Lattice and QR Decomposition-Based Algorithms for Recursive Least Squares Adaptive Nonlinear Filters*

Mushtaq A. Syed and V. John Mathews

Department of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112
(801) 581-7869

1 Abstract

This paper presents a lattice structure for adaptive Volterra systems. The stucture is applicable to arbitrary planes of support of the Volterra kernels. A fast least squares lattice and a fast QR-lattice adaptive nonlinear filtering algorithms based on the lattice structure are also presented. These algorithms share the fast convergence property of fast least squares transversal Volterra filters; however, unlike the transversal filters they do not suffer from numerical instability.

1 Introduction

In this paper we present computationally efficient and numerically stable algorithms for recursive least squares adaptive nonlinear filters. The nonlinearity is modeled using a second order Volterra series expansion [11] of the input signal. (There is no loss of generality in the choice of a second order nonlinearity; it merely simplifies the presentation.) Adaptive nonlinear filters where the nonlinearity is modeled using low order Volterra systems have several applications, including channel equalization and noise cancellation in high speed communication systems, modeling biological phenomenon, image processing, and several others.

In the Volterra series representation of systems , which is an extension of linear system theory, the output $y[n]$ of any causal, discrete-time, time-invariant nonlinear system can be represented as a function of the input sequence $x[n]$ using the Volterra series expansion

$$\begin{aligned} y(n) &= h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)x(n-m_1) + \\ &\sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \\ &\cdots + \\ &\sum_{m_1=0}^{\infty}\cdots\sum_{m_p=0}^{\infty} h_p(m_1,m_2,\cdots,m_p)x(n-m_1)\cdots x(n-m_p) \\ &+\cdots , \end{aligned} \quad (1)$$

where $h_p(m_1, m_2, \cdots, m_p)$ is the $p-th$ order Volterra kernel [11] of the system. One can think of the Volterra series expansion as a Taylor series expansion with memory. A nonlinear system that can be represented by a Volterra series is completely represented by its Volterra kernels. Without loss of generality, we will assume that the Volterra kernels are symmetric, i.e., $h_p(m_1, m_2, \cdots, m_p)$ is left unchanged by any of the possible $p!$ interchanges of the arguments $m_1, m_2, \cdots, m_p$.

Possibly because of the extremely complex nature of the nonlinear filters, very little work has been done in adaptively tracking the time varying coefficients of such filters. Most of the work in adaptive Volterra filters is very recent and many of them [3, 5] employ the LMS algorithm or its variations. Unfortunately, such filters have convergence rates that are too slow to be useful in many applications. More recently, Mathews and Lee [8] presented a fast algorithm for recursive least-squares (RLS) adaptive Volterra filters. Their work translated the nonlinear filtering problem into a multichannel filtering problem and then used ideas employed for developing fast multichannel RLS adaptive filters for their derivation. While this algorithm provides good convergence and tracking properties, it, like most other fast transversal adaptive filters, suffers from problems with numerical instability.

*This work was supported in part by the National Science Foundation under Grant MIP-8708970 and by a University of Utah Research Grant

In this paper, we will present two algorithms for fast adaptive least squares lattice second order Volterra filters. The structures presented here can be easily extended to higher order nonlinearities. The first algorithm extends the notion of traditional least squares lattice linear filters to the nonlinear case. The second algorithm is implemented solely using Given's rotations and is therefore expected to exhibit better numerical stability than the first algorithm. This algorithm is similar in its development to the QRD-based fast adaptive RLS linear filters that have been recently developed [2, 9, 10]. We believe that both algorithms for lattice Volterra filtering are novel.

The lattice structure presented in this paper is based on the earlier work by Ling and Proakis [7] on multichannel lattice filters and is different from nonlinear lattice structures available in the literature [5, 6] in the sense that the structure can be applied to Volterra systems with arbitrary input signals and arbitrary shapes for the Volterra kernels. The lattice filter structures in [6] are applicable only to systems with very special shapes for the Volterra kernel and the structure presented in [5] required that the system be Gaussian. The predictor structure in out filter is similar to that in [12].

The rest of the paper is organized as follows: The lattice filter structure for Volterra systems and the first least squares lattice adaptive Volterra filtering algorithm are presented in section 2. The QRD-based adaptive filter is presented in section 3. Finally, the concluding remarks are made in section 4.

2 The Fast RLS Lattice Adaptive Volterra Filter

Consider the problem of recursively estimating the desired signal $d(n)$ as a truncated second order Volterra series expansion in the primary input signal $x(n)$ such that the exponentially weighted sum of squared errors

$$\begin{aligned} \xi_N(n) &= \sum_{k=1}^{n} \lambda^{n-k}(d(k) - \sum_{m_1=0}^{N} \hat{a}_{m_1}(n)x(k-m_1) - \\ &\sum_{m_1=0}^{N}\sum_{m_2=m_1}^{N} \hat{b}_{m_1,m_2}(n)x(k-m_1)x(k-m_2))^2 \end{aligned} \quad (2)$$

is minimized. In equation 2 $\hat{a}_{m_1}(n)$ and $\hat{b}_{m_1,m_2}(n)$ are the linear and quadratic coefficients, respectively, of the second order Volterra filter. Also, $0 < \lambda \leq 1$ is the parameter of the exponential window that controls the rate at which the adaptive filter tracks time varying parameters.

Let us define the input vector X_n (of size $N(N+3)/2$) at time n as

$$X_n = [x(n), x^2(n), x(n-1), x^2(n-1), x(n)x(n-1), \cdots, x(n)x(n-N+1)]^T, \quad (3)$$

where $(\cdot)^T$ denotes matrix transpose of $(\cdot)$. Let us also define the coefficient vector as

$$\underline{W}_n = [\hat{a}_1(n), \hat{b}_{11}(n), \hat{a}_2(n), \hat{b}_{22}(n), \hat{b}_{12}(n), \cdots, \hat{b}_{1\,N-1}(n)]^T . \quad (4)$$

Using the above matrix definitions in equations 3 and 4 we can rewrite equation 2 more compactly as

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

The optimal solution to the problem is given by

$$\underline{W}_{n,opt} = \Omega_n^{-1} P_n \,, \tag{6}$$

where

$$\Omega_n = \sum_{k=0}^{n} \lambda^{n-k} X_k X_k^T \,, \tag{7}$$

and

$$P_n = \sum_{k=0}^{n} \lambda^{n-k} X_k d(k) \,, \tag{8}$$

From the above problem formulation it is relatively easy to see that we can treat the nonlinear least squares estimation problem as a multichannel linear LS estimation problem with $N+1$ channels. The adaptive filter inputs at time n from the $N+1$ channels are $x(n)$, $x^2(n)$, $x(n)x(n-1), \cdots, x(n)x(n-N+1)$. The adaptive filter uses N coefficients for the first two channels that correspond to $x(n)$ and $x^2(n)$, $N-1$ coefficients for the third channel which corresponds to $x(n)x(n-1)$, $N-2$ coefficients for the next channel and so on till it uses a single coefficient for the last channel.

The lattice implementation of a second order Volterra filter with $N-1$ delays has $N-1$ stages in the predictor section. The way in which our lattice structure differs from usual multichannel filters is that the different predictor sections have different number of coefficients. Each predictor stage is a lattice of one dimension greater than the preceding stage. The first stage is a 2-channel lattice stage.

A basic description may be made as follows: The problem that we are interested in is that of estimating the desired signal $d(n)$ as a linear combination of the following samples:

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

The lattice structure first creates a set of N orthogonal vectors $\underline{b}_i(n)$, $i = 0, 1, 2, \cdots, N-1$. These vectors span the linear space spanned by the elements of the input matrix $\underline{X}$. $\underline{b}_i(n)$ is a vector of $i+2$ elements and is obtained as the optimal least squares backward prediction error when the vector

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

is estimated using the elements of columns 0 through $i-1$ of $\underline{X}$. ($\underline{b}_0(n)$ is defined as $[x(n), x^2(n)]^T$) In order to compute $\underline{b}_i(n)$ efficiently we also need to compute a set of N forward prediction error vectors $f_i(n)$, $i = 0, 1, 2, \cdots, N-1$ defined as the $i+2$ element vector obtained as the least squares forward prediction error in estimating the vector

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

using the elements of the matrix

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

Note that at each stage one additional forward prediction error and a backward prediction error, both corresponding to $x(n)x(n-i)$ for the $i-th$ stage, have to be computed since the dimension of each stage is greater than the preceding stage by one. These additional computations have to performed outside the basic lattice predictor. Once the computations of the backward error vectors are performed, the elements of these vectors can be linearly combined to get the best estimate of the desired signal $d(n)$. The complete algorithm is given in Table 1.

Table 1: THE FAST RLS LATTICE ADAPTIVE VOLTERRA FILTER

```
INITIALIZATION
    $\alpha_0(-1) = 1 \,,\ \underline{r}_0^f(0) = \underline{r}^b(0) = \delta\, I_2 \,,\ \delta > 0$
    $\alpha_j(0) = 1 \,,\ j = 0, 1, 2, \cdots, N+1$
    $\underline{b}_0(n) = \underline{f}_0(n) = \underline{Y}(n) = [x(n) \,,\ x^2(n)]^T$
    $e_0(n) = d(n)$
    $\underline{r}_0^f(n) = \underline{r}_0^b(n) = \lambda \underline{r}_0^f(n-1) + \underline{Y}(n)\underline{Y}^T(n)$
    $f_0^{(j)}(n) = x(n)x(n-j+2) \,;\ j = 3, 4, \cdots, N+1$
DO 1 TO 18 FOR $n = 1$ ONWARDS
DO 1 TO 16 FOR $i = 2, 3, \cdots, N$
    $\underline{k}_{i-1}(n) = \lambda \underline{k}_{i-1}(n-1) + \underline{b}_{i-2}(n-1)\underline{f}_{i-2}^T(n)/\alpha_{i-2}(n-1)$   (1)
    $\underline{f}_{i-1}(n) = \underline{f}_{i-2}(n) - \underline{k}_{i-1}(n)\underline{r}_{i-2}^{-b}(n-1)\underline{b}_{i-2}(n-1)$   (2)
    $\underline{b}_{i-1}(n) = \underline{b}_{i-2}(n-1) - \underline{k}_{i-1}(n)\underline{r}_{i-2}^{-f}(n)\underline{f}_{i-2}(n)$   (3)
    $\alpha_{i-1}(n) = \alpha_{i-2}(n) - \underline{b}_{i-2}^T(n)\underline{r}_{i-2}^{-b}(n)\underline{b}_{i-2}(n)$   (4)
    $\underline{r}_{i-1}^f(n) = \lambda \underline{r}_{i-1}^f(n-1) + \underline{f}_{i-1}(n)\underline{f}_{i-1}^T(n)/\alpha_{i-1}(n-1)$   (5)
    $\underline{r}_{i-1}^b(n) = \lambda \underline{r}_{i-1}^b(n-1) + \underline{b}_{i-1}(n)\underline{b}_{i-1}^T(n)/\alpha_{i-1}(n)$   (6)
    $\underline{k}_{i-1}^x(n) = \lambda \underline{k}_{i-1}^x(n-1) + e_{i-2}(n)\underline{b}_{i-2}(n)/\alpha_{i-2}(n)$   (7)
    $e_{i-1}(n) = e_{i-2}(n) - \underline{k}_{i-1}^{x^T}(n)\underline{r}_{i-2}^{-b}(n)\underline{b}_{i-2}(n)$   (8)
DO 9 TO 10 FOR $j = i+1, \cdots, N+1$
    $\hat{\underline{k}}_{f\,i-1}^{(j)}(n) = \lambda \hat{\underline{k}}_{f\,i-1}^{(j)}(n-1) + \underline{b}_{i-2}(n-1) f_{i-2}^{(j)}(n)/\alpha_{i-2}(n-1)$   (9)
    $f_{i-1}^{(j)}(n) = f_{i-2}^{(j)}(n) - \hat{\underline{k}}_{f\,i-1}^{(j)^T}(n)\underline{r}_{i-2}^{-b}(n-1)\underline{b}_{i-2}(n-1)$   (10)
    $\hat{\underline{k}}_{b\,i-1}^{(i+1)}(n) = \lambda \hat{\underline{k}}_{b\,i-1}^{(i+1)}(n-1) + \underline{f}_{i-2}(n) f_{i-2}^{(i+1)}(n)/\alpha_{i-2}(n-1)$   (11)
    $b_{i-1}^{(i+1)}(n) = f_{i-2}^{(i+1)}(n) - \hat{\underline{k}}_{b\,i-1}^{(i+1)^T}(n)\underline{r}_{i-2}^{-f}(n)\underline{f}_{i-2}(n)$   (12)
    $\underline{f}_{i-1}(n) = [\underline{f}_{i-1}^T(n)\ f_{i-1}^{(i+1)}(n)]^T$   (13)
    $\underline{b}_{i-1}(n) = [\underline{b}_{i-1}^T(n)\ b_{i-1}^{(i+1)}(n)]^T$   (14)
    $\underline{r}_{i-1}^f(n) = \lambda \underline{r}_{i-1}^f(n-1) + \underline{f}_{i-1}(n)\underline{f}_{i-1}^T(n)/\alpha_{i-1}(n-1)$   (15)
    $\underline{r}_{i-1}^b(n) = \lambda \underline{r}_{i-1}^b(n-1) + \underline{b}_{i-1}(n)\underline{b}_{i-1}^T(n)/\alpha_{i-1}(n)$   (16)
    $\underline{k}_N^x(n) = \lambda \underline{k}_N^x(n-1) + e_{N-1}(n)\underline{b}_{N-1}(n)/\alpha_{N-1}(n)$   (17)
    $e_N(n) = e_{N-1}(n) - \underline{k}_N^{x^T}(n)\underline{r}_{N-1}^{-b}(n)\underline{b}_{N-1}(n)$   (18)
```

3 QR-Decomposition Based Adaptive Lattice Volterra Filter

In the previous section we presented an adaptive lattice second order Volterra filter. The filter consisted of a cascade of lattice sections with increasing dimension and the first section being a 2-channel lattice stage.

Cioffi [2] recently introduced a fast QR-Decomposition (QRD) based algorithms for RLS linear adaptive filtering. The key idea is that every operation in these algorithms is implemented using numerically stable Given's rotations and therefore the adaptive filtering algorithms are guaranteed to be numerically stable. Furthermore, the algorithms are amenable to implementation using array architectures. Following up on Cioffi's work, Regalia and Bellanger [10] and Proudler *et al.* [9] have introduced QRD-based adaptive lattice linear filtering algorithms.

We have developed a QRD-based multichannel lattice algorithm for RLS adaptive filtering. We can use this algorithm to implement the lattice sections of the adaptive Volterra lattice filter we developed in the previous section, instead of the conventional multichannel lattice algorithm. The algorithms are fundamentally similar. There exists a duality between the two algorithms [10] which arises because both algorithms are based on identical geometrical principles. Upon solving the N^{th} order estimation problem, both methods provide solutions to all lower order problems. Also, both methods exploit the forward and backward linear prediction to reduce computation. Both methods carry out a QR-decomposition of the associated matrix; the conventional lattice algorithms apply Gram-Schmidt orthogonalization procedures to the data to generate an orthogonal basis set, *viz.,* the backward prediction errors. On the otherhand, QRD-based algorithms use Given's rotations to triangularize the data matrix.

We will now present a brief description of the ideas used for deriving fast, RLS, QRD-based second order lattice Volterra filter. For simplicity of notation, we will use the following notations:

$$\underline{x}_l(n) = [x(n), \lambda^{1/2}x(n-1), \cdots, \lambda^{(n-1)/2}x(1)]^T \tag{13}$$

$$\underline{x}_{q,i}(n) = [x(n)x(n-i), \cdots, \lambda^{(n-1)/2}x(1)x(1-i)]^T \tag{14}$$

$$\textit{and } \underline{X}(i,n) = [\underline{x}_l(n-i), \underline{x}_{q,0}(n-i), \cdots, \underline{x}_{q,i}(n)]. \tag{15}$$

Note that $\underline{X}(i,n)$ is a matrix of $i+2$ columns. Let $\underline{W}_n$ be the coefficient vector as before and let

$$\underline{d}(n) = [d(n), \lambda^{1/2}d(n-1), \cdots, \lambda^{n-1/2}d(1)]^T \tag{16}$$

be the desired response vector at time n. The problem can then be reformulated so that the squared norm of the error vector defined as

$$\underline{e}(n) = \underline{d}(n) - [\underline{X}(0,n), \underline{X}(1,n), \cdots, \underline{X}(N-1,n)]\underline{W}_n \tag{17}$$

is minimized. In the QR-RLS algorithm [2], an orthonormal matrix $Q_{N,n}$ triangularizes the data matrix such that

$$Q_{N,n}[\underline{X}(0,n), \cdots, \underline{X}(N-1,n)] = \begin{bmatrix} \underline{0} \\ S_{N,n} \end{bmatrix}, \tag{18}$$

where $S_{N,n}$ is an $N(N+1)/2 \times N(N+1)/2$ upper triangular matrix in the sense that small

$$S_{N,n} = \begin{bmatrix} S^{(1)}_{N,n}(P-1) & S^{(2)}_{N,n}(P-1) & \cdots & S^{(P)}_{N,n}(P-1) \\ S^{(1)}_{N,n}(P-2) & S^{(2)}_{N,n}(P-2) & \cdots & 0 \\ \cdot\ \vdots & \vdots & \vdots & \vdots \\ S^{(1)}_{N,n}(1) & S^{(2)}_{N,n}(1) & \cdots & 0 \\ S^{(1)}_{N,n}(0) & 0 & \cdots & 0 \end{bmatrix} \tag{19}$$

where $P = N(N+1)/2$. Let

$$Q_{N,n}\underline{d}(n) = \begin{bmatrix} \tilde{d}(n) \\ \bar{d}(n) \end{bmatrix} \tag{20}$$

where $\bar{d}(n)$ has P elements. It is straightforward to show [4] that the least squares solution we are seeking is given by

$$\bar{d}(n) = S_{N,n}\underline{W}_n \tag{21}$$

Since, $S_{N,n}$ is a traingular matrix, W_n can be solved using back substitution. Also, the squared norm of the error vector is the same as the squared norm of $\tilde{d}(n)$.

The basic problem then is that of recursively obtaining $Q_{N,n}$ from $Q_{N,n-1}$ in an efficient manner. It is easy to show [4] that

$$Q_{N,n} = \hat{Q}_{N,n} \begin{bmatrix} 1 & 0 \\ 0 & Q_{N,n-1} \end{bmatrix} \tag{22}$$

where $\hat{Q}_{N,n-1}$ is a series of P, 2×2 rotations that are applied in succession to rotate the top row of the data matrix $[\underline{X}(0,n), \cdots, \underline{X}(N-1,n)]$ into $\lambda^{1/2}S_{N,n-1}$.

The QRD-based lattice filter obtains $Q_{N,n}$ in an order recursive manner by first triangularizing $\underline{X}(0,n)$. The rotations that triangularize $\underline{X}(0,n)$ are then applied to $\underline{X}(1,n)$. Complete triangularization of the matrix $[\underline{X}(0,n)$, $\underline{X}(1,n)]$ requires additional rotations. These additional rotations are computed in an efficient manner in our algorithm. As in all fast RLS algorithms, our algorithm also uses forward and backward (nonlinear) predictors to update the rotations. The derivations are done by extending the ideas used for developing the single channel, linear QRD-based fast lattice filters [9]. Details are omitted because of page limitations. The idea is to triangularize $[\underline{X}(0,n), \cdots, \underline{X}(i-1,n)]$ at the i^{th} stage of the lattice, augment the data matrix with $\underline{X}(i,n)$, and then compute the additional rotations that are required to triangularize the augmented matrix for the $(i+1)^{th}$ stage of the lattice. Note that, similar to the situation in section 2, the number of additional column vectors added to the data matrix after each stage increases by one with every stage. Special care, again similar to what was done in section 2, must be taken to handle this situation. The complete algorithm is presented in Table 2.

TABLE 2: QR-DECOMPOSITION BASED SECOND ORDER VOLTERRA LATTICE FILTER

INITIALIZATION

$$e^b_{m-2,-1}(k,k) = e^f_{m-2,0}(k,k) = \delta \ ; \ m = 2, \cdots, N \ , \ k = 1, \cdots, m$$

DO 1 TO 42 FOR $n = 1$ *ONWARDS*

$$\underline{\alpha}^f_{0,n} = [x(n), x^2(n)], \ \ \underline{\alpha}^b_{0,n} = [x(n-1), x^2(n-1)], \ \ \gamma_{0,n} = 1 \tag{1}$$

$$\alpha^{j\,(m)}_{0,n} = x(n)x(n-m+2) \ ; \ m = 3, \cdots, N+1$$

$$\gamma^j_{0,n} = 1 \ , \ \alpha^{j\,(b)\,(3)}_{0,n} = x(n)x(n-1) \tag{2}$$

DO 3 TO 42 FOR $m = 2 \cdots, N$
DO 3 TO 7 FOR $k = 1 \cdots m$

$$e^b_{m-2,n-1}(k,k) = \sqrt{(\lambda^{1/2}e^b_{m-2,n-2}(k,k))^2 + (\alpha^b_{m-2,n-1}(k))^2} \tag{3}$$

$$\cos\theta^f_{m-2,n}(k) = \lambda^{1/2}e^b_{m-2,n-2}(k,k)/e^b_{m-2,n-1}(k,k) \tag{4}$$

$$\sin\theta^f_{m-2,n}(k) = \alpha^b_{m-2,n-1}(k)/e^b_{m-2,n-1}(k,k) \tag{5}$$

DO 6 TO 7 FOR $l = k+1 \cdots m$

$$\begin{aligned} e^b_{m-2,n-1}(k,l) &= \lambda^{1/2}e^b_{m-2,n-2}(k,l)\cos\theta^f_{m-2,n}(k) \\ &\quad +\alpha^b_{m-2,n-1}(l)\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{6}$$

$$\begin{aligned} \alpha^b_{m-2,n-1}(l) &= \alpha^b_{m-2,n-1}(l)\cos\theta^f_{m-2,n}(k) \\ &\quad -\lambda^{1/2}e^b_{m-2,n-2}(k,l)\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{7}$$

$$\gamma_{m-1,n} = \gamma_{m-2,n} \tag{8}$$

DO 9 TO 15 FOR $k = 1 \cdots m$
DO 9 TO 10 FOR $l = 1 \cdots, m$

$$\begin{aligned} \mu^f_{m-2,n}(k,l) &= \lambda^{1/2}\mu^f_{m-2,n-1}(k,l)\cos\theta^f_{m-2,n}(k) \\ &\quad +\alpha^f_{m-2,n}(l)\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{9}$$

$$\begin{aligned} \alpha^f_{m-1,n}(l) &= \alpha^f_{m-2,n}(l)\cos\theta^f_{m-2,n}(k) \\ &\quad -\lambda^{1/2}\mu^f_{m-2,n-1}(k,l)\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{10}$$

$$\mu^j_{m-2,n}(k) = \lambda^{1/2}\mu^j_{m-2,n-1}(k)\cos\theta^f_{m-2,n}(k) + \alpha^j_{m-2,n}\sin\theta^f_{m-2,n}(k) \tag{11}$$

$$\alpha^j_{m-1,n} = \alpha^j_{m-2,n}\cos\theta^f_{m-2,n}(k) - \lambda^{1/2}\mu^j_{m-2,n-1}(k)\sin\theta^f_{m-2,n}(k) \tag{12}$$

$$\gamma_{m-1,n} = \gamma_{m-1,n}\cos\theta^f_{m-2,n}(k) \tag{13}$$

$$\begin{aligned} \mu^{j\,(b)\,(m+1)}_{m-2,n}(k) &= \lambda^{1/2}\mu^{j\,(b)\,(m+1)}_{m-2,n-1}(k)\cos\theta^f_{m-2,n}(k) \\ &\quad +\alpha^{j\,(b)\,(m+1)}_{m-2,n}\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{14}$$

$$\begin{aligned} \alpha^{j\,(b)\,(m+1)}_{m-1,n} &= \alpha^{j\,(b)\,(m+1)}_{m-2,n}\cos\theta^f_{m-2,n}(k) \\ &\quad -\lambda^{1/2}\mu^{j\,(b)\,(m+1)}_{m-2,n-1}(k)\sin\theta^f_{m-2,n}(k) \end{aligned} \tag{15}$$

$$\epsilon^f_{m-1,n} = \gamma_{m-1,n}\underline{\alpha}^f_{m-1,n}\text{Forward Prediction Error} \tag{16}$$

DO 17 TO 21 FOR $k = 1 \cdots m$

$$e^{f}_{m-2,n}(k,k) = \sqrt{(\lambda^{1/2} e^{f}_{m-2,n-1}(k,k))^2 + (\alpha^{f}_{m-2,n}(k))^2} \tag{17}$$

$$\cos\theta^{b}_{m-2,n}(k) = \lambda^{1/2} e^{f}_{m-2,n-1}(k,k) / e^{f}_{m-2,n}(k,k) \tag{18}$$

$$\sin\theta^{b}_{m-2,n}(k) = \alpha^{f}_{m-2,n}(k) / e^{f}_{m-2,n}(k,k) \tag{19}$$

DO 20 TO 21 FOR $l = k+1 \cdots m$

$$\begin{aligned} e^{f}_{m-2,n}(k,l) &= \lambda^{1/2} e^{f}_{m-2,n-1}(k,l) \cos\theta^{b}_{m-2,n}(k) \\ &\quad + \alpha^{f}_{m-2,n}(l) \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{20}$$

$$\begin{aligned} \alpha^{f}_{m-2,n}(l) &= \alpha^{f}_{m-2,n}(l) \cos\theta^{b}_{m-2,n}(k) \\ &\quad - \lambda^{1/2} e^{f}_{m-2,n-1}(k,l) \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{21}$$

$$\gamma_{m-1,n+1} = \gamma_{m-2,n} \tag{22}$$

DO 23 TO 25 FOR $k = 1, \cdots, m$
DO 23 TO 24 FOR $l = 1, \cdots, m$

$$\begin{aligned} \mu^{b}_{m-2,n-1}(k,l) &= \lambda^{1/2} \mu^{b}_{m-2,n-2}(k,l) \cos\theta^{b}_{m-2,n}(k) \\ &\quad + \alpha^{b}_{m-2,n-1}(l) \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{23}$$

$$\begin{aligned} \alpha^{b}_{m-1,n}(l) &= \alpha^{b}_{m-2,n-1}(l) \cos\theta^{b}_{m-2,n}(k) \\ &\quad - \lambda^{1/2} \mu^{b}_{m-2,n-2}(k,l) \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{24}$$

$$\gamma_{m-1,n+1} = \gamma_{m-1,n+1} \cos\theta^{b}_{m-2,n}(k) \tag{25}$$

$$\epsilon^{b}_{m-1,n} = \gamma_{m-1,n+1} \underline{\alpha}^{b}_{m-1,n} \text{ Backward Prediction Error} \tag{26}$$

DO 27 TO 30 FOR $k = 1, \cdots, m$

$$\phi^{j}_{m-2,n}(k) = \lambda^{1/2} \phi^{j}_{m-2,n-1}(k) \cos\theta^{b}_{m-2,n}(k) + \alpha^{j}_{m-2,n} \sin\theta^{b}_{m-2,n}(k) \tag{27}$$

$$\tilde{\alpha}^{j}_{m-1,n} = \alpha^{j}_{m-2,n} \cos\theta^{b}_{m-2,n}(k) - \lambda^{1/2} \phi^{j}_{m-2,n-1}(k) \sin\theta^{b}_{m-2,n}(k) \tag{28}$$

$$\begin{aligned} \phi^{j\,(b)\,(m+1)}_{m-2,n}(k) &= \lambda^{1/2} \phi^{j\,(b)\,(m+1)}_{m-2,n-1}(k) \cos\theta^{b}_{m-2,n}(k) \\ &\quad + \alpha^{j\,(b)\,(m+1)}_{m-2,n} \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{29}$$

$$\begin{aligned} \tilde{\alpha}^{j\,(b)\,(m+1)}_{m-1,n} &= \alpha^{j\,(b)\,(m+1)}_{m-2,n} \cos\theta^{b}_{m-2,n}(k) \\ &\quad - \lambda^{1/2} \phi^{j\,(b)\,(m+1)}_{m-2,n-1}(k) \sin\theta^{b}_{m-2,n}(k) \end{aligned} \tag{30}$$

$$\tilde{\gamma}_{m-1,n} = \gamma_{m-1,n+1} \ , \ \tilde{e}_{m-1,n} = \tilde{\gamma}_{m-1,n} \tilde{\alpha}^{j}_{m-1,n} \tag{31}$$

$$\tilde{e}^{(b),(m+1)}_{m-1,n} = \tilde{\gamma}_{m-1,n} \tilde{\alpha}^{j\,(b)\,(m+1)}_{m-1,n} \tag{32}$$

DO 33 TO 41 FOR $k = m+1, \cdots, N+1$

$$\gamma^{(k)}_{m-1,n} = \gamma^{(k)}_{m-2,n} \tag{33}$$

DO 34 TO 36 FOR $l = 1, \cdots, m$

$$\mu^{j\,(k)}_{m-2,n}(l) = \lambda^{1/2} \mu^{j\,(k)}_{m-2,n-1}(l) \cos\theta^{f}_{m-2,n-1}(l) + \alpha^{j\,(k)}_{m-2,n} \sin\theta^{f}_{m-2,n-1}(l) \tag{34}$$

$$\alpha^{j\,(k)}_{m-1,n} = \alpha^{j\,(k)}_{m-2,n} \cos\theta^{f}_{m-2,n-1}(l) - \lambda^{1/2} \mu^{j\,(k)}_{m-2,n-1}(l) \sin\theta^{f}_{m-2,n-1}(l) \tag{35}$$

$$\gamma^{(k)}_{m-1,n} = \gamma^{(k)}_{m-1,n} \cos\theta^{f}_{m-2,n-1}(l) \tag{36}$$

$$\gamma^{(k)}_{m-1,n+1} = \gamma^{(k)}_{m-2,n} \tag{37}$$

DO 38 TO 40 FOR $l = 1, \cdots, m$

$$\phi^{j\,(k)}_{m-2,n}(l) = \lambda^{1/2} \phi^{j\,(k)}_{m-2,n-1}(l) \cos\theta^{b}_{m-2,n-1}(l) + \alpha^{j\,(k)}_{m-2,n} \sin\theta^{b}_{m-2,n-1}(l) \tag{38}$$

$$\tilde{\alpha}^{j\,(k)}_{m-1,n} = \alpha^{j\,(k)}_{m-2,n} \cos\theta^{b}_{m-2,n-1}(l) - \lambda^{1/2} \phi^{j,(k)}_{m-2,n-1}(l) \sin\theta^{b}_{m-2,n-1}(l) \tag{39}$$

$$\gamma^{(k)}_{m-1,n+1} = \gamma^{(k)}_{m-1,n+1} \cos\theta^{b}_{m-2,n-1}(l) \tag{40}$$

$$\tilde{\gamma}^{(k)}_{m-1,n} = \gamma^{(k)}_{m-1,n+1} \ , \ \tilde{e}^{(k)}_{m-1,n} = \tilde{\gamma}^{(k)}_{m-1,n} \tilde{\alpha}^{(k)}_{m-1,n} \tag{41}$$

COMMENT: Now set-up the inputs for the next stage

$$\underline{\alpha}^{f}_{m-1,n} = [\underline{\alpha}^{f}_{m-1,n} \ \tilde{\alpha}^{j\,(m+1)}_{m-1,n}] \ , \ \underline{\alpha}^{b}_{m-1,n} = [\underline{\alpha}^{b}_{m-1,n} \ \tilde{\alpha}^{j\,(b)\,(m+1)}_{m-1,n}] \tag{42}$$

4 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. A fast least squares lattice and a fast QR-lattice adaptive filtering algorithms based on the above structure were also presented. The predictor section of the nonlinear filter structure employed in our work is similar to that derived by Zarzycki [12]. The adaptive filtering algorithms are based on the multichannel lattice filters developed by Ling and Proakis [7]. These algorithms share the fast convergence property of fast least squares transversal Volterra filters [8] without suffering from problems with numerical instability. The QRD-based algorithm is expected to have particularly good numerical properties since every operation for its implementation is done using numerically stable Given's rotation.

References

[1] S. Bendetto, E. Biglieri, and V. Castellini, *Digital Transmission Theory,* New Jersey: Prentice-Hall, 1987.

[2] J. M. Cioffi, " The fast adaptive rotors RLS algorithm, " to appear in *IEEE Trans. on Acoustics, Speech, and Signal Processing,* April 1990.

[3] M. J. Coker and D. N. Simkins, "A nonlinear adaptive noise canceller," *Proc of the ICASSP,* pp. 470-473, 1980.

[4] S. Haykin, "Adaptive Filter Theory," New Jersey: Prentice-Hall, 1985.

[5] T. Koh and E. J. Powers, "An adaptive nonlinear digital filter with lattice orthogonalization," *Proc. ICASSP,* pp. 37-40, April 1983.

[6] P. J. Lenk and S. R. Parker, " Nonlinear modeling by discrete orthogonal lattice structure, " *Proc. IEEE Int. Symp. Circuits and Systems,* 1984.

[7] F. Ling and J. Proakis, "A generalized multichannel least-squares lattice algorithm based on sequential processing stages, " *IEEE Trans. Acoustics, Speech, and Signal Processing,* Vol. ASSP-32, No. 2, April 1984.

[8] V. J. Mathews and J. Lee, "A fast recursive least-squares second order Volterra filter," *Proc. of ICASSP,* 1989.

[9] I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, " QRD-based lattice filter algorithms, " *Proc. SPIE: Advanced Algorithms and Architectures for Signal Processing,* San Diego, California, Aug. 1989.

[10] P. A. Regalia and M. G. Bellanger, " On the duality between fast QR methods and lattice methods in least-squares adaptive filtering, " submitted to *IEEE Trans. on Acoustics, Speech, and Signal Processing,* 1989.

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

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