Lattice Algorithms for Recursive Least Squares Adaptive Second-Order Volterra Filtering

Mushtaq A. Syed and V. John Mathews

***Abstract*— This paper presents two computationally efficient recursive least-squares (RLS) lattice algorithms for adaptive nonlinear filtering based on a truncated second-order Volterra system model. The lattice formulation transforms the nonlinear filtering problem into an equivalent multichannel, linear filtering problem and then generalizes the lattice solution to the nonlinear filtering problem. One of the algorithms is a direct extension of the conventional RLS lattice adaptive linear filtering algorithm to the nonlinear case. The other algorithm is based on the QR decomposition of the prediction error covariance matrices using orthogonal transformations. Several experiments demonstrating and comparing the properties of the two algorithms in finite and "infinite" precision environments are included in the paper. The results indicate that both the algorithms retain the fast convergence behavior of the RLS Volterra filters and are numerically stable.**

I. Introduction

LINEAR FILTERING, because of its analytical simplicity, has progressed quite rapidly. However, there are a number of applications in which the performance of linear filters is unacceptable and one has to resort to nonlinear filters. Nonlinear filters have been used in such diverse areas as communications [1], [4], [34] and [35], biological signal processing [6], [9], [13], [16], image processing [27], and semiconductor modeling [24], [30].

The truncated Volterra series expansion is a commonly used nonlinear model. In this model, the output $y(n)$ of any causal, discrete-time, time-invariant nonlinear system is represented as a function of the input $x(n)$ using the Volterra series expansion

$$\begin{aligned} y(n) &= h_0 + \sum_{m_1=0}^{N_1-1} h_1(m_1)\, x(n-m_1) \\ &+ \sum_{m_1=0}^{N_2-1} \sum_{m_2=0}^{N_2-1} h_2(m_1,m_2)\, x(n-m_1)\, x(n-m_2) + \cdots \\ &+ \sum_{m_1=0}^{N_p-1} \cdots \sum_{m_p=0}^{N_p-1} h_p(m_1,m_2,\ldots,m_p) \\ &\qquad x(n-m_1)\cdots x(n-m_p), \end{aligned} \tag{1}$$

Manuscript received February, 1992; revised September, 1993. This paper was recommended by Associate Editor G. S. Moschytz. This research was supported in part by NSF Grants MIP-8708970 and MIP-8922146, and by a University of Utah Faculty Fellow Award.

Parts of this paper were presented at the IEEE International Symposium on Circuits and Systems, New Orleans, LA, May 1990.

M. A. Syed was with the Department of Electrical Engineering, the University of Utah, Salt Lake City, UT. He is now with Digicom Systems, Inc., Milpitas, CA. V. J. Mathews is with Department of Electrical Engineering, the University of Utah, Salt Lake City, UT.

IEEE Log Number 9216362.

where $h_r(m_1,m_2,\ldots,m_r)$ is the r-th order Volterra kernel [31], [33] of the system. We will assume, without loss of generality, that the Volterra kernels are symmetric, i.e., $h_r(m_1,m_2,\ldots,m_r)$ is left unchanged by any of the $r!$ permutations of the arguments $m_1,m_2,\ldots,m_r$. Because of the relatively large number of potential applications, there has been quite an increase in the research activities on adaptive Volterra filtering in recent years. Early works on adaptive Volterra filters [6], [11], [12] were based on the LMS algorithm. Even though they are computationally simple, they suffer from slow and input signal-dependent convergence behavior and hence are not useful in many applications. More recently, Lee and Mathews [14] presented a fast transversal algorithm for recursive least-squares (RLS) adaptive Volterra filtering. The algorithm was derived by transforming the nonlinear problem into a multichannel linear filtering problem and then using the ideas employed for developing computationally efficient, multichannel RLS adaptive transversal filters. The fast RLS Volterra filter is rapidly convergent and has good tracking properties; however, it suffers from poor numerical properties.

In this paper we present two computationally efficient, RLS adaptive lattice second-order Volterra filters. The algorithms can be easily extended to higher order nonlinearities. The first algorithm extends the conventional RLS lattice linear filter to the nonlinear case. The second algorithm is based on QR-decomposition (QRD) of the prediction error covariance matrices. It can be obtained from the first algorithm by Cholesky factorization of the error covariance matrices and every component of the algorithm can be implemented using Givens rotations [7] alone.

The structure presented in this paper is based on the earlier work of Ling and Proakis [17], [18]. It is also very similar to the lattice structure developed by Zarzycki [38]. It is different from the structures presented in [11], [15] in the sense that it can be applied to Volterra systems with arbitrary input signals and arbitrary shapes for the Volterra kernels. The lattice structure presented in [11] requires that the input signals be Gaussian and the lattice filter presented in [15] is applicable only to systems with very special shapes for the Volterra kernels. Our structure does not have these drawbacks. Also, it can be extended to more general types of nonlinear models [2]. A recent tutorial introduction to nonlinear lattice filters can be found in [22]. While both the lattice structures are novel, we believe that this is the first time that a QRD-based approach has been employed in any type of adaptive Volterra filtering problem. This paper also presents extensive simulation results comparing the performance of QRD-based and conventional nonlinear lattice filters in finite precision environments. While such studies are obviously new in the context of nonlinear filtering, we believe that some of the results will shed more light on the properties of these two lattice structures even in linear filtering applications.

The rest of the paper is organized as follows. In Section II we present the lattice structure and the conventional RLS adaptive lattice Volterra filtering algorithm. The QRD-based adaptive nonlinear filtering algorithm is presented in Section III. Experimental results are presented in Section IV and the concluding remarks in Section V.

II. The Conventional RLS Adaptive Second-Order Lattice VolterraFilter

Consider the problem of recursively estimating the desired signal $d(n)$ using a truncated second-order Volterra series expansion in the primary input signal $x(n)$ by minimizing the exponentially weighted least-squares cost function

$$\xi_N(n) = \sum_{k=0}^{n} \lambda^{n-k}\Big(d(k) - \hat{d}_n(k)\Big)^2 \tag{2}$$

where

$$\hat{d}_n(k) = \sum_{m_1=0}^{N-1} \hat{a}_{m_1}(n)\, x\,(k-m_1) - \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} \hat{b}_{m_1,m_2}(n) \times (k-m_1)x(k-m_2), \tag{3}$$

is the estimate of the desired signal at time k obtained using the adaptive filter coefficients at time n and λ is a constant weighting factor in the range (0, 1] that controls the speed of convergence and the tracking ability of the adaptive filter. $\hat{a}_{m_1}(n)$ and $\hat{b}_{m_1,m_2}(n)$ are the linear and quadratic coefficients, respectively, of the second-order Volterra filter. (The upper limits of all three summations in the Volterra series expansion in (2) have been set equal only for convenience in presentation. The generalization to arbitrary limits is straightforward). Let us define the input vector $\underline{X}(n)$ and the coefficient vector $\underline{W}(n)$, both of size $N(N+3)/2$ entries, at time n as

$$\underline{X}(n) = [x(n), x^2(n), x(n-1)x^2(n-1), x(n)x(n-1), \ldots, x(n)\,x\,(n-N+1)]^T \tag{4a}$$

and

$$\underline{W}(n) = \Big[\hat{a}_1(n), \hat{b}_{1,1}(n), \hat{a}_2(n), \hat{b}_{2,2}(n), \hat{b}_{1,2}(n), \ldots, \hat{b}_{1,N-1}(n)\Big]^T, \tag{4b}$$

respectively. In the above $(\bullet)^T$ denotes transpose of the matrix $(\bullet)$. Equation (2) can be rewritten using (4a) and (4b) as.

$$\xi_N(n) = \sum_{k=0}^{n} \lambda^{n-k}\big(d(k) - \underline{W}^T(n)\underline{X}(k)\big)^2. \tag{5}$$

The optimal solution to the problem can be easily shown to be

$$\underline{W}_{\rm opt}(n) = \Omega^{-1}(n)p(n), \tag{6}$$

where

$$\Omega(n) = \sum_{k=0}^{n} \lambda^{n-k}\underline{X}(k)\underline{X}^T(k) \tag{7}$$

and

$$P(n) = \sum_{k=0}^{n} \lambda^{n-k} d(k)\underline{X}(k). \tag{8}$$

Direct evaluation of (6) is, in general, computationally inefficient and often prone to numerical instability. We would like to develop computationally efficient and numerically stable algorithms to iteratively solve the optimization problem. In order to accomplish this we develop a lattice structure for second-order Volterra filters. As discussed in the introduction, we restrict ourselves to the second-order Volterra structure because of the pedagogical simplicity it provides and the ideas are equally applicable to higher-order nonlinearities also. It should be pointed out here that the computational efficiency of our algorithms is only in applications where direct-form parameters are not required. The algorithms estimate the desired response signal using a lattice parameterization of the nonlinear system.

For the development of the lattice structure, it is convenient to rewrite the entries of the input vector $\underline{X}(n)$ in the following matrix form. (See bottom of page.) Now, each row of the above matrix can be considered as made up of samples of signals from different channels. However, the number of samples used in the estimation problem is different for different channels. There are N samples used in the first two channels, $N-1$ in the third, $N-2$ in the fourth, and so on to the $(N+1)$-th channel which uses only a single sample at each time. As is well known, the key step in the development of the lattice structure is the Gram-Schmidt orthogonalization of the

$$\underline{X}_N(n) = \begin{bmatrix} x(n) & x(n-1) & x(n-2) & \ldots & x(n-N+1) \\ x^2(n) & x^2(n-1) & x^2(n-2) & \ldots & x^2(n-N+1) \\ & x(n)x(n-1) & x(n-1)x(n-2) & \ldots & x(n-N+2)x(n-N+1) \\ & & x(n)x(n-2) & \ldots & x(n-N+3)x(n-N+1) \\ & & & & \bullet \\ & & & & \bullet \\ & & & & x(n)x(n-N+1) \end{bmatrix}. \tag{9}$$

TABLE I
DEFINITIONS OF THE VARIABLES EMPLOYED IN THE M-TH STAGE OF THE LS LATTICE ADAPTIVE SECOND-ORDER VOLTERRA FILTER.

$\underline{f}_m(n)$	Forward prediction error vector	$r^b_m(n)$	Backward prediction error correlation matrix
$\underline{b}_m(n)$	Backward prediction error vector	$\Delta_m(n)$	Forward and Backward prediction error crosscorrelation matrix
$k^f_m(n)$	Forward reflection coefficient matrix	$\alpha_m(n)$	Likelihood variable
$k^b_m(n)$	Backward reflection coefficient matrix	$k^{f(j)}_m(n)$	Auxiliary reflection coefficient vector (j=m+2,...,N+1)
$e_m(n)$	Joint process estimation error at the m-th stage	$f^{(j)}_m(n)$	Auxiliary prediction error
$k^y_m(n)$	Joint process estimation coefficient error	$k^{b(j)}_m(n)$	Auxiliary reflection coefficient vector
$r^j_m(n)$	Forward prediction error correlation matrix	$b^{(j)}_m(n)$	Auxiliary prediction error

input data to generate an orthogonal basis for the vector space (defined by the least-squares optimization criterion) spanned by the input data. We can define this orthogonal basis set as a set of backward prediction errors. The backward prediction error vector $\underline{b}_i(n)$ is a vector of $i+2$ elements and is defined as the optimal LS estimation error vector when the i-th column vector of the data matrix $X_N(n)$,

$$[x(n-i), x^2(n-i), x(n-i+1)x(n-i), \ldots, x(n)x(n-i)]^T, \tag{10}$$

is estimated using the elements of columns 0 through $i-1$. $\underline{b}_0(n)$ is defined to be $[x(n), x^2(n)]^T$. (Note that the column vectors are numbered from zero onwards.)

Efficient computation of the backward prediction errors requires the computation of the forward prediction errors. The forward prediction error vector $\underline{f}_i(n), i = 0, 1, \ldots, N-1$, is defined as the optimal LS error in estimating the vector

$$[x(n), x^2(n), x(n)x(n-1), \ldots, x(n)x(n-i)]^T \tag{11}$$

(this vector is formed by the most recent samples belonging to each of the first $i+2$ channels) using the elements of the matrix.

$$\begin{bmatrix} x(n-1) & x(n-2) & \ldots & x(n-i) \\ x^2(n-1) & x^2(n-2) & \ldots & x^2(n-i) \\ & x(n-1)x(n-2) & \ldots & x(n-i+1)x(n-i) \\ & & & \vdots \\ & & & x(n-1)x(n-i) \end{bmatrix} \tag{12}$$

$\underline{f}_o(n)$ is defined as $[x(n), x^2(n)]^T$.

One major difference between traditional multichannel lattice filters and the lattice Volterra filters of this paper is the fact that the backward and forward prediction error vectors of different orders contain different number of elements in them. This in turn implies that as the order of prediction increases by one, both the forward and backward predictors must predict one extra signal than at the previous step. The new element that must be predicted for the i-th order ($i \geq 1$) is $x(n)x(n-i)$. These additional computations have to be done outside the basic lattice structure. In other words, each predictor stage is a lattice of one dimension greater than the preceding stage.

Details of derivations of the lattice update equations are presented in Appendix A and the relevant equations are given in Table II. All the variables used in Table II are defined in Table I. Note that we have used $\bar{\underline{b}}_i(n)$ and $\bar{\underline{f}}_i(n)$ in (T-2.4) and (T-2.5) to denote the vectors formed by the first $i+1$ elements of $\underline{b}_i(n)$ and $\underline{f}_i(n)$, respectively. Fig. 1 is a schematic block diagram of the lattice structure for $N = 3$.

Derivations of the recursive equations for updating the optimal forward and backward reflection coefficient matrices, the joint process estimation coefficient vector, and the auxiliary optimal coefficient vectors are quite straightforward. For example, consider the optimal LS forward reflection coefficient matrix $k^f_m(n)$ of the m-th stage. It is the optimal coefficient matrix in estimating $\underline{f}_{m-1}(n)$ using $\underline{b}_{m-1}(n-1)$. It can be directly computed as

$$k^f_m(n) = r^{-b}_{m-1}(n-1)\Delta_m(n), \tag{13}$$

where $(\bullet)^{-b}$ denotes the matrix inverse of $(\bullet)^b$,

$$r^b_{m-1}(n) = \sum_{k=1}^{n} \lambda^{n-k} \underline{b}_{m-1,n}(k)\underline{b}^T_{m-1,n}(k) \tag{14}$$

is the LS autocorrelation matrix of $\underline{b}_{m-1,n}(k)$ and

$$\Delta_m(n) = \sum_{k=1}^{n} \lambda^{n-k} \underline{b}_{m-1,n}(k)\underline{f}^T_{m-1,n}(k) \tag{15}$$

is the LS crosscorrelation matrix of $\underline{b}_{m-1,n}(k)$ and $\underline{f}_{m-1,n}(k)$. In (14) and (15) $\underline{b}_{m,n}(k)$ and $\underline{f}_{m,n}(k)$ are the backward and forward prediction error vectors of order m at time k that were computed using the optimal coefficient matrices at time n. In the sequel, we need to use only vectors of the form $\underline{b}_{m,k}(k)$ and $\underline{f}_{m,k}(k)$ and therefore we have dropped the second subscript as it causes no confusion. The correlation matrices can be recursively updated [10], [17] as

$$\Delta_m(n) = \lambda\Delta_m(n-1) + \frac{\underline{b}_{m-1}(n-1)\underline{f}^T_{m-1}(n)}{\alpha_{m-1}(n-1)} \tag{16}$$

and

$$r^b_{m-1}(n) = \lambda r^b_{m-1}(n-1) + \frac{\underline{b}_{m-1}(n)\underline{b}^T_{m-1}(n)}{\alpha_{m-1}(n)}, \tag{17}$$

where

$$\alpha_m(n) = \alpha_{m-1}(n) + \underline{b}^T_{m-1}(n)r^{-b}_{m-1}(n)\underline{b}_{m-1}(n) \tag{18}$$

is the "likelihood" variable for the m-th stage. It is easy to show that $0 \leq \alpha_m(n) \leq 1$. Using the matrix inversion lemma [10], [17], [18], we can recursively update the inverse of the

TABLE II
SECOND—ORDER VOLTERRA LATTICE FILTER

Initialization

$$\underline{f}_0(n) = \underline{b}_0(n) = \left[x(n),\ x^2(n)\right]^T \quad \text{(T-2.1)}$$

$$e_0(n) = d(n) \quad \text{(T-2.2)}$$

$$f_0^{(j)}(n) = x(n)x(n-j+2),\ j = 3, \ldots, N+1 \quad \text{(T-2.3)}$$

DO (T-2.4) to (T-2.7) for i=1, 2,..., N-1

$$f_i(n) = \left| \begin{array}{c} \overline{\underline{f}}_i(n) = \underline{f}_{i-1}(n) - k_i^{f^T} \underline{b}_{i-1}(n)(n-1) \\ f_i^{(i+2)}(n) = f_{i-1}^{(i+2)}(n) - k_i^{f(i+2)^T}(n)\underline{b}_{i-1}(n-1) \end{array} \right| \quad \text{(T-2.4)}$$

$$\underline{b}_i^{(n)} = \left| \begin{array}{c} \overline{\underline{b}}_i^{(n)} = \underline{b}_{i-1}(n-1) - k_i^{b^T}(n)\underline{f}_{i-1}(n) \\ b_i^{(i+2)}(n) = f_{i-1}^{(i+2)}(n) - k_i^{b(i+2)^T}(n)\underline{f}_{i-1}(n) \end{array} \right| \quad \text{(T-2.5)}$$

DO (T-2.6) for j+i+2,..., N+1

$$f_i^{(j)}(n) = f_{i-1}^{(j)}(n) - k_i^{f(j)^T}(n)\underline{b}_{i-1}(n-1) \quad \text{(T-2.6)}$$

$$e_i(n) = e_{i-1}(n) - k_i^{y^T}(n)\underline{b}_{i-1}(n) \quad \text{(T-2.7)}$$

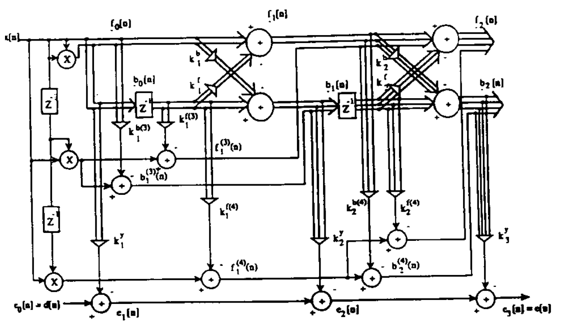

Fig. 1. Block diagram of filter structure for Volterra systems with $N = 3$.

autocorrelation matrix as

$$r_m^{-b}(n) = \lambda^{-1} r_m^{-b}(n-1) - \frac{\lambda^{-2} r_m^{-b}(n-1)\underline{b}_m(n)\underline{b}_m^T(n) r_m^{-b}(n-1)}{\alpha_m(n) + \lambda^{-1}\underline{b}_m^T(n) r_m^{-b}(n-1)\underline{b}_m(n)}. \quad (19)$$

Similarly, we can derive recursive equations for updating the backward reflection coefficient matrix $k_m^b(n)$, the joint process estimation coefficient vector $k_m^y(n)$, and the auxiliary coefficient vectors $k_m^{f(m)}(n)$ and $k_m^{b(m)}(n)$. Also, an equation similar to (19) can be derived for recursively updating the forward prediction error covariance matrix.

Equation (13) for computing the forward reflection coefficient matrix, and similar equations for the other coefficients, indicate that the lattice parameters are updated indirectly as the ratio of a "prediction error crosscorrelation" and a "prediction error autocorrelation." This indirect method of updating the lattice parameters has two drawbacks [20]. First, it can lead to a degradation in the accuracy of the algorithm. Second, it makes the algorithm unsuitable for fixed-point implementations. These problems can be overcome by directly updating the coefficient matrices. We will derive the direct update equation for the forward reflection coefficient matrix $k_m^f(n)$. Substituting (16) in (13) we get

$$k_m^f(n) = \lambda r_{m-1}^{-b}(n-1)\Delta_m(n-1) + \frac{r_{m-1}^{-b}(n-1)\underline{b}_{m-1}(n-1)\underline{f}_{m-1}^T(n)}{\alpha_{m-1}(n-1)} \quad (20)$$

Using (13) and (17) we can express the first term of (20) as

$$\lambda r_{m-1}^{-b}(n-1)\Delta_m(n-1) = k_m^f(n-1) - \frac{r_{m-1}^{-b}(n-1)\underline{b}_{m-1}(n-1)\underline{b}_{m-1}^T(n-1)k_m^f(n-1)}{\alpha_{m-1}(n-1)}. \quad (21)$$

Substituting (21) in (20) we get the equation

$$k_m^f(n) = k_m^f(n-1) + \frac{r_{m-1}^{-b}(n-1)\underline{b}_{m-1}(n-1)}{\alpha_{m-1}(n-1)} \times \left[\underline{f}_{m-1}^T(n) - \underline{b}_{m-1}^T(n-1)k_m^f(n-1)\right] \quad (22)$$

for directly updating the forward reflection coefficient matrix.

Similarly, we can derive direct update equations for the other coefficient matrices and vectors. The complete algorithm for the fast RLS second-order Volterra adaptive lattice algorithm is presented in Table III. A count of the arithmetical operations involved in the implementation of the algorithm will show that it requires $\frac{13N^3}{3} + 17N^2 + \frac{137N}{3} - 27$ multiplications and $\frac{2N^3}{3} + 6N^2 + \frac{43N}{3} - 21$ divisions per iteration. Notice that this complexity is comparable to that of the fast RLS transversal Volterra filter in [14].

III. QRD-BASED ADAPTIVE LATTICE SECOND-ORDER VOLTERRA FILTER

In recent years, the quest for numerically stable least-squares adaptive filtering algorithms has kindled interest in

TABLE III
THE CONVENTIONAL LEAST-SQUARES ADAPTIVE LATTICE SECOND-ORDER VOLTERRA FILTER

Time Initialization:

$$\alpha_m(0) = 1.0, \quad m = 0, 1, \ldots, N-1 \tag{T-3.1}$$

$$r^f_m(0) = r^b_m(0) = \delta I_{m+2}, \quad m = 0, 1, \ldots, N-1 \quad \delta > 0 \tag{T-3.2}$$

Order Initialization:

DO (T-3.3) to (T-3.24) for n=1 onwards

$$\alpha_0(n) = 1.0 \tag{T-3.3}$$

$$e_0(n) = d(n) \tag{T-3.4}$$

$$\underline{f}_0(n) = \underline{b}_0(n) = \left[x(n),\, x^2(n)\right]^T \tag{T-3.5}$$

$$f_0^{(j)}(n) = x(n)\, x(n-j+2), \quad j = 3, \ldots, N+1 \tag{T-3.6}$$

$$r^f_0(n) = r^b_0(n) = \lambda r^f_0(n-1) + \underline{f}_0(n)\, \underline{f}^T_0(n) \tag{T-3.7}$$

DO (T-3.8) to (T-3.22) for i=1, 2, ..., N-1

$$\alpha_i(n) = \alpha_{i-1}(n) - \underline{b}^T_{i-1}(n)\, r^{-b}_{i-1}(n)\, \underline{b}_{i-1}(n) \tag{T-3.8}$$

$$k^f_i(n) = k^f_i(n-1) + \frac{r^{-b}_{i-1}(n-1)\underline{b}_{i-1}(n-1)}{\alpha_{i-1}(n-1)} \left[\underline{f}^T_{i-1}(n) - \underline{b}^T_{i-1}(n-1)\, k^f_i(n-1)\right] \tag{T-3.9}$$

$$k^b_i(n) = k^b_i(n-1) + \frac{r^{-f}_{i-1}(n)\underline{f}_{i-1}(n)}{\alpha_{i-1}(n-1)} \left[\underline{b}^T_{i-1}(n-1) - \underline{f}^T_{i-1}(n) k^b_i(n-1)\right] \tag{T-3.10}$$

$$\overline{\underline{f}}_i(n) = \underline{f}_{i-1}(n) - k^{f^T}_i(n)\, \underline{b}_{i-1}(n-1) \tag{T-3.11}$$

$$\overline{\underline{b}}_i(n) = \underline{b}_{i-1}(n-1) - k^{b^T}_i(n)\, \underline{f}_{i-1}(n) \tag{T-3.12}$$

$$k^y_i(n) = k^y_i(n-1) + \frac{r^{-b}_{i-1}(n)\, \underline{b}_{i-1}(n)}{\alpha_{i-1}(n)} \left[e_{i-1}(n) - \underline{b}^T_{i-1}(n) k^y_i(n-1)\right] \tag{T-3.13}$$

$$e_i(n) = e_{i-1}(n) - k^{y^T}_i(n)\, \underline{b}_{i-1}(n) \tag{T-3.14}$$

DO (T-3.15) to (T-3.16) for j=1+2,..., N+1

$$k^{f(j)}_i(n) = k^{f(j)}_i(n-1) + \frac{r^{-b}_{i-1}(n)\, \underline{b}_{i-1}(n-1)}{\alpha_{i-1}(n-1)} \left[f^{(j)}_{i-1}(n) - \underline{b}^T_{i-1}(n-1) k^{f(j)}_i(n-1)\right] \tag{T-3.15}$$

$$f^{(j)}_i(n) = f^{(j)}_{i-1}(n) - k^{f(j)^T}_i(n)\, \underline{b}_{i-1}(n-1) \tag{T-3.16}$$

$$k^{b(i+2)}_i(n) = k^{b(1+2)}_i(n-1) + \frac{r^{-f}_{i-1}(n)\underline{f}_{i-1}(n)}{\alpha_{i-1}(n-1)} \left[f^{(i+2)}_{i-1}(n) - \underline{f}^T_{i-1}(n) k^{b(i+2)}_i(n-1)\right] \tag{T-3.17}$$

$$b^{(i+2)}_i(n) = f^{(i+2)}_{i-1}(n) - K^{b(i+2)^T}_i(n)\underline{f}_{i-1}(n) \tag{T-3.18}$$

$$\underline{f}_i(n) = \begin{bmatrix} \overline{f}_i(n) \\ f^{(\overline{1}+2)}_i(n) \end{bmatrix} \tag{T-3.19}$$

$$\underline{b}_i(n) = \begin{bmatrix} \overline{b}_i(n) \\ b^{(i+2)}_i(n) \end{bmatrix} \tag{T-3.20}$$

$$r^{-f}_i(n) = \lambda^{-1} r^{-f}_i(n-1) - \frac{\lambda^{-2} r^{-f}_i(n-1)\, \underline{f}_i(n)\, \underline{f}^T_i(n) r^{-f}_i(n-1)}{\alpha_i(n-1) + \lambda^{-1} f^T_i(n) r^{-f}_i(n-1)\underline{f}_i(n)} \tag{T-3.21}$$

$$r^{-b}_i(n) = \lambda^{-1} r^{-b}_i(n-1) - \frac{\lambda^{-2} r^{-b}_i(n-1)\underline{b}_i(n)\underline{b}^T_i(n) r^{-b}_i(n-1)}{\alpha_i(n) + \lambda^{-1}\underline{b}^T_i(n) r^{-b}_i(n-1)\underline{b}_i(n)} \tag{T-3.22}$$

$$k^y_N(n) = k^y_N(n-1) + \frac{r^{-b}_{N-1}(n)\underline{b}_{N-1}(n)}{\alpha_{N-1}(n)} \left[e_{N-1}(n) - \underline{b}^T_{N-1}(n) k^y_N(n-1)\right] \tag{T-3.23}$$

$$e_N(n) = e_{N-1}(n) - k^{y^T}_N(n)\, \underline{b}_{N-1}(n) \tag{T-3.24}$$

adaptive algorithms that can be implemented using only orthogonal transformations such as Givens rotations [3], [5], [16], [19], [25], [26], [29], [37]. The key idea is that every operation in these algorithms is implemented using numerically stable orthogonal transformations and therefore one could expect that the adaptive algorithms are also numerically stable.

QR-decomposition of a matrix involves the application of an orthogonal transformation Q to a matrix A such that

$$QA = \begin{bmatrix} R \\ \underline{0} \end{bmatrix} \tag{23}$$

where R is a triangular matrix and $\underline{0}$ is a matrix of all zero elements. There are basically two methods of deriving

QRD-based fast RLS algorithms. The first involves the QR decomposition of the underlying data matrix. This is the method that has been adopted by Cioffi [5], Bellanger [3], Proudler *et al.* [25], [26]. The second method involves a direct transformation of the least-squares lattice algorithm to the QR-RLS algorithm by using the Cholesky factorization of the estimation error covariances. This method has been adopted by Lewis [16] and Yang and Bohme [37]. The method presented in [37] is quite straightforward and elegant. We have used this method in transforming the conventional LS adaptive lattice Volterra filtering algorithm of Table III into a QRD-based adaptive algorithm.The basic idea behind the derivation is the observation that the complete set of lattice recursions ((T-3.8)–(T-3.14) and (T-3.21)–(T-3.22) in Table III) can be reformulated by applying two orthogonal transformations to suitably constructed vectors and matrices. We shall give only a brief outline of the derivation and refer the interested reader to [37] for further details. Once again, the key difference between our derivations and those in [16], [37] is the fact that our model requires different numbers of coefficients in different "channels".

Recall that the optimal solution to the RLS adaptive filtering problem is given by

$$\underline{W}_{\text{opt}}(n) = \Omega^{-1}(n)P(n) \tag{24}$$

where $\Omega(n)$ and $P(n)$ are as defined in (7) and (8), respectively. It is easy to show from (7) and (8) that $\Omega(n)$ and $P(n)$ can be recursively updated as follows.

$$\Omega(n) = \lambda\Omega(n-1) + \underline{X}(n)\underline{X}^T(n) \tag{25}$$

and

$$P(n) = \lambda P(n-1) + \underline{X}(n)d(n). \tag{26}$$

In practice, the autocorrelation matrix $\Omega(n)$ is positive definite. Hence, we can express it in terms of its Cholesky factorization

$$\Omega(n) = R^T(n)R(n) \tag{27}$$

where $R(n)$ is upper triangular.

Let

$$S(n) = R^{-T}(n)P(n) \tag{28}$$

and

$$\beta(n) = R^{-T}(n)\underline{X}(n). \tag{29}$$

In the above equations, $(\bullet)^{-T}$ represents the matrix transpose of $(\bullet)^{-1}$. Substituting the above definitions in (24), it follows that

$$\underline{W}_{\text{opt}}(n) = R^{-1}(n)S(n). \tag{30}$$

Similarly, we can express the joint process estimation error as

$$e(n) = d(n) - S^T(n)\beta(n). \tag{31}$$

The transformation of the conventional RLS lattice algorithm to the QR-RLS lattice algorithm is based on the following two properties of orthogonal matrices:

1) Orthogonal matrices are length preserving, i.e., if Q is an orthogonal matrix,

$$Q^TQ = QQ^T = I \tag{32}$$

and

$$||Q\underline{x}||^2 = ||\underline{x}||^2, \tag{33}$$

where $\underline{x}$ is a vector of appropriate dimension and $|| \bullet ||$ denotes the L_2 norm.

2) Let

$$B_1 = QA_1 \text{ and } B_2 = QA_2. \tag{34}$$

Then

$$B_1^TB_2 = A_1^TA_2. \tag{35}$$

Let $R(n-1)$ be the Cholesky factor of the least-squares autocorrelation matrix $\Omega(n-1)$. Let $Q(n)$ be an orthogonal matrix of appropriate dimensions such that

$$Q(n) = \begin{bmatrix} \sqrt{\lambda}R(n-1) \\ \underline{X}^T(n) \end{bmatrix} = \begin{bmatrix} R(n) \\ \underline{0}^T \end{bmatrix} \tag{36}$$

where $R(n)$ is an upper triangular matrix and $\underline{0}$ is a vector of all zero elements. Essentially, $Q(n)$ annihilates the input vector $\underline{X}(n)$ by rotating it into $\sqrt{\lambda}R(n-1)$. $Q(n)$ consists of a cascade of $K = \frac{(N(N+3)}{2}$ Givens rotations

$$Q(n) = Q_K(n)Q_{K-1}(n)\cdots Q_1(n), \tag{37}$$

where

$$Q_i(n) = \begin{bmatrix} I_{i-1} & \vdots & & \vdots \\ \cdots & \cos\theta_i(n) & \cdots & \sin\theta_i(n) \\ & \vdots & I_{K-i-1} & \vdots \\ & -\sin\theta_i(n) & \cdots & \cos\theta_i(n) \end{bmatrix}, \quad i = 1, 2, \ldots, K \tag{38}$$

and $\cos\theta_i(n)$ and $\sin\theta_i(n)$ are selected such that the i-th element of the last row of the product matrix is zero. For example, let $r_i(n)$ and $x_i(n)$ represent the (i,i)-th element and the i-th element of the last row, respectively, of the matrix on which $Q_i(n)$ operates. Then, if we choose

$$\cos\theta_i(n) = \frac{r_i(n)}{\sqrt{r_i^2(n) + x_i^2(n)}} \tag{39}$$

and

$$\sin\theta_i(n) = \frac{x_i(n)}{\sqrt{r_i^2(n) + x_i^2(n)}} \tag{40}$$

$x_i(n)$ will be annihilated by $Q_i(n)$. Premultiplying both sides of (36) by their respective transposes, we find that

$$\begin{aligned} R^T(n)R(n) &= \lambda R^T(n-1)R(n-1) + \underline{X}(n)\underline{X}^T(n) \\ &= \lambda\Omega(n-1) + \underline{X}(n)\underline{X}^T(n) = \Omega(n). \end{aligned} \tag{41}$$

It immediately follows that $R(n)$ is indeed a Cholesky factor of $\Omega(n)$ and therefore the use of $R(n)$ in (36) is consistent with the notation adopted in (27).

Consider the matrix product

$$\begin{aligned} Q(n)&\begin{bmatrix} \sqrt{\lambda}R(n-1) & \sqrt{\lambda}S(n-1) & \underline{0} \\ \underline{X}^T(n) & d(n) & 1 \end{bmatrix} \\ &= \begin{bmatrix} R(n) & S(n) & \beta(n) \\ \underline{0}^T & \tilde{e}(n) & \tilde{\alpha}(n) \end{bmatrix} \end{aligned} \tag{42}$$

TABLE IV
RELATIONSHIPS BETWEEN THE VARIABLES EMPLOYED IN THE QRD-BASED LEAST–SQUARES LATTICE SECOND–ORDER VOLTERRA FILTER AND THE CONVENTIONAL LEAST-SQUARES LATTICE VOLTERRA FILTER.

$\tilde{\alpha}_i(n) = \sqrt{\alpha_i(n)}$	$\Omega^f_{i-1}(n) = R^{f^T}_i(n) R^f_i(n)$	$s^f_i(n) = R^{f^T}_i(n) k_{i-1}(n)$
$\bar{e}_i(n) = \frac{e_i(n)}{\tilde{\alpha}_i(n)}$	$\Omega^b_{i-1}(n-1) = R^{b^T}_i(n) R^b_i(n)$	$s^b_i(n) = R^{b^T}_i(n) k^T_{i-1}(n)$
$\underline{\bar{f}}_i(n) = \frac{\underline{f}_i(n)}{\tilde{\alpha}_i(n-1)}$	$\beta^f_i(n) = R^{f^T}_i(n) \underline{f}_{i-1}(n)$	$s^x_i(n) = R^{b^T}_i(n) k^x_{i-1}(n-1)$
$\underline{\tilde{b}}_i(n) = \frac{\underline{b}_i(n)}{\tilde{\alpha}_i(n)}$	$\beta^b_i(n) = R^{b^T}_i(n) \underline{b}_{i-1}(n-1)$	

Premultiplying both sides of (42) with their respective transposes yields several important results.

1) $R^T(n)S(n) = \lambda R^T(n-1)S(n-1) + \underline{X}(n)d(n)$
$$= \lambda P(n-1) + \underline{X}(n)d(n) = P(n). \quad (43)$$

2) $R^T(n)\beta(n) = \underline{X}(n) \quad (44)$

3) $S^T(n)\beta(n) + \tilde{e}(n)\tilde{\alpha}(n) = d(n) = S^T(n)\beta(n) + e(n) \quad (45)$

Also, it can be shown [8] that $\tilde{\alpha}(n)$ and the likelihood variable $\alpha(n)$ are related as.

$$\tilde{\alpha}(n) = \sqrt{\alpha(n)}. \quad (46)$$

Direct evaluation of $\tilde{\alpha}(n)$, after writing $Q(n)$ as a product of $\frac{N(N+3)}{2}$ matrices in (37) and noting the structure of each individual matrix, will show that

$$\tilde{\alpha}(n) = \prod_{i=1}^{N(N+3)/2} \cos\theta_i(n) \quad (47)$$

where $\theta_i(n)$ is the angle that defines the i-th Givens rotation. Also, from (45),

$$\tilde{e}(n) = \frac{e(n)}{\tilde{\alpha}(n)}. \quad (48)$$

The update strategy for a general estimation problem should be clear from the above derivations. Assuming that we have $R(n-1), S(n-1)$, and $\beta(n-1)$ available to us at time $n-1$, the procedure consists of the following operations:

1) Find the appropriate set of orthogonal rotations $Q(n)$ that zeros out the vector $\underline{X}^T(n)$ that contains all the "new" input data samples at time n into the rows of $\sqrt{\lambda}R(n-1)$. The resulting triangular matrix is $R(n)$.
2) Rotate $d(n)$ (the new sample of the desired response signal at time n) into $\sqrt{\lambda}S(n-1)$ using the same $Q(n)$ obtained in part 1. The bottom element is $\tilde{e}(n)$ and the rest of the elements will form the vector $S(n)$.
3) Evaluate $\tilde{\alpha}(n)$ as in (47). The estimation error $e(n)$ can be calculated immediately as in (42).

The derivation of a lattice filter that is implemented solely using Givens rotations can be done by adapting the above approach for solving the forward prediction, backward prediction, and joint process estimation problems associated with each stage. The first task is to define the appropriate variables for each problem and the definitions are tabulated in Table IV. Let $Q^f_i(n)$ define an orthogonal rotation matrix so that it zeroes out $\underline{\tilde{f}}^T_{i-1}(n)$ into the rows of $\sqrt{\lambda}R^f_i(n-1)$. Also, let $Q^b_i(n)$ be another orthogonal rotation matrix that annihilates $\underline{\tilde{b}}^T_{i-1}(n-1)$ into the rows of $\sqrt{\lambda}R^b_i(n-1)$. Then, the following matrix equalities hold.

$$Q^f_i(n)\begin{bmatrix} \sqrt{\lambda}R^f_i(n-1) & \sqrt{\lambda}S^f_i(n-1) & \sqrt{\lambda}S^{f(i+2)}_i(n-1) \\ \underline{\tilde{f}}^T_{i-1}(n) & \underline{\tilde{b}}^T_{i-1}(n-1) & \underline{\tilde{f}}^{(i+2)}_{i-1}(n) \end{bmatrix} = \begin{bmatrix} R^f_i(n) & S^f_i(n) & S^{f(i+2)}_i(n) \\ \underline{0} & \underline{\hat{\tilde{b}}}^T_i(n) & \tilde{b}^{(i+2)}_i(n) \end{bmatrix} \quad (49)$$

The equalities can be easily verified as follows. Premultiplying both sides of (49) by their respective transposes we essentially obtain (T-3.12) and (T-3.18) of Table III. Similarly, (50) essentially leads to (T-3.8), (T-3.11), (T-3.14), and (T-3.16). Equations (49) and (50) completely specify the updated equations for the QRD-based algorithm for adaptive Volterra filtering. The complete algorithm is given in Table VI. An operations count will show that this algorithm requires $\frac{35N^3}{6} + 3N^2 + \frac{295N}{6} - 2$ multiplications and $N^2 + 3N$ square roots.

IV. EXPERIMENTAL RESULTS

In this section we present the results of several experiments that were performed to evaluate the performance of the two algorithms presented in the previous sections. In these experiments the adaptive filters were used in the system identification mode. The system that was identified was a

$$Q^b_i(n)\begin{bmatrix} \sqrt{\lambda}R^b_i(n-1) & 0 & \sqrt{\lambda}S^b_i(n-1) & \sqrt{\lambda}S^x_i(n-1) & \sqrt{\lambda}S^{b(j)}_i(n-1) \\ \underline{\tilde{b}}^T_{i-1}(n-1) & \tilde{\alpha}_{i-1}(n-1) & \underline{\tilde{f}}^T_{i-1}(n) & \tilde{e}_{i-1}(n-1) & \tilde{f}^{(j)}_{i-1}(n) \end{bmatrix}$$
$$= \begin{bmatrix} R^b_i(n) & \beta^b_i(n) & S^b_i(n) & S^x_i(n) & s^{b(j)}_i(n) \\ \underline{0} & \tilde{\alpha}_i(n-1) & \underline{\tilde{f}}^{\wedge T}_i(n) & \tilde{e}_i(n-1) & \tilde{f}^{(j)}_i(n) \end{bmatrix} \quad (50)$$

TABLE V
ORD-BASED LEAST-SQUARES LATTICE ADAPTIVE SECOND–ORDER VOLTERRA FILTER

Order Initialization:
DO (T-5.1) to (T-5.11) for n=1 onwards

$$\tilde{\alpha}_0(n-1) = 1.0 \tag{T-5.1}$$

$$\tilde{e}_0(n) = d(n) \tag{T-5.2}$$

$$\underline{\tilde{f}}_0^T(n) = \underline{\tilde{b}}_0^{-T}(n) = [x(n),\ x^2(n)] \tag{T-5.3}$$

$$\tilde{f}_0{}^{(j)} = x(n)x(n-j+2),\ \ j = 3,\dots,N+1 \tag{T-5.4}$$

DO (T-5.5) to (T-5.8) for i=1,..., N-1

$$Q_i^f(n)\begin{bmatrix} \sqrt{\lambda}R_i^f(n-1) & \sqrt{\lambda}S_i^f(n-1) & \sqrt{\lambda}S_i^{f(i+2)}(n-1) \\ \underline{\tilde{f}}_{i-1}{}^T(n) & \underline{\tilde{b}}_{i-1}^T(n-1) & \tilde{f}_{i-1}^{(i+2)}(n) \end{bmatrix} = \begin{bmatrix} R_i^f(n) & S_i^f(n) & S_i^{f(i+2)}(n) \\ 0 & \underline{\hat{\tilde{b}}}_i^T(n) & \tilde{b}_i^{(i+2)}(n) \end{bmatrix} \tag{T-5.5}$$

$$Q_i^b(n)\begin{bmatrix} \sqrt{\lambda}R_i^b(n-1) & 0 & \sqrt{\lambda}S_i^b(n-1) & \sqrt{\lambda}S_i^x(n-1) & \sqrt{\lambda}S_i^{b(j)}(n-1) \\ \tilde{b}_{i-1}^T(n-1) & \tilde{\alpha}_{i-1}(n-1) & \underline{\tilde{f}}_{i-1}^T(n) & \tilde{e}_{i-1}(n-1) & \tilde{f}_{i-1}^{(j)}(n) \end{bmatrix} = \begin{bmatrix} R_i^b(n) & \beta_i^b(n) & S_i^b(n) & S_i^x(n) & S_i^{b(j)}(n) \\ 0 & \tilde{\alpha}_i(n-1) & \underline{\tilde{f}}_i^T(n) & \tilde{e}_i(n-1) & \tilde{f}_i^{(j)}(n) \end{bmatrix},$$

j=i+2,..., N+1 (T-5.6)

$$\underline{\hat{\tilde{f}}}_i^T(N) = \left[\underline{\hat{\tilde{f}}}_i^T(n),\ \tilde{f}_i^{(i+2)}(n)\right] \tag{T-5.7}$$

$$\underline{\hat{\tilde{b}}}_i^T(n) = \left[\underline{\hat{\tilde{b}}}_i^T(n),\ \tilde{b}_i^{(i+2)}(n)\right] \tag{T-5.8}$$

$$Q_N^f(n)\begin{bmatrix} \sqrt{\lambda}R_N^f(n-1) & \sqrt{\lambda}S_N^f(n-1) \\ \underline{\tilde{f}}_{N-1}^T(n) & \underline{\tilde{b}}_{N-1}^T(n-1) \end{bmatrix} = \begin{bmatrix} R_N^f(n) & S_N^f(n) \\ 0 & \underline{\tilde{b}}_N^T(n) \end{bmatrix} \tag{T-5.9}$$

$$Q_N^b(n)\begin{bmatrix} \sqrt{\lambda}R_N^b(n-1) & 0 & \sqrt{\lambda}S_N^b(n-1) & \sqrt{\lambda}S_N^x(n-1) \\ \underline{\tilde{b}}_{N-1}^T & \tilde{\alpha}_{N-1}(n-1) & \underline{\tilde{f}}_{N-1}^T & \tilde{e}_{N-1}(n-1) \end{bmatrix} = \begin{bmatrix} R_N^b(n) & \beta_N^b(n) & S_N^b(n) & S_N^x(n) \\ 0 & \tilde{\alpha}_N(n-1) & \underline{\tilde{f}}_N^T(n) & \tilde{e}_N(n-1) \end{bmatrix} \tag{T-5.10}$$

$$eN(n-1) = \tilde{e}_N(n-1)\tilde{\alpha}_N(n-1) \tag{T-5.11}$$

tenth order Volterra system with 10 linear coefficients and 55 quadratic coefficients. These coefficients are given in Table VI. The adaptive filter had the same number of coefficients as the unknown system. The input signal to the unknown system was colored noise obtained by filtering a pseudo-Gaussian white noise sequence with zero mean and variance 0.0248 by an FIR filter with impulse response $h(n)$ given by

$$h(n) = \begin{cases} 0.9045, & n = 0 \\ 1.0, & n = 1 \\ .9045, & n = 2 \\ 0, & \text{otherwise} \end{cases} \tag{51}$$

The parameters of the input signal were chosen such that the output of the unknown system had approximately unit variance. The desired response signal $d(n)$ was obtained by adding a zero-mean white Gaussian noise sequence to the output of the unknown system. The measurement noise was uncorrelated with the input signal. We conducted several experiments using two output signal-to-measurement noise ratios of 20 dB and 30 dB and two weighting factors 0.995 and 0.9975. The results presented are ensemble averages of 50 independent runs of 5000 samples each. Performance evaluations were carried out by plotting the mean-squared value of the a *posteriori error* ($e_N(n)$ in the tables) and the squared norms of the linear and quadratic coefficient error vectors. The linear and quadratic coefficients were obtained by converting the lattice reflection coefficients to the equivalent transversal filter coefficients, details of which can be found in [2] and [17]. In order to evaluate the effects of finite precision the algorithms were implemented with a fixed number of bits for both the integer and fractional parts. The number of bits for the integer part was the minimum required to avoid almost all of the overflows for a given number of fractional bits. The finite precision effects were evaluated by calculating the difference between the joint process estimation error which was computed using the maximum precision available in the

TABLE VI
LINEAR AND QUADRATIC COEFFICIENTS OF THE UNKNOWN SYSTEM USED IN THE SYSTEM IDENTIFICATION EXPERIMENTS

	a_i	$b_{i,j}$									
i	$j=$	0	1	2	3	4	5	6	7	8	9
0	-0.052	1.020	-1.812	-1.138	-0.592	-0.144	-0.966	-1.454	1.820	-4.022	5.562
1	0.723		1.389	-2.608	-1.486	-1.382	-2.308	4.256	0.626	-0.264	2.890
2	0.435			-0.635	-0.468	-1.508	0.812	1.284	1.580	-1.800	0.748
3	-0.196				-1.044	0.536	-2.092	-0.774	-3.314	-0.348	0.272
4	-0.143					0.011	2.918	0.698	0.752	-3.496	0.460
5	0.812						-0.987	3.940	2.926	-0.508	1.648
6	0.354							0.198	-0.362	-2.402	1.646
7	0.077								-1.732	-1.334	-3.070
8	-1.379									0.860	0.648
9	2.251										0.305

computer and by using a given precision. The long-term behavior of the two algorithms was evaluated by repeating the experiments with a million samples and was found to be similar to that exhibited in the shorter duration experiments.

The plots of the ensemble averaged a posteriori mean-squared error and the norms of the linear and quadratic error coefficients are given in Figs. 2–4. These plots demonstrate the rapid convergence of the two algorithms, which is a characteristic of all RLS algorithms. Both the algorithms appear to be numerically stable, at least for the signal configuration used in the experiments. In Table VII we have given values of the mean-squared difference between the joint process error computed using the maximum precision available in the computer (64-bit floating-point arithmetic with 16-bit exponent) and that obtained using a given precision. These values were obtained by time averaging the mean-squared difference over the last 1000 samples.

The results presented in Figs. 2–4 and Table VII indicate that both the algorithms perform well in a finite precision environment, even though the unknown nonlinear system had a fairly large number of coefficients. The results presented in Table VII also give some idea of the dynamic ranges of the internal variables of the two algorithms. In the case of the conventional lattice Volterra filter, it was observed that when the number of bits for the fractional part was less than 10, the learning curve converged to a value that was greater than the variance of the measurement noise by a factor greater than two. In the case of the QRD-based lattice Volterra filter, it was observed that below 12 bits for the fractional part the learning curve converged to a value that is more than four times the variance of the measurement noise. Comparing the two algorithms, it appears that the conventional lattice algorithm has an edge over the QRD-based lattice filter in numerical accuracy, whereas the QRD-based filter seems to have a slight edge over the conventional lattice filter when it comes to the dynamic range of the internal variables.

In Tables VIII and IX we present the steady-state mean-squared numerical errors at different stages of the conventional RLS and QR-RLS lattice Volterra filters, respectively. From these results we observe that the numerical errors are larger for higher stages of the lattice than the lower stages. This is generally true of all lattice structures [23], [32], [36]. One interesting observation is that the rate of growth of the accumulated numerical error from a lower stage to a higher stage seems to be somewhat larger for the QRD-based algorithm than for the conventional lattice filter. Further analysis must be done before it can be verified that this statement is true in general.

TABLE VII
VALUES OF THE TIME AVERAGED MEAN-SQUARED DIFFERENCE BETWEEN THE "INFINITE" PRECISION IMPLEMENTATION AND THE IMPLEMENTATION WITH A FIXED NUMBER OF BITS FOR BOTH THE INTEGER AND FRACTIONAL PARTS

QR Lattice Volterra Filter		Conventional Lattice Volterra Filter	
λ=0.995, SNR=20 dB			
Int./Frac. Bits	Mean-Squared Numerical Error	Bits	Mean-Squared Numerical Error
6-14	5.87×10^{-5}	7-12	4.92×10^{-4}
6-16	2.01×10^{-6}	7-14	3.20×10^{-5}
6-18	1.00×10^{-7}	8-16	2.09×10^{-6}
λ=0.9975, SNR=20 dB			
6-14	1.58×10^{-4}	6-10	7.56×10^{-3}
7-16	3.44×10^{-6}	6-14	6.94×10^{-5}
7-18	1.34×10^{-7}	6-16	2.84×10^{-6}
λ=0.995, SNR=30 dB			
Int./Frac. Bits	Mean-Squared Numerical Error	Bits	Mean-Squared Numerical Error
6-14	5.79×10^{-5}	7-12	4.86×10^{-4}
6-16	1.98×10^{-6}	7-14	3.16×10^{-5}
6-18	9.92×10^{-8}	8-16	2.06×10^{-6}
λ=0.9975, SNR=30 dB			
Int./Frac. Bits	Mean-Squared Numerical Error	Bits	Mean-Squared Numerical Error
6-14	1.57×10^{-4}	6-10	7.46×10^{-3}
7-16	3.43×10^{-6}	6-14	7.26×10^{-5}
7-18	1.29×10^{-7}	6-16	2.83×10^{-6}

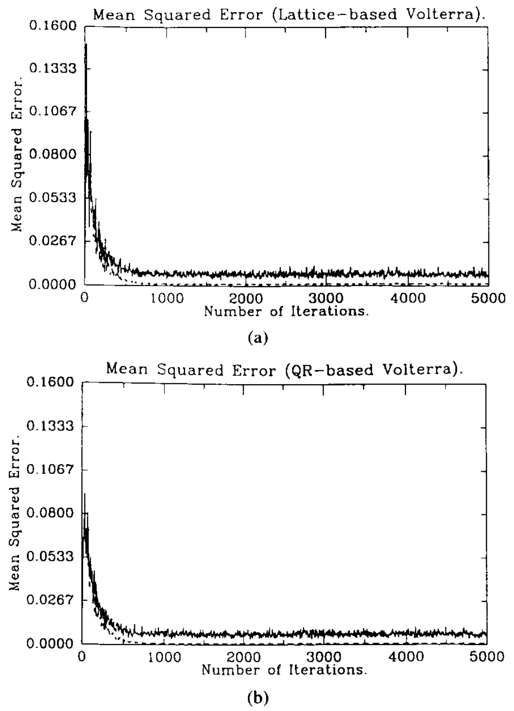

Fig. 2. Learning curves for conventional and QR lattice adaptive Volterra filters. Weighting factor = 0.995. Number of bits for integer part = 7 for conventional lattice and 6 for QR lattice. Number of bits for fractional part = 12 for conventional lattice and 14 for QR lattice. Solid curve: 20 dB measurement noise. Dashed curve: 30 dB measurement noise.

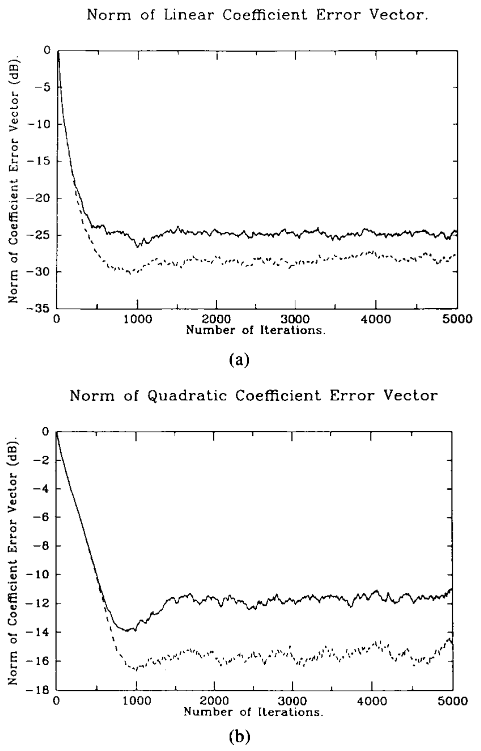

Fig. 3. Norm of coefficient error vector for conventional lattice adaptive Volterra filter. Weighting factor = 0.995. Numberof bits for integer part = 7 and number of bits for fractional part = 12. Solid curve: 20 dB measurement noise. Dashed curve: 30 dB measurement noise.

V. Conclusion

In this paper we presented a lattice structure for second-order Volterra systems. The structure is different from most previously published lattice Volterra structures in that it is applicable to arbitrary planes of support of the Volterra kernels and arbitrary input signals. Computationally efficient conventional RLS lattice and QR-RLS lattice adaptive algorithms based on this structure were also presented. These algorithms share the fast convergence property of fast RLS transversal Volterra filters without suffering from problems of numerical instability. Both the algorithms appear to be numerically robust under finite precision conditions. The conventional lattice Volterra filter appears to be numerically more accurate than the QR-based Volterra filter; however, the QR-based Volterra filter seems to have a slight edge in terms of the dynamics of the internal variables. Both algorithms can be easily extended to higher-order nonlinearities. Also, both the algorithms are amenable to parallel implementations. A theoretical finite-precision error analysis of the RLS and QR-RLS Volterra lattice could be highly complex; however, one could expect that the numerical properties of the nonlinear lattice filters would be similar to their linear counterparts and analysis of the numerical properties of such filters [21], [28], [36] have shown that they indeed possess the good finite precision characteristics that were demonstrated in the experimental results presented.

Appendix A

Let

$$x^f_m(n) = \left[x(n), x^2(n), x(n)x(n-1), \dots, x(n)x(n-m)\right]^T \quad \text{(A-1)}$$

and

$$x^b_m(n) = \left[x(n-m), x^2(n-m), x(n-m+1)\dots, x(n)x(n-m)\right]^T \quad \text{(A-2)}$$

For $m = 0$, we will define these two vectors as

$$x^f_0(n) = x^b_0(n) = \left[x(n), x^2(n)\right]^T. \quad \text{(A-3)}$$

Clearly, the above vectors can be partitioned as

$$x^f_m(n) = \left[\frac{x^f_{m-1}(n)}{x(n)x(n-m)}\right]; \quad m = 1, 2\dots, N-1 \quad \text{(A-4)}$$

and

$$x^b_m(n) = \left[\frac{x^b_{m-1}(n-1)}{x(n)x(n-m)}\right]; \quad m = 1, 2, \dots, N-1. \quad \text{(A-5)}$$

The above partitionings are useful in deriving the order update equations.

The m-th order backward prediction error vector $\underline{b}_m(n)$ is defined as the optimal LS error in estimating $x^b_m(n)$ using $x^b_0(n), \dots, x^b_{m-1}(n)$, where $\underline{b}_0(n) = x_0(n)$. Similarly, the m-th order forward prediction error is defined as the optimal

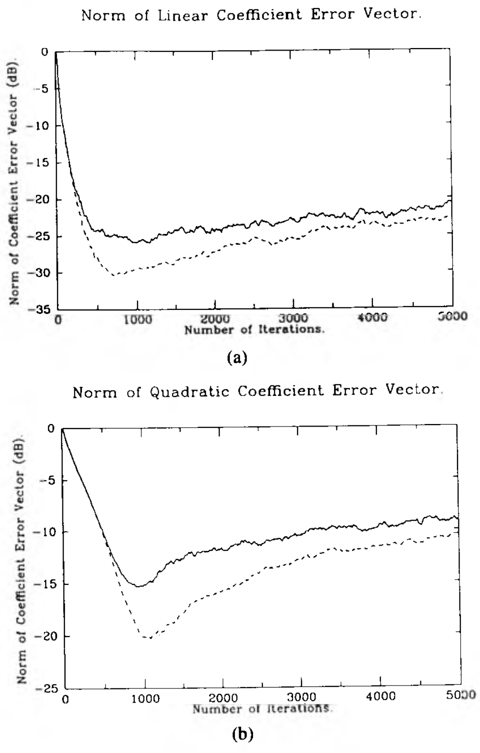

Fig. 4. Norm of coefficient error vector for QR lattice adaptive Volterra filter. Weighting factor = 0.995. Number of bits for integer part = 6 and number of bits for fractional part = 14. Solid curve: 20 dB measurement noise. Dashed curve: 30 dB measurement noise.

TABLE VIII
STEADY—STATE MEAN—SQUARED VALUES OF THE NUMERICAL ERRORS AT DIFFERENT STAGES OF THE CONVENTIONAL RLS VOLTERRA LATTICE FILTER.

Int/Frac Bits	Stage 1	Stage 5	Stage 10
λ=0.995, SNR=20 DB			
7-12	1.02×10^{-4}	1.73×10^{-4}	4.92×10^{-4}
7-14	4.02×10^{-7}	4.81×10^{-6}	3.20×10^{-5}
8-16	2.68×10^{-8}	3.19×10^{-7}	2.09×10^{-6}
λ=0.9975, SNR=20 DB			
6-10	6.62×10^{-4}	2.53×10^{-3}	7.56×10^{-3}
6-14	2.71×10^{-5}	3.82×10^{-5}	6.94×10^{-5}
6-16	3.27×10^{-7}	7.95×10^{-7}	2.84×10^{-6}
λ=0.995, SNR=30 DB			
7-12	1.00×10^{-4}	1.73×10^{-4}	4.86×10^{-4}
7-14	3.91×10^{-8}	4.80×10^{-6}	3.16×10^{-5}
8-16	2.63×10^{-8}	3.18×10^{-7}	2.06×10^{-6}
λ=0.9975, SNR=30 DB			
6-10	6.58×10^{-4}	2.51×10^{-3}	7.46×10^{-3}
6-14	2.70×10^{-5}	3.85×10^{-5}	7.26×10^{-5}
6-16	3.28×10^{-7}	7.95×10^{-7}	2.83×10^{-6}

LS error in estimating $x_m^f(n)$ using $x_0^f(n-m), x_1^f(n-m+1), \ldots, x_{m-1}^f(n-1)$. As is well known, the backward prediction error vectors, $\underline{b}_m(n), m = 0, 1, \ldots, N-1$ form an orthogonal basis set for the vector space spanned by the input vectors $x_i(n), i = 0, 1, \ldots, N-1$. Hence the desired signal $y(n)$ can be estimated as a linear combination of the orthogonal backward prediction error vectors.

Similar to (A-4) and (A-5) we can partition the prediction error vectors as

$$\underline{f}_m(n) = \begin{bmatrix} \bar{\underline{f}}_m(n) \\ \hline f_m^{(m)}(n) \end{bmatrix} \quad \text{(A-6)}$$

$$\underline{b}_m(n) = \begin{bmatrix} \bar{\underline{b}}_m(n) \\ \hline b_m^{(m)}(n) \end{bmatrix} \quad \text{(A-7)}$$

where $\bar{\underline{f}}_m(n)$ and the $\bar{\underline{b}}_m(n)$ are the error vectors in estimating $x_{m-1}^f(n)$ and $x_{m-1}^b(n-1)$, respectively. Similarly, $f_m^{(m)}(n)$ and $b_m^{(m)}(n)$ are the errors in estimating $x(n)x(n-m)$ using the elements of the sets $\{x_0^f(n-1), x_1^f(n-2), \ldots, x_{m-1}^f(n-m)\}$ and $\{x_0^b(n), x_1^b)(n), \ldots, x_{m-1}^b(n)\}$, respectively. Now, $\underline{f}_{m-1}(n)$ is the error in predicting $x_{m-1}^f(n)$ using $x_0^b(n-1), x_1^b(n-1), \ldots, x_{m-2}^b(n-1)$ and $\underline{b}_{m-1}(n-1)$ is the error in predicting $x_{m-1}^b(n-1)$ using $x_0^b(n-1), x_1^b(n-1), \ldots, x_{m-2}^b(n-1)$. Hence, $\bar{\underline{f}}_m(n)$, which is the error in predicting $x_{m-1}^f(n)$ using $x_0^b(n-1), x_1^b(n-1), \ldots, x_{m-2}^b(n-1), x_{m-1}^b(n-1)$ can be computed as

$$\bar{\underline{f}}_m(n) = \underline{f}_{m-1}(n) - k_m^{f^T}(n)\underline{b}_{m-1}(n-1) \quad \text{(A-8)}$$

because $\underline{b}_{m-1}(n-1)$ is orthogonal to the space spanned by $x_0^b(n-1), x_1^b(n-1), \ldots, x_{m-2}^b(n-1)$. $k_m^f(n)$ is the LS optimal coefficient matrix in estimating $\underline{f}_{m-1}(n)$ using $\underline{b}_{m-1}(n-1)$ and can be computed as

$$k_m^f(n) = r_{m-1}^{-b}(n-1)\Delta_m(n) \quad \text{(A-9)}$$

where $r_{m-1}^b(n-1)$ is the LS autocorrelation matrix of the backward prediction error vector $\underline{b}_{m-1,n}(k)$ and $\Delta_m(n)$ is the LS crosscorrelation matrix of $\underline{b}_{m-1,n}(k)$ and $\underline{f}_{m-1,n}(k)$. Similarly, one can show that

$$\bar{\underline{b}}_m(n) = \underline{b}_{m-1}(n-1) - k_m^{b^T}(n)\underline{f}_{m-1}(n) \quad \text{(A-10)}$$

$$f_m^{(m)}(n) = f_{m-1}^{(m)}(n) - k_m^{f(m)^T}(n)\underline{b}_{m-1}(n-1) \quad \text{(A-11)}$$

$$b_m^{(m)}(n) = f_{m-1}^{(m)}(n) - k_m^{b(m)^T}(n)\underline{f}_{m-1}(n) \quad \text{(A-12)}$$

where $f_m^{(m)}(n)$ and $b_m^{(m)}(n)$ are auxiliary estimation errors that have to be computed outside the basic lattice structure. It can be shown that $f_j^{(m)}(n), j = 1, 2, \ldots, m-1$, can be recursively updated as

$$f_j^{(m)}(n) = f_{j-1}^{(m)}(n) - k_j^{f(m)^T}(n)\underline{b}_{j-1}(n-1) \quad \text{(A-13)}$$

where $f_0^{(m)}(n) = x_m(n)$.

TABLE IX
STEADY—STATE MEAN—SQUARED VALUES OF THE NUMERICAL ERRORS AT DIFFERENT STAGES OF THE CONVENTIONAL QR-RLS VOLTERRA LATTICE FILTER.

λ=0.995, SNR=20 DB			
Int/Frac Bits	Stage 1	Stage 5	Stage 10
6-14	1.63×10^{-8}	8.67×10^{-6}	5.87×10^{-5}
6-16	1.55×10^{-9}	3.23×10^{-7}	2.01×10^{-6}
6-18	6.96×10^{-10}	2.03×10^{-8}	1.00×10^{-7}
λ=0.9975, SNR=20 DB			
Int/Frac Bits	Stage 1	Stage 5	Stage 10
6-14	2.04×10^{-6}	2.53×10^{-5}	1.58×10^{-4}
7-16	2.13×10^{-9}	4.90×10^{-7}	3.44×10^{-6}
7-18	3.28×10^{-10}	1.98×10^{-8}	1.30×10^{-7}
λ=0.995, SNR=30 DB			
Int/Frac Bits	Stage 1	Stage 5	Stage 10
6-14	1.58×10^{-8}	8.69×10^{-6}	5.79×10^{-5}
6-16	1.54×10^{-9}	3.22×10^{-7}	1.98×10^{-6}
6-18	6.91×10^{-10}	2.01×10^{-8}	9.92×10^{-8}
λ=0.9975, SNR=30 DB			
Int/Frac Bits	Stage 1	Stage 5	Stage 10
6.14	2.02×10^{-6}	2.53×10^{-5}	1.57×10^{-4}
7-16	2.08×10^{-9}	4.88×10^{-7}	3.43×10^{-6}
7-18	3.31×10^{-10}	1.97×10^{-8}	1.29×10^{-7}

Since the backward prediction error vectors $\underline{b}_0(n)$, $\underline{b}_1(n)$, $\ldots$, $\underline{b}_{N-1}(n)$ span the same space as the elements of the input matrix $\underline{X}_N(n)$, the joint process estimation error $e_m(n)$ can be recursively computed as

$$\begin{aligned} e_m(n) &= y(n) - \sum_{j=1}^{m} k_j^{y^T}(n)\underline{b}_{j-1}(n) \qquad \text{(A-14)} \\ &= e_{m-1}(n) - k_m^{y^T}(n)\underline{b}_{m-1}(n), \end{aligned}$$

where $k_m^y(n)$ is the LS optimal coefficient vector in estimating $e_{m-1}(n)$ using $\underline{b}_{m-1}(n), e_0(n) = y(n)$. The rest of the derivations are very similar to the above and omitted here.

REFERENCES

[1] O. Agazzi and D. Hodges, "Nonlinear echo cancellation of data signals," *IEEE Trans. Commun.*, vol. COM-30, pp. 2421–2433, Nov. 1982.
[2] H. K. Baik and V. J. Mathews, "Adaptive lattice bilinear filters," *IEEE Trans. Sig. Proc.*, vol. 41, no. 6, pp. 2033–2046, June 1993.
[3] M. G. Bellanger, "The FLS-QR algorithm for adaptive filtering," *Sig. Proc.*, vol. 17, pp. 291–304, 1989.
[4] S. Benedetto *et al.*, *Digital Transmission Theory*, New Jersey: Prentice-Hall, 1987.
[5] J. M. Cioffi, "The fast adaptive rotors RLS algorithm," *IEEE Trans. Signal Proc.*, vol. 38, pp. 631–653, April 1990.
[6] M. J. Coker and D. N. Simkins, "A nonlinear adaptive noise canceller," *Proc. ICASSP*, pp. 470–473, 1980.
[7] G. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore: The Johns Hopkins University Press, 1983.
[8] Simon Haykin, *Adaptive Filter Theory*, New Jersey: Prentice-Hall, 1985.
[9] I. W. Hunter and M. J. Korenberg, "The identification of nonlinear biological systems: wiener and hammerstein cascade models," *Biological Cybernetics*, vol. 55, pp. 135–144, 1986.
[10] N. Kalouptsidis *et al.*, "A family of computationally efficient algorithms for multichannel signal processing," *Sig. Proc.*, vol. 5, no. 1, pp. 5–19, Jan. 1983.
[11] T. Koh and E. J. Powers, "An adaptive nonlinear digital filter with lattice orthogonalization," *Proc. ICASSP*, Boston, April 1983, pp. 37–40.
[12] T. Koh and E. J. Powers, "Second order volterra filtering and its applications to nonlinear system identification," *IEEE Trans. on Acoust., Speech and Sig. Proc.*, vol. ASSP-33, no. 6, pp. 1445–1455, Dec. 1985.
[13] M. J. Korenberg and I. W. Hunter, "The identification of nonlinear biological systems: LNL cascade models," *Bio. Cyber.*, vol. 55, pp. 125–134, 1986.
[14] J. Lee and V. J. Mathews, "A fast recursive least squares second order adaptive volterra filter and its performance analysis," *IEEE Trans. on Sig. Proc.*, vol. 41, no. 3, pp. 1087–1102, Mar. 1993.
[15] P. J. Lenk and S. R. Parker, "Nonlinear modeling by discrete orthogonal lattice structure," *Proc. IEEE Int. Symp. Circ. and Syst.*, 1986.
[16] P. S. Lewis, *Algorithms and Architectures for Adaptive Least Squares Signal Processing, with Applications in Magnetoencephalography, Ph.D. Thesis*, University of Southern California, 1988.
[17] F. Ling and J. Proakis, "A generalized multichannel least-squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acous., Speech, and Sig. Proc.*, vol. ASSP-32, no. 2, pp. 381–389, April 1984.
[18] Fuyun Ling, "Rapidly convergent adaptive filtering algorithms for adaptive equalization and channel estimation," *Ph.D. Thesis, Dept. Elec. and Comput. Eng.*, Northeastern University, Boston, MA, 1984.
[19] Fuyun Ling, "Givens rotation based least-squares lattice and related algorithms," *IEEE Trans. on Sig. Proc.*, vol. 39, no. 7, pp. 1541–1551, July 1991.
[20] F. Ling *et al.*, "Numerically robust least squares lattice-ladder algorithms with direct updating of the reflection coefficients," *IEEE Trans. Acous., Speech, Sig. Proc.*, vol. ASSP-34, no. 4, pp. 837–845, Aug. 1986.
[21] S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, pp. 157–167, 1985.
[22] V. J. Mathews, "Adaptive polynomial filters," *IEEE Sig. Proc. Mag.*, vol. 8, no. 3, pp. 10–26, July 1991.
[23] V. J. Mathews and Z. Xie, "Fixed-point error analysis of stochastic gradient adaptive lattice filters," *IEEE Trans. Acoust., Speech, and Sig. Proc.*, vol. 38, no. 1, pp. 70–80, Jan. 1990.
[24] 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.
[25] I. K. Proudler *et al.*, "Fast QRD-based algorithms for least squares linear prediction," *Proc. SPIE: Advanced Algorithms and Architec. for Sig. Proc.*, vol. 1152, San Diego, California, August 1989.
[26] I. K. Proudler *et al.*, "Fast QRD-based algorithms for least squares linear prediction," *Proc. IMA Math. in Sig. Proc. Conf.*, Warwick, England, Dec. 1988.
[27] G. Ramponi and G. L. Sicuranza, "Decision-directed nonlinear filters for image processing," *Electronic Lett.*, vol. 23, no. 23, pp. 1218–1219, Nov. 1987.
[28] P. A. Regalia, "Numerical stability properties of a QR-based fast least squares algorithm," *IEEE Trans. Sig. Proc.*, vol. 41, no. 6, pp. 2096–2109, June 1993.
[29] P. A. Regalia and M. G. Bellanger, "On the duality between fast qr methods and lattice methods in least-squares adaptive filtering," *IEEE Trans. on Sig. Proc.*, vol. 89, no. 4, pp. 879–891, April 1991.
[30] W. Reiss, "Nonlinear distortion analysis of P-I-N diode attenuators using volterra series representation," *IEEE Trans. Circ. and Sys.*, vol. CAS-31, no. 6, pp. 535–542, June 1984.
[31] W. J. Rugh, *Nonlinear System Theory*, Baltimore: Johns Hopkins University Press, 1981.
[32] C. Samson and V. U. Reddy, "Fixed-point error analysis of the normalized ladder algorithm," *IEEE Trans. Acoustics, Speech, and Sig. Proc.*, vol. ASSP-31, no. 5, pp. 11-77-1191, 1983.
[33] M. Schetzen, *The Volterra and Wiener Theory of the Nonlinear Systems*, New York: John Wiley, 1980.
[34] G. L. Sicuranza *et al.*, "Adaptive echo cancellation with nonlinear digital filters," *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, San Diego, CA, pp. 3.10.1-3.10.4, Mar. 1984.
[35] M. J. Smith *et al.*, "Nonlinear echo cancellers based on transpose distributed arithmetic," *IEEE Trans. on Circ. and Syst.*, vol. CAS-35, no. 1, pp. 6–18, Jan. 1988.
[36] M. A. Syed and V. J. Mathews, "Finite precision error analysis of a QR-decomposition-based lattice predictor," *J. Optical Eng.*, vol. 31, no. 6, pp. 1170–1180, June 1992.
[37] B. Yang and J. F. Bohme, "Rotation based RLS algorithms: unified derivations, numerical properties and parallel implementations," *IEEE Trans. on Sig. Proc.*, vol. 40, no. 5, pp. 1151–1167, May 1992.

[38] J. Zarzycki, *Nonlinear Prediction Ladder Filters for Higher Order Stochastic Sequences*, Berlin: Springer-Verlag, 1985.

Mushtaq A. Syed (Photo not available at the time of publication.) received the B.E. degree in instrumentation and control from the University of Poona, the M.Sc. degrees in biomedical engineering from the University of Saskatchewan, Canada, and the M.E. and Ph.D. degree in electrical engineering from the University of Utah.

At present, he is working as a DSP Engineer with Digicom Systems, Inc., of Milpitas, Calif. His research interests include nonlinear adaptive signal processing, image processing, and digital communications.

V. John Mathews (M'84–M'90) (Photo not available at the time of publication.) received the B.E. (Hons.) degree in electronics and communication engineering from the University of Madras, India, in 1980, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Iowa, Iowa City, in 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. Since 1985, he has been with the Department of Electrical Engineering, University of Utah, Salt Lake City, where he is now an Associate Professor. His research interests include adaptive filtering, spectrum estimation, and data compression.

Dr. Mathews is a past Associate Editor of the *IEEE Transactions on Signal Processing* and is currently an Associate Editor of *IEEE Signal Processing Letters*. He is a member of the Digital Signal Processing Technical Committee of the IEEE Signal Processing Society.