ADAPTIVE ALGORITHMS FOR IDENTIFYING RECURSIVE NONLINEAR SYSTEMS*

Heung Ki Baik** and V. John Mathews

Department of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112

ABSTRACT

This paper presents two fast least-squares lattice algorithms for adaptive non-linear filters equipped with system models involving nonlinear feedback. Such models can approximate a large class of non-linear systems adequately, and usually with considerable parsimony in the number of coefficients required. For simplicity of presentation, we consider the bilinear system model in the paper, even though the results are applicable to more general system models. The computational complexity of the algorithms is an order of magnitude smaller than previously available methods. Results of several experiments that demonstrate the properties of the adaptive bilinear filters as well as compare their performances with two other algorithms that are computationally more expensive are also presented in this paper.

I. INTRODUCTION

While linear filters and system models have been very useful in a large variety of applications and are conceptually and implementationally simple, there are several applications in which they will not perform well at all. This paper is concerned with developing adaptive filtering algorithms for nonlinear systems. System analysis using nonlinear structures has several applications, including those in channel equalization, echo cancellation, noise cancellation, characterizing semiconductor devices, modeling biological phenomenon, 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 [3]. Several researchers have developed adaptive filters based on truncated Volterra series expansion [1, 2, 4 - 6]. 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 that is 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 [7]

$$y(n) = \sum_{i=1}^{M} P_i[x(n),\ldots,x(n-N+1), y(n-1),\ldots,y(n-N+1)] \quad (1)$$

where $P_i[\ldots]$ 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 efficiency. Perhaps the simplest among the nonlinear feedback models is the bilinear system model. The input-output relationship for a bilinear system is given by the nonlinear difference equation

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij} x(n-i)\, y(n-j) + \sum_{j=1}^{N-1} c_j y(n-j). \quad (2)$$

An 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 [8]. 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 [9, 10]. 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 [11 - 14]. The results in [11, 14] involve direct-form structures and employ the conventional recurvise 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 [12] 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 [13] 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, the methods presented in this paper can be very easily extended to more general nonlinear output feedback structures.

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 the cost function

$$\xi_N(n) = \sum_{k=1}^{n} \lambda^{n-k}[y(k) - \hat{y}_n(k)]^2 \quad (3)$$

is minimized at each instant. In this expression λ, $0 < \lambda \leq 1$, 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,

* This work was supported in parts by a Fellowship from the Korean Science Foundation, NSF Grants MIP 8708970 and MIP 8922146, and a University of Utah Faculty Fellow Award.

** On leave from Chonbuk National University, Chonju, Korea.

in which the adaptive filter output is estimated as

$$\hat{y}_n(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij}(n)x(n-i)\, y(n-j) + \sum_{j=1}^{N-1} c_j(n)y(n-j) \tag{4}$$

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

$$\hat{y}_n(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} b_{ij}(n)x(n-i)\, y(n-j) + \sum_{j=1}^{N-1} c_j(n)\hat{y}_{n-j}(n-j) \tag{5}$$

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 $\hat{y}_{n-j}(n-j)$; $j = 1,2,\ldots,N-1$ as can be seen from the above equation.

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 [6]. The basic idea is to partition the input vector (say $U_N(n)$) 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.)

CH 1	:	$[x(n), x(n-1), \ldots, x(n-N+1)]$
CH 2	:	$[d(n-1), d(n-2), \ldots, d(n-N+1)]$
CH 3	:	$[x(n-1)d(n-1), x(n-2)d(n-2), \ldots, x(n-N+1)d(n-N+1)]$
CH 4	:	$[x(n)d(n-1), x(n-2)d(n-2), \ldots, x(n-N+2)d(n-N+1)]$
		......
CH 2N-1:		$[x(n-N+1)d(n-1)]$
CH 2N	:	$[x(n)d(n-N+1)]$. (6)

Now, we can consider each of the above vectors as being formed using successive samples of signals from a different input channel. Channel 1 has N coefficients associated with it while channels 2, 3 and 4 have N-1 channels associated for the (2k-1)-th and 2k-th channels is N-k+1 for k≥3. To simplify the notations, let $x_i(n)$; i=0,1,...,N-1 be defined as

$$\begin{aligned}
x_0(n) &= [x(n)] \\
x_1(n) &= [d(n-1), x(n-1)d(n-1, x(n)d(n-1)]^T \\
x_2(n) &= [x(n-2)d(n-1), x(n)d(n-2)]^T \\
&\ldots \\
x_{N-1}(n) &= [x(n-N+1)d(n-1), x(n)d(n-N+1)]^T .
\end{aligned} \tag{7}$$

The above representation is useful since all the channels belonging to each subset have the same number of delays.

The first task involved in developing a lattice structure for bilinear filters is to obtain a Gram-Schmidt orthogonalization of the input data. While there are several approaches for performing the Gram-Schmidt orthogonalization, our method follows the technique in [16] very closely. The development of the lattice structure depends critically on two different partitions of the input vector. The first partition divides the elements of $U_N(n)$ into N "backward" input vectors as follows.

$$\begin{aligned}
x_0^b(n) &= [x_0(n)] \\
x_1^b(n) &= [x_0(n-1), x_1(n)]^T \\
x_1^b(n) &= [x_0(n-2), x_1(n-1), x_2(n)]^T \\
&\ldots \\
x_{N-1}^b(n) &= [x_0(n-N+1), x_1(n-N+2), \ldots, x_{N-1}(n)]^T .
\end{aligned} \tag{8}$$

Let $b_m(n)$ be the optimal estimation error vector when $x_m^b(n)$ is estimated using $x_0^b(n), x_1^b(n), \ldots, x_{m-1}^b(n)$ (Note that $b_0(n) = x_0^b(n)$ by definition). It is well known that the "backward prediction" residuals $\{b_m(n);\ m = 0,1,\ldots,N-1\}$ form an orthogonal decomposition of $x_i^b(n)$; $i = 0,1,\ldots,m-1$. 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 (8) (which is the same as the elements of $U_m(n)$) as

$$\begin{aligned}
x_{m:0}^f(n) &= [x_0(n-m)] \\
x_{m:1}^f(n) &= [x_0(n-m+1), x_1(n-m+1)]^T \\
x_{m:2}^f(n) &= [x_0(n-m+2), x_1(n-m+2), x_2(n-m+2)]^T \\
&\ldots \\
x_{m:m}^f(n) &= [x_0(n), x_1(n), \ldots, x_m(n)]^T .
\end{aligned} \tag{9}$$

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 $x_{m:0}^f(n), x_{m:1}^f(n), \ldots x_{m:m-1}^f(n)$. As usual we will define $f_0(n)$ to be the same as $x_{0:0}^f(n)$

The derivation of the lattice equations is somewhat lengthy and is omitted here. The details can be found in [15]. The algorithm is given in Table I. An operations count will show that the computational complexity is proportional to $O(N^3)$ multiplications per iteration.

III. EXPERIMENTAL RESULTS

The input signal to the adaptive filter was a colored, zero-mean and pseudorandom Gaussian noise. The desired response signal y(n) was obtained as a noisy measurement of a bilinear system with coefficients as given in Table II. The input signal was such that the output of the unknown system had unit mean squared value. The adaptive filter was run with the same number of coefficients as the unknown system. All of the results presented are ensemble averages over fifty independent runs.

The time averages of the ensemble-averaged mean-squared error during the interval from n = 4001 to n = 5000 are given in Table III for three different noise levels and two different values of λ. These tabulations 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 $c_1(n)$ are plotted in Fig. 1 for different noise levels and λ = 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. The mean behavior of the coefficients in the output error adaptive filter appears to be much less sensitive to measurement noise. The mean behavior of $c_1(n)$ obtained using the extended least-squares (ELS) algorithm and the recursive prediction-error method (RPEM), which are discussed in [14] are displayed in Fig. 2. Similar to some of the results in [14], the behavior of the coefficient obtained using the ELS algorithm is very erratic. RPEM performs the best among all the $O(N^4)$ complexity algorithms discussed in [14]. 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. 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.

REFERENCES

1. G. L. Sicuranza, A. Bucconi, and P. Mitri, "Adaptive echo cancellation with nonlinear digital filters," *Proc. ICASSP* (San Diego, California), pp. 3.10.1-4, Mar. 1984.
2. M. J. Coker and D. M. Simkins, "A nonlinear adaptive noise canceller," *Proc. ICASSP*, pp. 470-473, 1980.
3. M. Schetzen, *The Volterra Wiener Theory of the Nonlinear Systems*, New York: Wiley and Sons, 1980.
4. 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.
5. V. J. Mathews and J. Lee, "A fast least-squares second-order Volterra filter," *ICASSP* (New York),pp. 1383-1386, 1988.
6. M. A. Syed and V. J. Mathews, "Lattice and QR decompositon-based algorithms for recursive least squares adaptive nonlinear filters," *Proc. IEEE Int. Symp. Circuits and Systems* (New Orleans, Louisiana), May 1990.
7. S. A. Billings, "Identification of nonlinear systems-a survey," *IEE Proceedings*, Vol. 127, Part D, No. 6, pp. 272-285, Nov. 1980.
8. Roger W. Brockett, "Volterra series and geometric control theory," *Automatica*, Vol. 12, pp. 167-176, 1976.
9. 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.
10. 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.
11. S. A. Billings and W. S. F. Voon, "Least square parameter estimation algorithms for nonlinear systems," *Int. J. System Sci.*, Vol. 15, No. 6, pp. 601-615, 1984.
12. X. Y. Gao, W. M. Snelgrove, and D. A. Johns, "Nonlinear IIR adaptive filtering using a bilinear structure," *Proc. IEEE Int. Symp. Circuits and Systems*, (Portland, Oregon), May 1989.
13. S. R. Parker and F. A. Perry, "A discrete ARMA model for nonlinear system identification," *IEEE Trans. Circuits and Systems*, Vol. CAS-28, No. 3, Mar. 1981.
14. F. Fnaiech and L. Ljung, "Recursive identification of bilinear systems," *Int. J. Control*, Vol. 45, No. 2, pp. 453-470, 1987.
15. H. K. Baik and V. J. Mathews, "Adaptive lattice biinear filters," submitted to *IEEE Trans. Signal Processing*.
16. F. Ling and J. G. Proakis, "A generalized multichannel least squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acoust., Speech, Signal Proc.*, Vol. ASSP-32, No. 2, pp. 381-389, April 1984.

TABLE I.
THE RECURSIVE LEAST-SQUARES ADAPTIVE LATTICE BILINEAR FILTER

Time Initialization

DO (T1)-(T2), for m = 0,1,...,N-1

$$\alpha_m(0) = 1 \tag{T1}$$

$$R^f_m(0) = R^b_m(0) = \begin{cases} \delta \geq 0 & \text{if } m = 0 \\ \delta I_{2m+2}, & \text{otherwise} \end{cases} \tag{T2}$$

Order Initialization

DO (T3)-(T7), for n = 1,2,...

$$\alpha_0(n) = 1 \tag{T3}$$

$$R^f_0(n) = R^b_0(n) = \lambda R^f_0(n-1) + x^2(n) \tag{T4}$$

$$f_0(n) = b_0(n) = x(n) \tag{T5}$$

$$f^{(p)}_0(n) = \begin{cases} [d(n-1), x(n-1)d(n-1), x(n)d(n-1)]^T\ \dagger, & p=1 \\ [x(n-p)d(n-1), x(n)d(n-p)]^T, & p=2,3,\ldots,N-1 \end{cases} \tag{T6}$$

$$e_0(n) = y(n) \tag{T7}$$

Iteration Procedure

DO (T8)-(T21), for n = 1,2,...

DO (T8)-(T19), for m = 1,2,...,N-1

$$\Delta_m(n) = \lambda\Delta_m(n-1) + b_{m-1}(n-1)f^T_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T8}$$

$$\Delta^{f(m)}_m(n) = \lambda\Delta^{f(m)}_m(n-1) + b_{m-1}(n-1)f^{(m)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T9}$$

$$\Delta^{b(m)}_m(n) = \lambda\Delta^{b(m)}_m(n-1) + f_{m-1}(n)f^{(m)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T10}$$

$$f_m(n) = \begin{bmatrix} f_{m-1}(n) - \Delta^T_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \\ f^{(m)}_{m-1}(n) - \Delta^{f(m)T}_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \end{bmatrix} \tag{T11}$$

$$b_m(n) = \begin{bmatrix} b_{m-1}(n-1) - \Delta_m(n)R^{-f}_{m-1}(n)f_{m-1}(n) \\ f^{(m)}_{m-1}(n) - \Delta^{b(m)T}_m(n)R^{-f}_{m-1}(n)f_{m-1}(n) \end{bmatrix} \tag{T12}$$

DO (T13)-(T14) for p=m+1,m+2,...,N-1

$$\Delta^{f(p)}_m(n) = \lambda\Delta^{f(p)}_m(n-1) + b_{m-1}(n-1)f^{(p)T}_{m-1}(n)/\alpha_{m-1}(n-1) \tag{T13}$$

$$f^{(p)}_m(n) = f^{(p)}_{m-1}(n) - \Delta^{f(p)T}_m(n)R^{-b}_{m-1}(n-1)b_{m-1}(n-1) \tag{T14}$$

$$\Delta^y_m(n) = \lambda\Delta^y_m(n-1) + b_{m-1}(n)e_{m-1}(n)/\alpha_{m-1}(n) \tag{T15}$$

$$e_m(n) = e_{m-1}(n) - \Delta^{yT}_m(n)R^{-b}_{m-1}(n)b_{m-1}(n) \tag{T16}$$

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

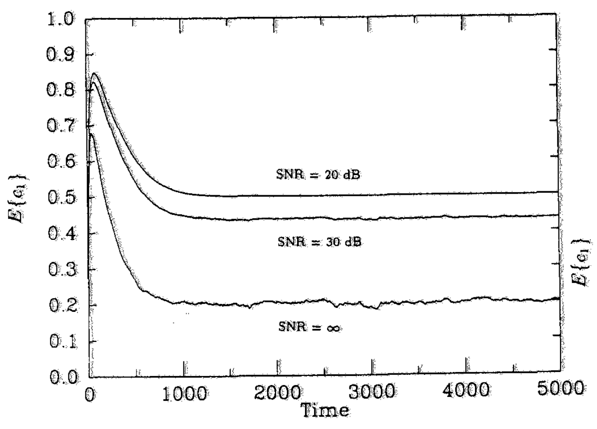
$$\alpha_m(n) = \alpha_{m-1}(n) - b^T_{m-1}(n)R^{-b}_{m-1}(n)b_{m-1}(n) \qquad \text{(T17)}$$

$$R^{-f}_m(n) = \lambda^{-1}R^{-f}_m(n-1) - \frac{\lambda^{-2}R^{-f}_m(n-1)f_m(n)f^T_m(n)R^{-f}_m(n-1)}{\alpha_m(n-1) + \lambda^{-1}f^T_m(n)R^{-f}_m(n-1)f_m(n)} \qquad \text{(T18)}$$

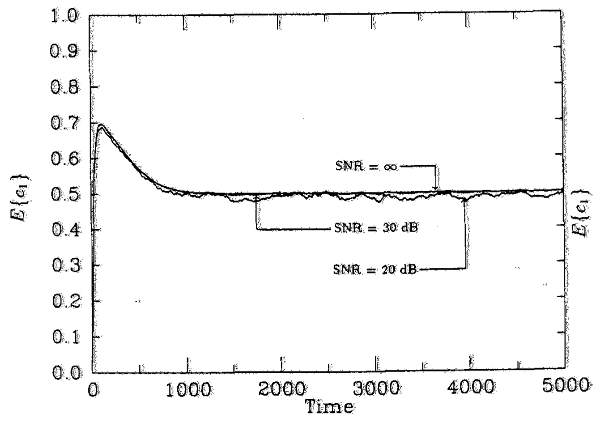
$$R^{-b}_m(n) = \lambda^{-1}R^{-b}_m(n-1) - \frac{\lambda^{-2}R^{-b}_m(n-1)b_m(n)b^T_m(n)R^{-b}_m(n-1)}{\alpha_m(n) + \lambda^{-1}b^T_m(n)R^{-b}_m(n-1)b_m(n)} \qquad \text{(T19)}$$

$$\Delta^y_N(n) = \lambda\Delta^y_N(n-1) + b_{N-1}(n)e_{N-1}(n)/\alpha_{N-1}(n) \qquad \text{(T20)}$$

$$e_N(n) = e_{N-1}(n) - \Delta^{yT}_N(n)R^{-b}_{N-1}(n)b_{N-1}(n) \qquad \text{(T21)}$$



(a)



(b)

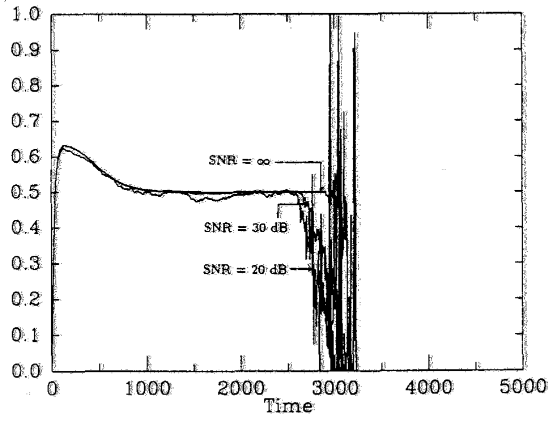
Figure 1. Mean trajectories of $c_1(n)$: (a) equation error algorithm and (b) output error algorithm

TABLE II.
COEFFICIENTS OF THE UNKNOWN BILINEAR SYSTEM USED IN THE EXPERIMENTS

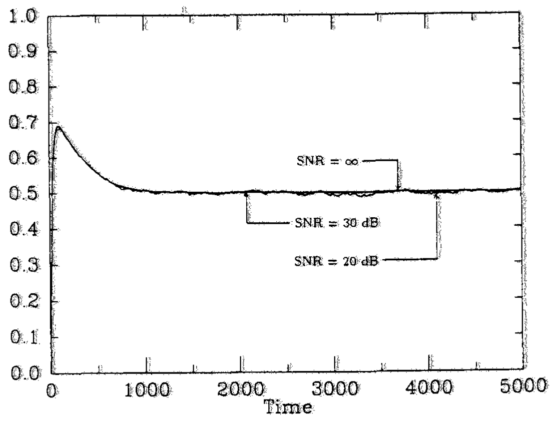
$a_0 = 1.0$	$a_1 = 1.0$	$a_2 = 1.0$
$b_{0,1} = 0.3$	$b_{1,1} = -0.2$	$b_{2,1} = 0.1$
$b_{0,2} = 0.1$	$b_{1,2} = -0.2$	$b_{2,2} = 0.3$
	$c_1 = 0.5$	$c_2 = -0.5$

TABLE III
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
∞	0.104E-9	0.577E-10	0.137E-9	0.714E-10
30 dB	0.136E-2	0.127E-2	0.942E-3	0.887E-3
20 dB	0.118E-1	0.109E-1	0.941E-2	0.866E-2



(a)



(b)

Figure 2. Mean trajectories of $c_1(n)$: (a) ELS algorithm and (b) RPEM algorithm