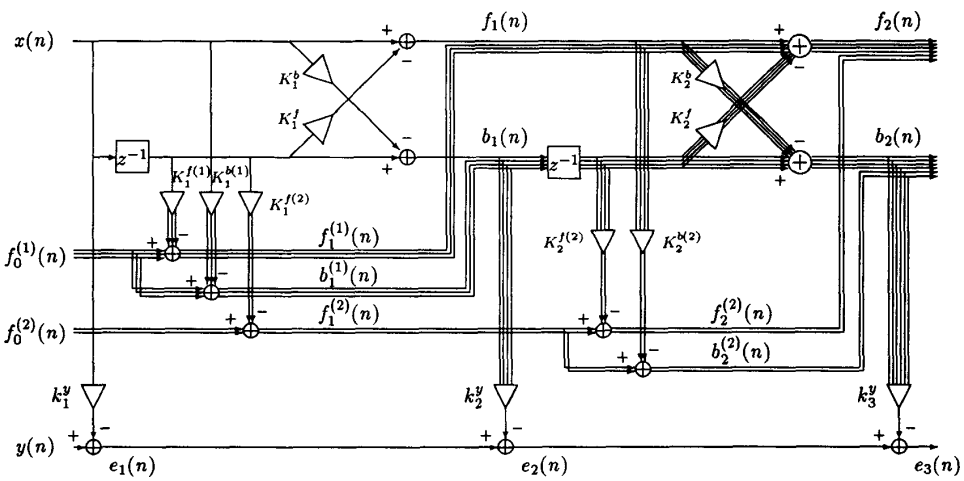



Fig. 1. Block diagram of the bilinear lattice structure for $N = 3$. The number of lines going into and out of any component represents the number of input channels and output channels, respectively, of that component.

the orthogonalization of the data makes it possible to consider each stage separately and the recursions defining the adaptive filter can be developed independently for each stage. Let us consider the process for updating $K_m^f(n)$. The derivation that follows is typical of every other case, and only sketches of the derivation will be given for the other variables. The complete set of recursions for the least squares lattice adaptive bilinear filter is given in Table II.

As discussed in Section II, $K_m^f(n)$ is the optimal coefficient matrix for estimating $f_{m-1}(n)$ as a linear combination of the elements of $b_{m-1}(n-1)$ and is given by

$$K_m^f(n) = R_{m-1}^{-b}(n-1)\Delta_m(n) \tag{3.1}$$

where

$$R_{m-1}^b(n) = \sum_{k=1}^{n} \lambda^{n-k} b_{m-1,n}(k) b_{m-1,n}^T(k) \tag{3.2}$$

and

$$\Delta_m(n) = \sum_{k=1}^{n} \lambda^{n-k} b_{m-1,n}(k-1) f_{m-1,n}^T(k) \tag{3.3}$$

are the least squares autocorrelation matrix of $b_{m-1,n}(n)$ and the least squares cross-correlation matrix of $b_{m-1,n}(n-1)$ and $f_{m-1,n}(n)$, respectively. In the above equations the second subscript $n$ for the prediction error sequences explicitly indicates that the prediction errors are com-

TABLE II
THE RECURSIVE LEAST SQUARES ADAPTIVE LATTICE BILINEAR FILTER

*Time initialization*

Do (T-2.1)-(T-2.2), for $m = 0, 1, \cdots, N-1$

$$\alpha_m(0) = 1 \tag{T-2.1}$$

$$R^f_m(0) = R^b_m(0) = \begin{cases} \delta, & \text{if } m = 0 \\ \delta I_{2m+2}, & \text{otherwise} \end{cases} \tag{T-2.2}$$

Do (T-2.3)-(T-2.21), for $n = 1, 2, \cdots$.

*Order initialization*

Do (T-2.3)-(T-2.7), for $n = 1, 2, \cdots$

$$\alpha_0(n) = 1 \tag{T-2.3}$$

$$R^f_0(n) = R^b_0(n) = \lambda R^f_0(n-1) + x(n)^2 \tag{T-2.4}$$

$$f_0(n) = b_0(n) = x(n) \tag{T-2.5}$$

$$f^{(p)}_0(n) = \begin{cases} [d(n-1), x(n-1)d(n-1), x(n)d(n-1)]^{T\dagger}, & p = 1 \\ [x(n-p)d(n-1), x(n)d(n-p)]^{T\dagger}, & p = 2, 3, \cdots, N-1 \end{cases} \tag{T-2.6}$$

$$e_0(n) = y(n). \tag{T-2.7}$$

*Iteration procedure*

Do (T-2.8)-(T-2.19) for $m = 1, 2, \cdots, N-1$

$$\Delta_m(n) = \lambda\Delta_m(n-1) + b_{m-1}(n-1)f^T_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T-2.8}$$

$$\Delta^{f(m)}_m(n) = \lambda\Delta^{f(m)}_m(n-1) + b_{m-1}(n-1)f^{(m)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T-2.9}$$

$$\Delta^{b(m)}_m(n) = \lambda\Delta^{b(m)}_m(n-1) + f_{m-1}(n)f^{(m)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T-2.10}$$

$$f_m(n) = \begin{bmatrix} f_{m-1}(n) - \Delta^T_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \\ f^{(m)}_{m-1}(n) - \Delta^{f(m)T}_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \end{bmatrix} \tag{T-2.11}$$

$$b_m(n) = \begin{bmatrix} b_{m-1}(n-1) - \Delta_m(n)R^{-f}_{m-1}(n)f_{m-1}(n) \\ f^{(m)}_{m-1}(n) - \Delta^{b(m)T}_m(n)R^{-f}_{m-1}(n)f_{m-1}(n) \end{bmatrix} \tag{T-2.12}$$

Do (T-2.13)-(T-2.14), for $p = m+1, m+2, \cdots, N-1$

$$\Delta^{f(p)}_m(n) = \lambda\Delta^{f(p)}_m(n-1) + b_{m-1}(n-1)f^{(p)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T-2.13}$$

$$f^{(p)}_m(n) = f^{(p)}_{m-1}(n) - \Delta^{f(p)T}_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \tag{T-2.14}$$

$$\Delta^{\gamma}_m(n) = \lambda\Delta^{\gamma}_m(n-1) + b_{m-1}(n)e_{m-1}(n)/\alpha_{m-1}(n) \tag{T-2.15}$$

$$e_m(n) = e_{m-1}(n) - \Delta^{\gamma T}_m(n)R^{-b}_{m-1}(n)b_{m-1}(n) \tag{T-2.16}$$

$$\alpha_m(n) = \alpha_{m-1}(n) - b^T_{m-1}(n)R^{-b}_{m-1}(n)b_{m-1}(n) \tag{T-2.17}$$

$$R^{-f}_m(n) = \lambda^{-1}R^{-f}_m(n-1) - \frac{\lambda^{-2}R^{-f}_m(n-1)f_m(n)f^T_m(n)R^{-f}_m(n-1)}{\alpha_m(n-1) + \lambda^{-1}f^T_m(n)R^{-f}_m(n-1)f_m(n)} \tag{T-2.18}$$

$$R^{-b}_m(n) = \lambda^{-1}R^{-b}_m(n-1) - \frac{\lambda^{-2}R^{-b}_m(n-1)b_m(n)b^T_m(n)R^{-b}_m(n-1)}{\alpha_m(n) + \lambda^{-1}b^T_m(n)R^{-b}_m(n-1)b_m(n)} \tag{T-2.19}$$

$$\Delta^{\gamma}_N(n) = \lambda\Delta^{\gamma}_N(n-1) + b_{N-1}(n)e_{N-1}(n)/\alpha_{N-1}(n) \tag{T-2.20}$$

$$e_N(n) = e_{N-1}(n) = \Delta^{\gamma T}_N(n)R^{-b}_{N-1}(n)b_{N-1}(n). \tag{T-2.21}$$

†For equation error formulation, $d(n) = y(n)$.
For output error formulation, $d(n) = y(n) - e_N(n)$.

puted using the optimal coefficients at time $n$. $b_m(n)$ and $f_m(n)$ defined in Section II are related to the above quantities as

$$b_m(n) = b_{m,n}(n) \tag{3.4}$$

and

$$f_m(n) = f_{m,n}(n). \tag{3.5}$$

In the rest of the paper, we will avoid the second subscript for the prediction error sequences, since this causes no confusion. It is shown in [30, sect. 9] that the cross-correlation matrices and autocorrelation matrix can be updated as

$$\Delta_m(n) = \lambda\Delta_m(n-1) + b_{m-1}(n-1)f^T_{m-1}(n)/\alpha_{m-1} \cdot (n-1) \tag{3.6}$$

$$R^b_{m-1}(n) = \lambda R^b_{m-1}(n-1) + b_{m-1}(n)b^T_{m-1} \cdot (n)/\alpha_{m-1}(n) \tag{3.7}$$

where

$$\alpha_m(n) = \alpha_{m-1}(n) = b^T_{m-1}(n)R^{-b}_{m-1}(n)b_{m-1}(n). \tag{3.8}$$

$\alpha_m(n)$, of course, is the familiar "likelihood" variable for

TABLE III
COMPUTATIONAL COMPLEXITY OF THE ADAPTIVE LATTICE BILINEAR FILTERS

| Equation Number | Number of Multiplication and Division | Equation Number | Number of Multiplication and Division |
|---|---|---|---|
| (T-2.4) | 2 | (T-2.14)†† | $n_m l_p$ |
| (T-2.6) | $2(N-1)$ | (T-2.15) | $2n_m + 1$ |
| (T-2.8) | $2n_m^2 + n_m$ | ‡ | $n_m^2$ |
| † | $l_m$ | (T-2.16) | $n_m$ |
| (T-2.9) | $2n_m l_m$ | (T-2.17) | $n_m$ |
| (T-2.10) | $2n_m l_m$ | (T-2.18)‡‡ | $n_m^2 + n_m l_m$ |
| (T-2.11)†† | $n_m^2 + n_m l_m$ | (T-2.19) | $2n_{m+1}^2 + 3n_{m+1} + 2$ |
| (T-2.12) | $2n_m^2 + n_m l_m$ | (T-2.20) | $2n_N + 1$ |
| (T-2.13) | $2n_m l_p + l_p$ | (T-2.21) | $n_N^2 + n_N$ |
| Total | $\sum_{m=1}^{N-1} (4n_{m+1}^2 + 6n_{m+1} + 6n_m l_m + 6n_m^2 + 5n_m + l_m + 5)$ $+ \sum_{m=1}^{N-1} \sum_{p=m+1}^{N-1} (3n_m l_p + l_p) - n_N^2 + 2N - 1$ $= 46/3N^3 + 10N^2 - 1/3N - 50; \quad N \geq 2$ | | |

$$n_m = \begin{cases} 1, & m = 1 \\ 2m, & m = 2, 3, \cdots, N-1 \end{cases}$$

$$l_p = \begin{cases} 3, & p = 1 \\ 2, & p = 2, 3, \cdots, N-1 \end{cases}$$

†Calculation of $f_{m-1}^{(m)T}(n)/\alpha_{m-1}(n-1)$ in (T-2.9) and (T-2.10).
‡Calculation of $R_{m-1}^{-b}(n)b_{m-1}(n)$.
††$R_{m-1}^{-b}(n-1)b_{m-1}(n-1)$ in (T-2.11) and (T-2.14) is just a delayed version of $R_{m-1}^{-b}(n)b_{m-1}(n)$.
‡‡(T-2.18) need not be computed for $m = N-1$.

the $m$th stage and it is relatively easy to show that [30]

$$0 \leq \alpha_m(n) \leq 1. \tag{3.9}$$

The inverse of the autocorrelation matrix can be recursively updated using the matrix inversion lemma as [28]

$$R_m^{-b}(n) = \lambda^{-1} R_m^{-b}(n-1) - \frac{\lambda^{-2} R_m^{-b}(n-1) b_m(n) b_m^T(n) R_m^{-b}(n-1)}{\alpha_m(n) + \lambda^{-1} b_m^T(n) R_m^{-b}(n-1) b_m(n)}. \tag{3.10}$$

The recursions for $K_m^b(n)$ can be derived in a similar fashion by simply recognizing that $K_m^b(n)$ is the optimal (in the least squares sense) coefficient matrix for estimating $b_{m-1}(n-1)$ as a linear combination of elements of $f_{m-1}(n)$. The relevant equations appear in Table II as (T-2.8), the top part of (T-2.12), and (T-2.17). The auxiliary variables $K_m^{f(p)}(n)$ and $K_m^{b(m)}(n)$ can also be similarly updated. Note that $K_m^{f(p)}(n)$ turns out to be the optimal coefficient matrix when $f_{m-1}^{(p)}(n)$ is estimated using $b_{m-1}(n-1)$ and that $K_m^{b(m)}(n)$is the optimal coefficient matrix that estimates $f_{m-1}^{(m)}(n)$ using elements of $f_{m-1}(n)$.

The derivations of the recursions for joint-process estimation are also similar to earlier derivations. Note that the orthogonality of $b_m(n)$; $m = 0, 1, \cdots, N-1$ implies that estimation of $y(n)$ using $b_m(n)$ will give the same result as estimating $e_{m-1}(n)$ (the error in estimating $y(n)$ using $b_0(n), b_1(n), \cdots, b_{m-1}(n)$). It is now straightforward to see that (T-2.16)–(T-2.19), (T-2.20), and (T-2.21) in Table II describe the set of recursions for updating the joint-process estimator coefficient vectors and the joint-process estimation error sequence.

The number of multiplications and divisions associated with each recursion is tabulated in Table III. The count of other arithmetical operations is comparable to that shown for multiplications and divisions. The computational complexity corresponds to $46/3N^3 + 10N^2 - 1/3N - 50$ multiplications and divisions per iteration for $N \geq 2$. This complexity is substantially smaller than the $O(N^4)$ complexity algorithms that were previously available. This statement is especially true for large values of $N$.

The initial conditions for this algorithm are the same as those for conventional recursive least squares lattice algorithms [30] and are given by

$$\Delta_m(0) = \Delta_m^{f(p)}(0) = \Delta_m^{b(m)}(0) = \Delta_m^{y}(0) = 0 \tag{3.11}$$

$$R_m^f(0) = R_m^b(0) = \begin{cases} \delta, & m = 0 \\ \delta I_{2m+2}, & m = 1, 2, \cdots, N-1 \end{cases} \tag{3.12}$$

$$\alpha_m(0) = 1 \tag{3.13}$$

where $\delta$ is a small positive constant and $I_l$ is a $l \times l$ identity matrix. Exact initialization is possible; however, we did not incorporate exact initialization techniques in our experiments.

## IV. Experimental Results

Several experiments for identifying an unknown bilinear system from measurement of its input and (a noisy version of) the corresponding output were conducted to evaluate the performance of the two algorithms. Performance comparisons were also made with several algorithms with $O(N^4)$ computational complexity. The input signal $x(n)$ to the adaptive filter was obtained as the output of a low-pass filter with transfer function

$$H(z) = \frac{1}{1 - 1.6z^{-1} + 0.95z^{-2}} \tag{4.1}$$

when its input was a white, zero-mean and pseudorandom Gaussian noise. The input signal variance was adjusted such that the filter output had unit variance. The desired response signal $y(n)$ was obtained as a noisy measurement of a bilinear system with coefficients as given in Table IV. The measurement noise signal was an additive white, zero-mean and pseudorandom Gaussian process and was uncorrelated with the input signal. The adaptive filter was run with the same number of coefficients as the unknown system. All of the results presented are ensemble averages over 50 independent runs.

Figs. 2 and 3 display the mean-squared estimation error associated with the equation error and output error methods, respectively, when $\lambda = 0.995$. Three different noise levels corresponding to output signal-to-noise ratios (SNR's) of $\infty$, 30, and 20 dB were considered in the experiments. Note that both algorithms have very good convergence speeds. The time averages of the ensemble-averaged mean-squared error during the interval from $n = 4001$ to $n = 5000$ are given in Table V.

The tabulations in Table V indicate that the output error algorithm performs better than the equation error algorithm in the presence of measurement noise. The difference in performance is evident in a more dramatic form when the mean coefficient trajectories are compared for different noise levels. The mean trajectories of $a_1(n)$, $c_1(n)$, and $b_{2,2}(n)$ are plotted in Figs. 4–6, respectively, for different noise levels and $\lambda = 0.995$. The ensemble averages were obtained by averaging the coefficients of the direct-form bilinear filter realizations after converting the lattice parameters to direct-form parameters using the results in Table VII. Table VI contains time averages of the coefficient trajectories over the last 1000 samples. We can observe that the coefficients converge to values different from those of the unknown system in the equation error method when the output measurements are noisy. The mean behavior of the coefficients in the output error adaptive filter appears to be much less sensitive to measurement noise. Note also that the coefficients corresponding to the nonlinear terms are more sensitive to noise in both algorithms.

We conducted several experiments to compare the performance of our algorithm with those of two of the computationally more expensive, direct-form techniques discussed in [23]. Figs. 7 and 8 display the mean behavior of $c_1(n)$ obtained using the extended least squares (ELS) algorithm and the recursive prediction-error method (RPEM), respectively. Similar to some of the results in [23], the behavior of the coefficient obtained using the

TABLE IV
Coefficients of the Unknown Bilinear System Used in the Experiments

| | | |
|---|---|---|
| $a_0 = 1.0$ | $a_1 = 1.0$ | $a_2 = 1.0$ |
| $b_{0,1} = 0.3$ | $b_{1,1} = -0.2$ | $b_{2,1} = 0.1$ |
| $b_{0,2} = 0.1$ | $b_{1,2} = -0.2$ | $b_{2,2} = 0.3$ |
| | $c_1 = 0.5$ | $c_2 = -0.5$ |

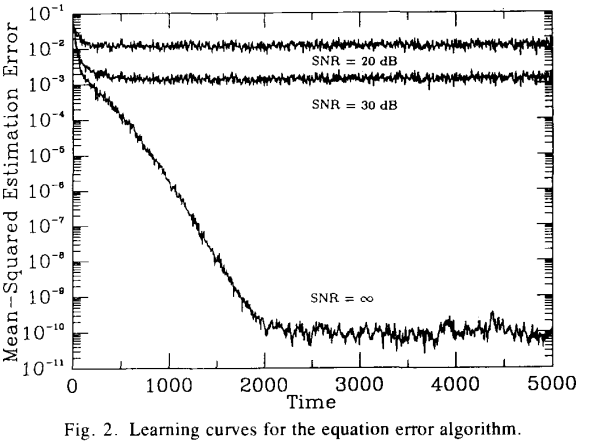



Fig. 2. Learning curves for the equation error algorithm.

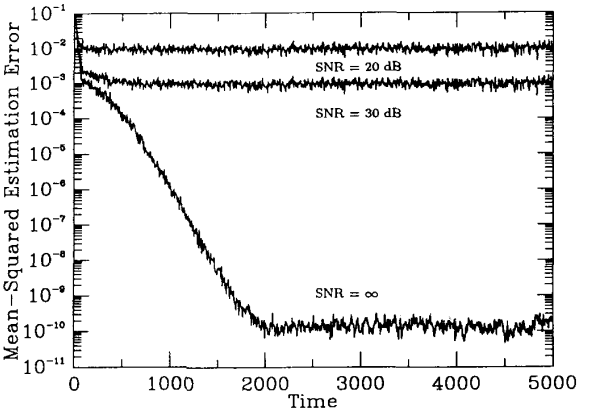



Fig. 3. Learning curves for the output error algorithm.

TABLE V
Time-Averaged Mean Squared Error Over the Last 1000 Data Samples

| SNR \ λ | Equation Error | | Output Error | |
|---|---|---|---|---|
| | 0.995 | 0.99 | 0.995 | 0.99 |
| ∞ | 0.104*E*-9 | 0.577*E*-10 | 0.137*E*-9 | 0.714*E*-10 |
| 30 dB | 0.136*E*-2 | 0.127*E*-2 | 0.942*E*-3 | 0.887*E*-3 |
| 20 dB | 0.118*E*-1 | 0.109*E*-1 | 0.941*E*-2 | 0.886*E*-2 |

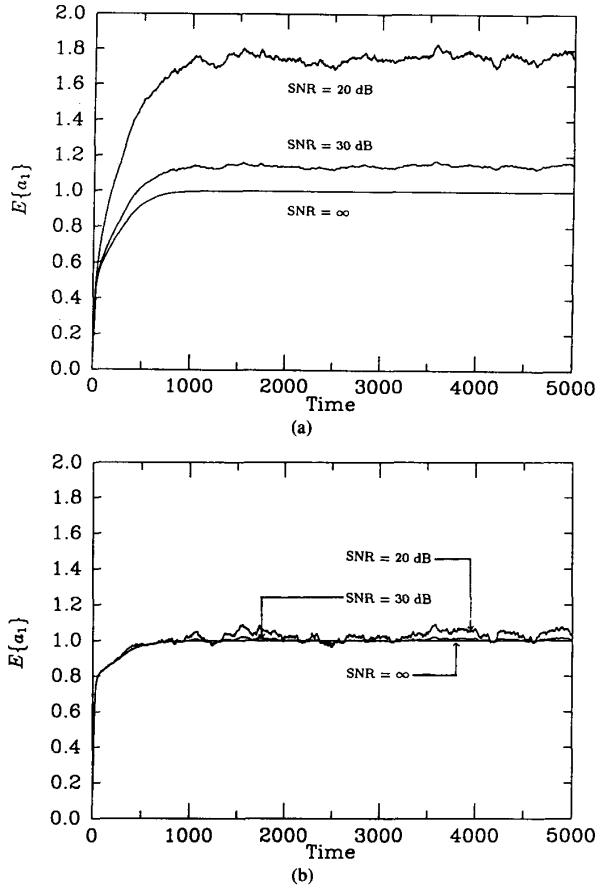



Fig. 4. Mean trajectories of coefficient $a_1(n)$ for different output SNR's: (a) Equation error algorithm. (b) Output error algorithm.

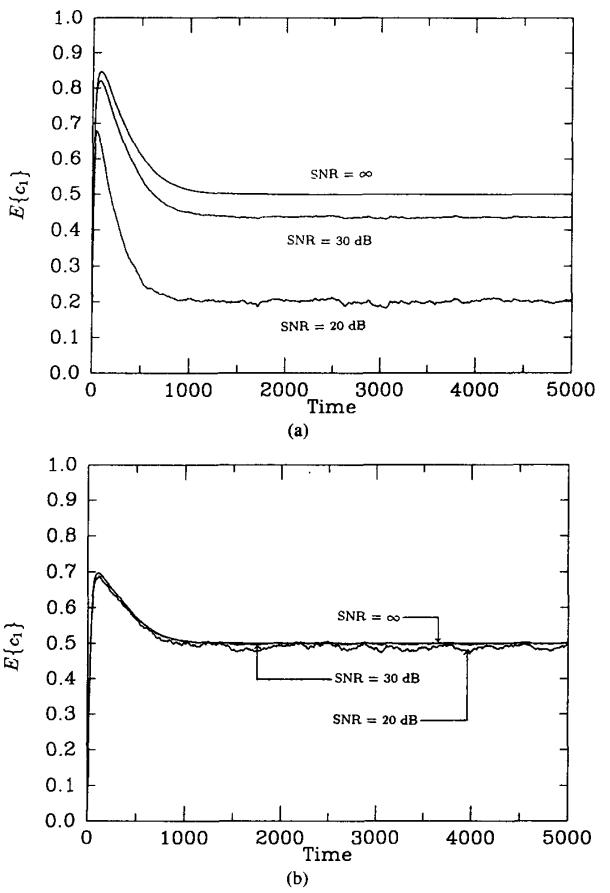



Fig. 5. Mean trajectories of coefficient $c_1(n)$ for different output SNR's: (a) Equation error algorithm. (b) Output error algorithm.

ELS algorithm is very erratic. This may have been caused by the poor numerical properties of the ELS algorithm. RPEM performs the best among all the $O(N^4)$ complexity algorithms discussed in [23]. Its performance seems to be slightly less noisy than the output error lattice bilinear filter of this paper. It is interesting to note that the output error version of our algorithm performs almost as well as the RPEM even though it has only $O(N^3)$ computational complexity.

## V. Concluding Remarks

This paper presented an adaptive least squares lattice structure for bilinear filtering. Two closely related algorithms of $O(N^3)$ complexity were introduced. This complexity is an order of magnitude better than algorithms of $O(N^4)$ complexity that were previously available. It is possible to develop $O(N^3)$ algorithms for direct-form realizations of adaptive bilinear filters by making use of the ideas in [13]. The advantage of the lattice filters that we presented over such algorithms is that they are numerically stable, whereas the "fast," direct-form algorithms will almost certainly have poor numerical characteristics. We believe that the computational efficiency and good numerical properties make the introduction of our algorithms a significant development in the area of adaptive bilinear filtering.

The equation error algorithm has the advantage that the adaptive algorithm will converge to a unique minimum. However, this minimum will not correspond to the coefficients of the unknown system in the presence of measurement noise in applications involving system identification. The effect of measurement noise on the performance of the equation error algorithm was clearly evident in the experimental results presented in Section IV. The output error algorithm was designed to reduce the effect of output measurement noise on the performance of the adaptive filter. Results of our experiments indicate that the output error algorithm is far less sensitive to measurement noise than the equation error technique. Even though we have not attempted a theoretical performance analysis of the output error algorithm, we believe that the results in [31] can be directly applied to our case.

One significant problem that we have not addressed in this paper is that of the stability of the identified bilinear

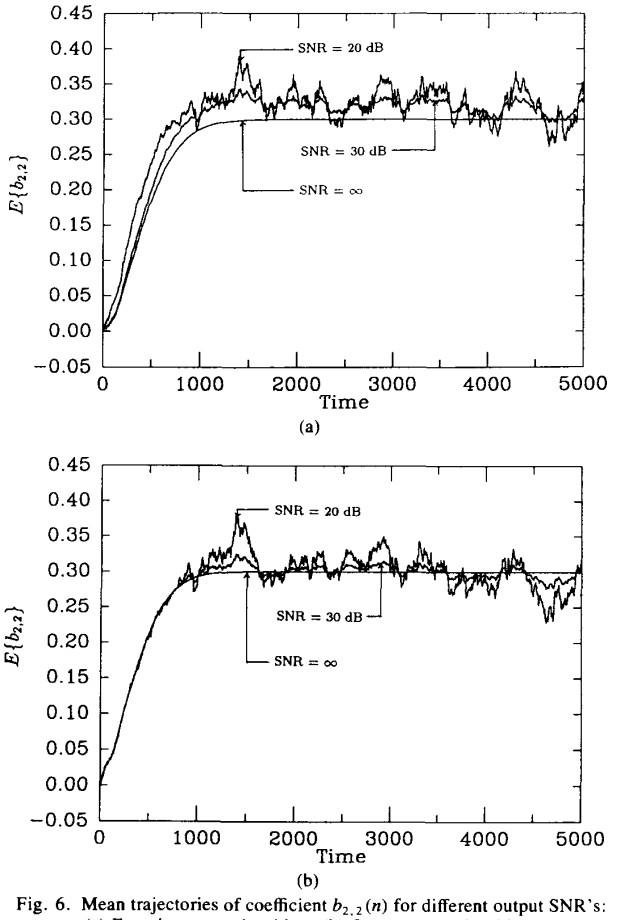



Fig. 6. Mean trajectories of coefficient $b_{2,2}(n)$ for different output SNR's: (a) Equation error algorithm. (b) Output error algorithm.

system. In general, it is impossible to derive stability conditions for bilinear systems that are independent of the characteristics of the input signal. Since one can rewrite the input–output relationship of bilinear systems as

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{j=1}^{N-1} \left[ c_j + \sum_{i=0}^{N-1} b_{ij} x(n-i) \right] y(n-j) \quad (5.1)$$

one may almost always be able to find a bounded input sequence that can drive a given bilinear system to instability. There are several results in literature available for (mean square) stability of bilinear systems for given input signal statistics [32]–[34]. While these results or their variations can be applied to check the stability of the bilinear systems, on-line implementation of such checks is a very costly proposition. One of the problems that the authors are currently working on involves the development of the computationally efficient stability checks that can be implemented on-line and incorporated into our algorithm.

## Appendix
## Conversion From Lattice to Direct Form

Conversion from the lattice parameterization to direct-form parameterization is a relatively straightforward task. We believe that the derivation presented here is much simpler than the one presented in [27] for the two-channel case. Before proceeding to the derivations, we need to define a few quantities related to the direct-form realization of bilinear filters. We define permutation matrices $T_m$ and $S_m$ as

$$T_m X_m(n) = \begin{bmatrix} x^f_{m-1:m-1}(n) \\ X_{m-1}(n-1) \end{bmatrix} \quad (A.1)$$

and

$$S_m X_m(n) = \begin{bmatrix} X_{m-1}(n) \\ x^b_{m-1}(n) \end{bmatrix}. \quad (A.2)$$

Note that $T_m$ and $S_m$ are orthogonal matrices, i.e., $T_m^{-1} = T_m^T$ and $S_m^{-1} = S_m^T$.

Let $A_m(n)$ and $B_m(n)$ be the coefficient vectors corresponding to the $m$th order "forward" and "backward" predictors defined earlier. That is

$$f_m(n) = x^f_{m:m}(n) - A_m^T(n) X_m(n-1) = [I, -A_m^T(n)]\, T_{m+1} X_{m+1}(n) \quad (A.3)$$

and

$$b_m(n) = x^b_m(n) - B_m^T(n) X_m(n) = [-B_m^T(n), I]\, S_{m+1} X_{m+1}(n). \quad (A.4)$$

Let us further partition the coefficient matrices as

$$A_m(n) = [\bar{A}_m(n), A_m^{(m)}(n)] \quad (A.5)$$

and

$$B_m(n) = [\bar{B}_m(n), B_m^{(m)}(n)] \quad (A.6)$$

so that

$$\bar{f}_m(n) = x^f_{m-1:m-1}(n) - \bar{A}_m^T(n) X_m(n-1) \quad (A.7)$$

$$f_m^{(m)}(n) = x_m(n) - A_m^{(m)T}(n) X_m(n-1) \quad (A.8)$$

$$\bar{b}_m(n) = x^b_{m-1}(n-1) - \bar{B}_m^T(n) X_m(n) \quad (A.9)$$

and

$$b_m^{(m)}(n) = x_m(n) - B_m^{(m)}(n) X_m(n). \quad (A.10)$$

Similarly, let us also define the direct-form coefficient matrix and vector, $A_m^{(p)}(n)$ and $C_m(n)$ for the auxiliary prediction and joint-process estimation problems, respectively, using the following definitions for the corresponding error vectors

$$f_m^{(p)}(n) = x_p(n) - A_m^{(p)T}(n) X_m(n-1); \quad p = m+1, m+2, \cdots, N-1 \quad (A.11)$$

TABLE VI
TIME-AVERAGED COEFFICIENTS OVER THE LAST 1000 DATA SAMPLES

| SNR | Equation Error | | | Output Error | | | Optimal Coefficient |
|---|---|---|---|---|---|---|---|
| | ∞ | 30 dB | 20 dB | ∞ | 30 dB | 20 dB | |
| $a_0$ | 1.0 | 0.997 | 0.993 | 1.0 | 0.998 | 0.994 | 1.0 |
| $a_1$ | 1.0 | 1.146 | 1.757 | 1.0 | 1.005 | 1.036 | 1.0 |
| $a_2$ | 1.0 | 1.003 | 0.808 | 1.0 | 0.997 | 0.990 | 1.0 |
| $b_{0,1}$ | 0.3 | 0.306 | 0.309 | 0.3 | 0.305 | 0.315 | 0.3 |
| $b_{0,2}$ | 0.1 | 0.095 | 0.091 | 0.1 | 0.096 | 0.086 | 0.1 |
| $b_{1,1}$ | −0.2 | −0.215 | −0.277 | −0.2 | −0.210 | −0.233 | −0.2 |
| $b_{1,2}$ | −0.2 | −0.171 | −0.077 | −0.2 | −0.193 | −0.176 | −0.2 |
| $b_{2,1}$ | 0.1 | 0.090 | 0.116 | 0.1 | 0.108 | 0.127 | 0.1 |
| $b_{2,2}$ | 0.3 | 0.313 | 0.311 | 0.3 | 0.294 | 0.281 | 0.3 |
| $c_1$ | 0.5 | 0.436 | 0.203 | 0.5 | 0.499 | 0.489 | 0.5 |
| $c_2$ | −0.5 | −0.471 | −0.323 | −0.5 | −0.499 | −0.494 | −0.5 |

TABLE VII
CONVERSION FROM LATTICE TO DIRECT FORM

*Initialization*

$$A_1(n) = [K_1^f(n),\ K_1^{f(1)}(n)] \tag{T-A.1}$$

$$B_1(n) = [K_1^b(n),\ K_1^{b(1)}(n)] \tag{T-A.2}$$

$$A_1^{(p)}(n) = K_1^{f(p)}(n); \quad p = 2, 3, \cdots, N-1 \tag{T-A.3}$$

$$C_1(n) = k_1^y(n) \tag{T-A.4}$$

*Iteration Procedure*

Do (T-A.5)-(T-A.8), for $m = 2, 3, \cdots, N-1$

$$A_m(n) = S_m^T \begin{bmatrix} A_{m-1}(n) - B_{m-1}(n-1)K_m^f(n), & A_{m-1}^{(m)}(n) - B_{m-1}(n-1)K_m^{f(m)}(n) \\ K_m^f(n) & K_m^{f(m)}(n) \end{bmatrix} \tag{T-A.5}$$

$$B_m(n) = T_m^T \begin{bmatrix} K_m^b(n) & K_m^{b(m)}(n) \\ B_{m-1}(n-1) - A_{m-1}(n)K_m^b(n), & A_{m-1}^{(m)}(n) - A_{m-1}(n)K_m^{b(m)}(n) \end{bmatrix} \tag{T-A.6}$$

Do (T-A.7), for $p = m+1, m+2, \cdots, N-1$

$$A_m^{(p)}(n) = S_m^T \begin{bmatrix} A_{m-1}^{(p)}(n) - B_{m-1}(n-1)K_m^{f(p)}(n) \\ K_m^{f(p)}(n) \end{bmatrix} \tag{T-A.7}$$

$$C_m(n) = S_m^T \begin{bmatrix} C_{m-1}(n) - B_{m-1}(n)k_m^y(n) \\ k_m^y(n) \end{bmatrix} \tag{T-A.8}$$

$$C_N(n) = S_N^T \begin{bmatrix} C_{N-1}(n) - B_{N-1}(n)k_N^y(n) \\ k_N^y(n) \end{bmatrix}. \tag{T-A.9}$$

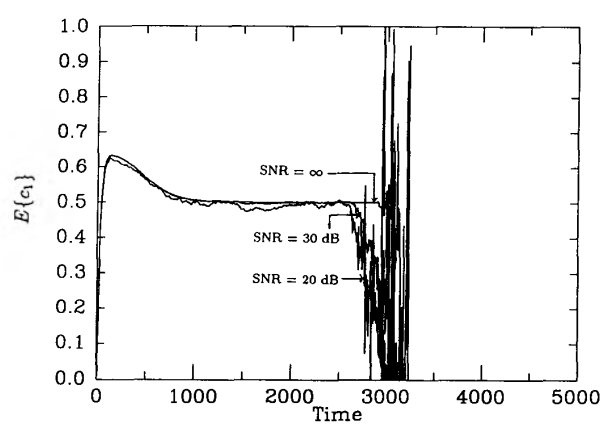



Fig. 7. Mean trajectories of coefficient $c_1(n)$ for different output SNR's for the ELS algorithm.

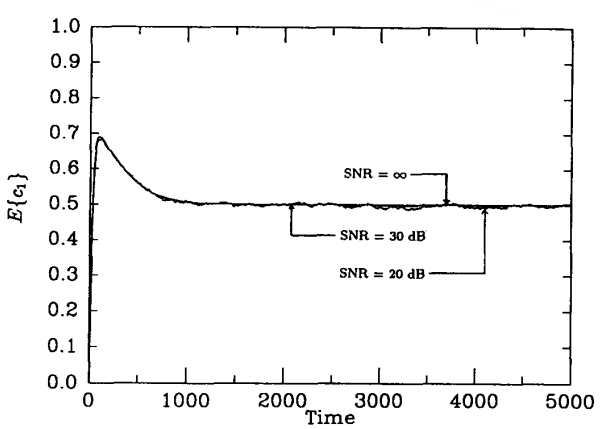



Fig. 8. Mean trajectories of coefficient $c_1(n)$ for different output SNR's for RPEM.

and

$$e_m(n) = d(n) - C_m^T(n)X_m(n). \tag{A.12}$$

Now we are ready to develop a set of recursions that relates the direct-form coefficient matrices at stage $m$ to the direct-form coefficient matrices at stage $m-1$ and the reflection coefficient matrices at stage $m$. Substituting (A.3) and (A.4) into (2.24) results in the following:

$$\begin{aligned}\bar{f}_m(n) &= f_{m-1}(n) - K_m^{fT}(n)b_{m-1}(n-1)\\ &= x^f_{m-1:m-1}(n) - A_{m-1}^T(n)X_{m-1}(n-1)\\ &= K_m^{fT}(n)[-B_{m-1}^T(n-1), I]S_mX_m(n-1)\\ &= x^f_{m-1:m-1}(n) - [A_{m-1}^T(n), \mathbf{0}]\begin{bmatrix}X_{m-1}(n-1)\\ x^b_{m-1}(n-1)\end{bmatrix}\\ &\quad -K_m^{fT}(n)[-B_{m-1}^T(n-1), I]S_mX_m(n-1)\\ &= x^f_{m-1:m-1}(n) - \{[A_{m-1}^T(n), \mathbf{0}]\\ &\quad + K_m^{fT}(n)[-B_{m-1}^T(n-1), I]\}S_mX_m(n-1)\end{aligned} \tag{A.13}$$

where $\mathbf{0}$ is a matrix of appropriate dimensions with all zero entries. Comparing the above equation with (A.7) and equating coefficients of $X_m(n-1)$ gives the following equation:

$$\bar{A}_m(n) = S_m^T\left\{\begin{bmatrix}A_{m-1}(n)\\ \mathbf{0}\end{bmatrix} + \begin{bmatrix}-B_{m-1}(n-1)\\ I\end{bmatrix}K_m^f(n)\right\}. \tag{A.14}$$

Similarly, substitution of (A.11) and (A.4) into (T-1.6) results in the following equation:

$$A_m^{(p)}(n) = S_m^T\left\{\begin{bmatrix}A_{m-1}^{(p)}(n)\\ \mathbf{0}\end{bmatrix} + \begin{bmatrix}-B_{m-1}(n-1)\\ I\end{bmatrix}K_m^{f(p)}(n)\right\}; \quad p = m, m+1, \cdots, N-1. \tag{A.15}$$

In a similar way, we can obtain the order update equations of $B_m(n)$ and $C_m(n)$ as

$$\bar{B}_m(n) = T_m^T\left\{\begin{bmatrix}\mathbf{0}\\ B_{m-1}(n-1)\end{bmatrix} + \begin{bmatrix}I\\ -A_{m-1}(n)\end{bmatrix}K_m^b(n)\right\} \tag{A.16}$$

$$B_m^{(m)}(n) = T_m^T\left\{\begin{bmatrix}\mathbf{0}\\ A_{m-1}^{(m)}(n)\end{bmatrix} + \begin{bmatrix}I\\ -A_{m-1}(n)\end{bmatrix}K_m^{b(m)}(n)\right\} \tag{A.17}$$

and

$$C_m(n) = S_m^T\left\{\begin{bmatrix}C_{m-1}(n)\\ \mathbf{0}\end{bmatrix} + \begin{bmatrix}-B_{m-1}(n)\\ I\end{bmatrix}k_m^y(n)\right\}. \tag{A.18}$$

Since the first-order realizations of the lattice and direct-form bilinear filters are exactly the same, the recursions in (A.14)–(A.18) can be initialized as

$$\bar{A}_1(n) = K_1^f(n) \tag{A.19}$$

$$\bar{B}_1(n) = K_1^b(n) \tag{A.20}$$

$$A_1^{(p)}(n) = K_1^{f(p)}(n); \quad p = 1, 2, \cdots, N-1 \tag{A.21}$$

$$B_1^{(1)}(n) = K_1^{b(1)}(n) \tag{A.22}$$

and

$$C_1(n) = k_1^y(n). \tag{A.23}$$

The complete recursions are given in Table VII.

## References


[1] S. Benedetto, E. Biglieri, and V. Castellini, *Digital Transmission Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[2] J. R. Casar-Corredera, M. Garcia-Otera, and A. Figeiras-Videl, "Data echo nonlinear cancellation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, Mar. 1985, pp. 32.4.1-4.

[3] G. L. Sicuranza, A. Bucconi, and P. Mitri, "Adaptive echo cancellation with nonlinear digital filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, Mar. 1984, pp. 3.10.1-4.

[4] M. J. Coker and D. M. Simkins, "A nonlinear adaptive noise canceller," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1980, pp. 470-473.

[5] J. C. Stapleton and S. C. Bass, "Adaptive noise cancellation for a class of nonlinear dynamic reference channels," *IEEE Trans. Circuits Syst.*, vol. CAS-32, no. 2, pp. 143-150, Feb. 1985.

[6] S. Narayanan, "Application of Volterra series to intermodulation distortion of transistor feedback amplifiers," *IEEE Trans. Circuit Theory*, vol. CT-17, pp. 518-527, Nov. 1970.

[7] P. J. Marmarelis and V. Z. Marmarelis, *Analysis of Physiological Systems*. New York: Plenum, 1978.

[8] W. J. Rugh, *Nonlinear System Theory: The Volterra/Wiener Approach*. Baltimore, MD: John Hopkins University Press, 1981.

[9] M. Schetzen, *The Volterra Wiener Theory of the Nonlinear Systems*. New York: Wiley, 1980.

[10] T. Koh and E. J. Power, "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, no. 6, pp. 1445-1455, Dec. 1985.

[11] C. F. N. Cowan and P. M. Grant, "Nonlinear system modeling—concept and application," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, Mar. 1984, pp. 45.6.1-4.

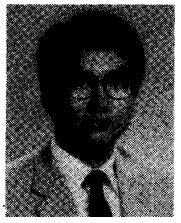
[12] C. E. Davila, A. J. Welch, and H. G. Rylander, III, "A second-order adaptive Volterra filter with rapid convergence," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1259-1263, Sept. 1987.

[13] V. J. Mathews and J. Lee, "A fast least squares second-order Volterra filter," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, 1988.

[14] G. L. Sicuranza and G. Ramponi, "Adaptive nonlinear digital filters using distributed arithmetics," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 3, June 1986.

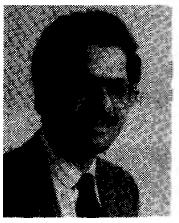
[15] M. A. Syed and V. J. Mathews, "Lattice and *QR* decomposition-based algorithms for recursive least squares adaptive nonlinear fil-

ters," in *Proc IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, May 1990.
[16] S. A. Billings, "Identification of nonlinear systems—a survey," *Proc. Inst. Elec. Eng.*, vol. 127, part D, no. 6, pp. 272–285, Nov. 1980.
[17] R. W. Brockett, "Volterra series and geometric control theory," *Automatica*, vol. 12, pp. 167–176, 1976.
[18] R. R. Mohler and W. J. Kolodziej, "An overview of bilinear system theory and applications," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, pp. 683–688, Oct. 1980.
[19] C. Bruni, G. Di Pillo, and G. Koch, "Bilinear systems: An appealing class of 'nearly linear systems' in theory and applications," *IEEE Trans. Automat. Contr.*, vol. AC-19, pp. 334–348, 1974.
[20] S. A. Billings and W. S. F. Voon, "Least square parameter estimation algorithms for nonlinear systems," *Int. J. Syst. Sci.*, vol. 15, no. 6, pp. 601–615, 1984.
[21] X. Y. Gao, W. M. Snelgrove, and D. A. Johns, "Nonlinear IIR adaptive filtering using a bilinear structure," in *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, May 1989.
[22] S. R. Parker and F. A. Perry, "A discrete ARMA model for nonlinear system identification," *IEEE Trans. Circuits Syst.*, vol. CAS-28, no. 3, Mar. 1981.
[23] F. Fnaiech and L. Ljung, "Recursive identification of bilinear systems," *Int. J. Contr.*, vol. 45, no. 2, pp. 453–470, 1987.
[24] X. Yang, R. R. Mohler, and R. M. Burton, "Adaptive suboptimal filtering of bilinear systems," *Int. J. Contr.*, vol. 52, no. 1, pp. 135–158, 1990.
[25] M. Korenberg, "Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm," *Ann. Biomed. Eng.*, vol. 16, pp. 123–142, 1988.
[26] C. R. Johnson, "Adaptive IIR filtering: current results and open issues," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 2, pp. 237–250, Mar. 1984.
[27] J. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Mag.*, vol. 6, no. 2, pp. 4–21, Apr. 1989.
[28] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[29] F. Ling and J. G. Proakis, "A generalized multichannel least squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 2, pp. 381–389, Apr. 1984.
[30] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A unified view of parametric processing algorithm for prewindowed signals," *Signal Processing*, vol. 10, pp. 335–368, 1986.
[31] J. B. Moore, "Global convergence of output error recursions in colored noise," *IEEE Trans. Automat. Contr.*, vol. AC-27, no. 6, pp. 1189–1199, Dec. 1982.
[32] C. S. Kubrusly and O. L. V. Costa, "Mean square stability conditions for discrete stochastic bilinear systems," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 11, pp. 1082–1087, Nov. 1985.
[33] C. S. Kubrusly, "On discrete stochastic bilinear systems stability," *J. Math. Anal. Appl.*, vol. 113, pp. 36–58, 1986.
[34] J. Liu, "On the existence of a general multiple bilinear time series," *J. Time Ser. Anal.*, vol. 10, no. 4, pp. 341–355, 1989.



**Heung Ki Baik** (S'80–M'83) was born in Chonju Korea, on January 5, 1955. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from the Seoul National University, Seoul, Korea, in 1977, 1979, and 1987, respectively.

He has been with the Department of Electronic Engineering, Chonbuk National University, Chonju, Korea, since 1981 and is currently an Associate Professor there. He was a Visiting Scholar in the Department of Electrical Engineering, University of Utah, Salt Lake City, UT, during 1990–1991. His research interests are in digital signal processing with emphasis on adaptive and nonlinear signal processing.



**V. John Mathews** (S'82–M'84–SM'90) was born in Nedungadappally, Kerala, India, in 1958. He received the B.E. (hons.) degree in electronics and communication engineering from the University of Madras, India, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Iowa, Iowa City, in 1980, 1981, and 1984, respectively.

From 1980 to 1984 he held a Teaching Research Fellowship at the University of Iowa, where he also worked as a Visiting Assistant Professor with the Department of Electrical and Computer Engineering from 1984 to 1985. He is currently an Associate Professor with the Department of Electrical Engineering, University of Utah, Salt Lake City. His research interests include adaptive filtering, spectrum estimation, and data compression.

Dr. Mathews was an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING during 1989–1991. He is a member of the Digital Signal Processing Technical Committee of the IEEE Signal Processing Society.