VOLTERRA AND GENERAL POLYNOMIAL RELATED FILTERING

V. John Mathews
Deptartment of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112, USA

Giovanni L. Sicuranza
DEEI
University of Trieste
Via A. Valerio, 10, 34127 Trieste, Italy

ABSTRACT

This paper presents a review of polynomial filtering and, in particular, of the truncated Volterra filters. Following the introduction of the general properties of such filters, issues such as efficient realizations, design, adaptive algorithms and stability are discussed.

1 INTRODUCTION

Volterra filters are discrete nonlinear time-invariant systems with memory described by the discrete Volterra series expansion

$$y(n) = h_0 + \sum_{p=1}^{\infty} \bar{h}_p[x(n)] \qquad (1)$$

where

$$\bar{h}_p[x(n)] = \sum_{m_1=0}^{\infty} \cdots \sum_{m_p=0}^{\infty} h_p(m_1, ..., m_p) \cdot x(n-m_1)...x(n-m_p) \qquad (2)$$

since the discrete independent variables $m_1, ..., m_p$ are usually defined on a causal support. Here h_0 is an offset term, $h_1(m_1)$ is the impulse response of a digital IIR filter, and $h_p(m_1, ..., m_p)$ can be considered as a generalized p-th order impulse response characterizing the nonlinear behaviour of the filter. The upper limit in the summations in (1) given as infinity indicates infinite memory of the system. As in the linear case, one can obtain practical realizations of the system by devising recursive models of finite order involving delayed output terms or by truncating equation (1) and the included summations so that it has only a finite number of terms. In the latter case $h_1(m_1)$ represents an FIR filter, and the effect of the nonlinearity on the output depends only on the input values. These filters have been extensively studied because of their relative simplicity.

It is worth noting that the Volterra series expansion can be in fact considered as a Taylor series with memory. Thus, like the Taylor series, it can not perform well in presence of discontinuities or saturation effects in the system description.

Two important aspects of the Volterra series expansion are evident from its definition in (1) and (2): i) the filter output is linear with respect to the elements of the Volterra kernels (this property is the basis of various developments described in the next sections); ii) the functionals $\bar{h}_p[x(n)]$ can be interpreted as multidimensional convolutions, and thus Volterra filters can be represented by means of multidimensional linear transforms, since their input-output relations are expressed by sums of multidimensional convolutions. Let us consider, as an example, the relation defining the quadratic term

$$y(n) = \bar{h}_2[x(n)] = \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} h_2(m_1, m_2) x(n-m_1) x(n-m_2). \qquad (3)$$

It can be considered as a particular form of a 2-D convolution

$$w(n_1, n_2) = \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} h_2(m_1, m_2) v(n_1 - m_1, n_2 - m_2) \qquad (4)$$

for $v(n_1 - m_1, n_2 - m_2) = x(n-m_1)x(n-m_2)$ and $n_1 = n_2 = n$, so that $y(n) = w(n, n)$ [14, 19]. It is possible to use, instead of the kernel $h_2(m_1, m_2)$, its 2-D z-transform $H_2(z_1, z_2)$ or its Fourier transform $H_2(e^{j\omega_1}, e^{j\omega_2})$ and then make use of the properties of these 2-D operators for the characterization of the quadratic kernel. So, for example, by using the 2-D Fourier transform, we can derive the expression for $y(n)$ in the double integral harmonic representation

$$y(n) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} H_2(e^{j\omega_1}, e^{j\omega_2}) \cdot X(\omega_1) X(\omega_2) e^{j(\omega_1+\omega_2)n} d\omega_1 d\omega_2 \qquad (5)$$

where $X(\omega)$ is the Fourier transform of $x(n)$. It is worth noting that if the input is a monochromatic signal at angular frequency ω_a, i.e., $X(\omega) = a u_0(\omega - \omega_a)$, the output given by (5) is $y(n) = a^2 H_2(e^{j\omega_a}, e^{j\omega_a}) e^{j2\omega_a n}$ which is still a monochromatic signal but with angular frequency $2\omega_a$.

These considerations, easily extended to the higher kernels of the Volterra series expansion, show one of the peculiarities of the nonlinear behaviour of a Volterra filter: new frequencies appear in the output signal that are not present in the input signal.

2 EFFICIENT REALIZATIONS OF VOLTERRA FILTERS

Two effective realizations for the Volterra filters defined on a finite support have been presented in the past: the first one based on matrix decompositions [12, 1], and the second one based on the distributed arithmetic technique [17]. Both realizations exhibit properties of modularity and regularity very suitable for VLSI implementations, and permit computational and hardware advantages.

The first structure, valid only for a quadratic filter, can be obtained by applying a matrix decomposition technique to the matrix $\mathbf{H}$ in the following equation

$$\bar{h}_2[x(n)] = \mathbf{X}^T\mathbf{H}\mathbf{X} \tag{6}$$

Here $\mathbf{X}$ is the vector of the N most recent input samples, and the matrix $\mathbf{H}$ is composed of the elements of $h_2(m_1, m_2)$. Furthermore, $\mathbf{H}$ is symmetric since the quadratic kernel can be considered, without loss of generality, as a symmetric function of its arguments. One of the most useful matrix decompositions is the so-called LU (Lower-Upper) decomposition. Using an LU decomposition, the matrix $\mathbf{H}$ can be written in the form

$$\mathbf{H} = \sum_{i=1}^{r} d_i \mathbf{L}_i \mathbf{L}_i^T \tag{7}$$

where $r = rank[\mathbf{H}] \leq N$, d_i's are real numbers and the $\mathbf{L}_i$ are $N \times 1$ vectors having $i-1$ leading zeros [1]. The substitution of (7) in (6) results in the equation

$$y(n) = \sum_{i=1}^{r} d_i [\mathbf{X}^T\mathbf{L}_i][\mathbf{L}_i^T\mathbf{X}] = d_i y_i^2(n) \tag{8}$$

where $y_i(n) = \mathbf{X}^T\mathbf{L}_i = \mathbf{L}_i^T\mathbf{X}$ represents a linear, finite-support filter. Hence the overall output $y(n)$ is computed by combining r linear FIR filters, each followed by a squaring operation. Since the FIR filters of the parallel channels have an increasing number of zero elements for i ranging from 1 to r, a set of sequential square-in add-out operations can be applied to reduce the data throughput delay and the computational requirements. Moreover, a preliminary SV decomposition allows us to obtain an approximate expansion by truncation of the exact expression for the matrix $\mathbf{H}$ to the first p terms ($p \leq r \leq N$). In this case, the number of parallel channels required by the subsequent LU decomposition is reduced to p, and thus substantial savings in hardware can be achieved.

While the above vector-matrix representation is valid only for a quadratic filter, an alternative practical description can be used for all the Volterra kernels. This description is based on a whole vector notation

$$\bar{h}_p[x(n)] = \mathbf{X}_p^T\mathbf{H}_p \tag{9}$$

where the vector $\mathbf{X}_p$ is defined by the recursive rule $\mathbf{X}_p = \mathbf{X}_1 \otimes \mathbf{X}_{p-1}$. Here $\mathbf{X}_1 = \mathbf{X}$, and the symbol $\otimes$ indicates the Kronecker product of vectors. The execution of the Kronecker product implies a well-defined ordering for the elements in $\mathbf{X}_p$ which in turn leads to a corresponding order in the vector $\mathbf{H}_p$. The number of the elements in $\mathbf{X}_p$ and $\mathbf{H}_p$ is equal to N^p; when the symmetry of the Volterra kernels is exploited, the length of the vectors $\mathbf{X}_p$ and $\mathbf{H}_p$ can be accordingly reduced.

Efficient realizations can now be derived by employing the so-called distributed arithmetic: this technique directly suggests both combinatorial structures based on logical operations at the bit level and memory oriented realizations which use suitable function values stored in ROM's or RAM's [17, 18]. The operator $\bar{h}_p$ is decomposed in the form

$$\bar{h}_p[x(n)] = \mathbf{P}_p^T\mathbf{Q}_p^T\mathbf{H}_p \tag{10}$$

where $\mathbf{Q}_p = \mathbf{Q}_1 \otimes \mathbf{Q}_{p-1}$, $\mathbf{P}_p = \mathbf{P}_1 \otimes \mathbf{P}_{p-1}$, $\mathbf{X}_1 = \mathbf{Q}_1\mathbf{P}_1$, and $\mathbf{X}_p = \mathbf{Q}_p\mathbf{P}_p$. The matrix $\mathbf{Q}_1$ is the input vector $\mathbf{X}_1$ subdivided in its B component bits, and $\mathbf{P}_1$ is the vector of the corresponding B binary weights. Combinatorial structures can be easily derived for quadratic filters from equation (13). Memory-oriented structures are obtained by using the rows of $\mathbf{Q}_p^T$ as addresses to the memory in which all the possible binary sums of the elements of $\mathbf{H}_p$ are stored. These values are summed up according to the weights in $\mathbf{P}_p$to calculate the output values. The complexity of this structure (i.e. the number of memory cells), which exponentially increases with N, can be accordingly reduced by using address-compression and split-address techniques [17],[18].

Two important comments can be made now:
i) The above results can be applied to the M-D finite-support Volterra filters also by arranging the values of the input array and the filter coefficients in vectors and matrices, respectively, according to some mapping function Φ which corresponds to an ordered scanning of the M-D data [12]. Therefore, in the case of the matrix decomposition of quadratic filters, equation (6) still holds. On the other hand, when exploiting the full vector approach, a suitable nesting of the ordering rule generated by the Kronecker product with the one imposed by the mapping function Φ is required.
ii) The efficient structures here considered can be made adaptive by using for example the well-known LMS algorithm. The structures based on matrix decompositions require the updating of

the vectors $\mathbf{L}_i$ and of the scalar values d_i [9], while in memory oriented-realizations the content of the memory has to be directly updated. For this, the relevant equations can be found in [18].

3 DESIGN OF VOLTERRA FILTERS

One of the most important aspects of Volterra filtering is the demand for efficient design techniques. Indeed, only a few results have been reported in the literature up to now and they are limited to quadratic filters. In particular, two design techniques have been recently proposed, both working in the data domain: the first technique [15] is based on an optimization approach, and the second one [16] exploits a set of responses of the quadratic component to couples of suitably located impulses (the "bi-impulse responses").

In the optimization approach the filter structure is considered already fixed and the filter coefficients are assumed as variables. In order to reduce the number of independent design variables some constraints can be added, as symmetry conditions, preservation of the input level in a uniform zone and 90^o isotropy in the 2-D case. In general, the objective function to be minimized is formulated as the difference between an ideal desired filter output y^o and the actual output y obtained from a noisy input x' resulting as the superposition of a noise n to a reference input x. In order to get a manageable objective function, simple schematic reference inputs x are used, such as unit steps and impulses, on windows of limited amplitude. Then the deviation measure used is the sum of the squared differences on all the samples of the reference window [15].

The second approach is more formal since it is based on the fact that a complete quadratic filter is uniquely defined once the impulse response of its linear components as well as a set of responses of its quadratic component to couples of suitably located impulses (the so-called bi-impulse response) are determined. Let us suppose that the input of a 1-D system is obtained as the sum of two components $x(n) = x_a(n) + x_b(n)$. By substitution in the 1-D input-output relation

$$y(n) = \bar{h}_1[x(n)] + \bar{h}_2[x(n)] \qquad (11)$$

we obtain

$$y(n) = y_a(n) + y_b(n) + 2h_2[x_a(n), x_b(n)] \quad (12)$$

where $y_a(n)$ and $y_b(n)$ are the outputs of the quadratic filter when the input $x(n)$ is equal to $x_a(n)$ or $x_b(n)$, respectively, and

$$\bar{h}_2[x_a(n), x_b(n)] = \sum_i \sum_j h_2(i,j) x_a(n-i) x_b(n-j). \qquad (13)$$

If $x_a(n)$ and $x_b(n)$ are two unitary impulses, $u_0(n)$ and $u_0(n-m)$, respectively at the origin and at a distance m from the origin, by substitution in (13), it can be seen that the expression (13) determines a coefficient of the bi-impulse (couple of impulses) response of the quadratic operator

$$\bar{h}_2[u_0(n), u_0(n-m)] = h_2(0,m). \qquad (14)$$

As a consequence, a 1-D operator having a support of N samples is uniquely defined by N bi-impulse responses (plus 1 impulse response for the linear component); its quadratic part has (when the kernel symmetry is taken into account) $N(N+1)/2$ independent coefficients, N of which are related to the response to a couple of impulses in the origin, $N-1$ to impulses having distance 1, $N-2$ to impulses having distance 2, ... , 1 to impulses having distance $N-1$.

The extension to the 2-D case is not straightforward and has been presented in detail in [16]. In summary, a 3x3 quadratic filter is formed by 81 independent coefficients; by considering the kernel symmetry, 45 coefficients remain independent. Each of these 45 coefficients is involved in one of the 13 independent responses to suitably located couples of impulses. When the isotropy constraints are added, the independent responses are reduced to 6 and the independent coefficients to 11. The 6 bi-impulse responses are then classified according to a measure of the distance between the corresponding input impulses. Therefore, we can refer to responses of the type 0, 1 or 2 for impulse distances of 0, 1 or 2 pixels, respectively. This classification greatly simplify the subsequent design procedure. The suggested method consists in fixing the coefficients of the linear part of the filter, according to various problem-dependent choices, and then fixing the quadratic coefficients in order to enhance, smooth or cancel the effects of well-defined bi-impulse responses [16]. This procedure can be extended by considering more precise subdivisions of the bi-impulse responses, as shown for example in another paper in these Proceedings [3].

The extension of these two design techniques to higher-order kernels is in principle feasible, even though quite complex to derive in practice.

4 ADAPTIVE ALGORITHMS

As mentioned earlier, one of the biggest advantages of using polynomial models is the fact that the system output is linear in some polynomial functions of the input and output signals. Consequently, many linear filtering concepts can be extended to the polynomial case. Much of the work in adaptive nonlinear filtering makes use of this approach. While the ideas discussed in this section are generally applicable, we will restrict ourselves to the quadratic model for pedogogic simplicity. Two of the most commonly employed classes of algorithms for adaptive filtering are gradient search algorithms and recursive least squares algorithms.

Least Mean Square (LMS) Quadratic Filters

Consider the problem of adaptively estimating a desired response signal d(n) as a quadratic expansion in the most recent N samples of the input signal x(n), i.e., we want to estimate d(n) as

$$\hat{d}(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} h_2(m_1, m_2; n) \cdot x(n-m_1)x(n-m_2) \quad (15)$$

where $h_2(m_1, m_2; n)$ for different values of m_1 and m_2 are the coefficients of the adaptive filter at time n. The objective of the LMS quadratic filters is to iteratively adjust the coefficients of the adaptive filter so as to reduce

$$e^2(n) = \left(d(n) - \hat{d}(n)\right)^2 \quad (16)$$

at each iteration. The coefficient adjustment is done as

$$h_2(m_1, m_2; n+1) = h_2(m_1, m_2; n) + \mu e(n)x(n-m_1)x(n-m_2) \quad (17)$$

where μ is a small positive constant that controls the speed of convergence and the steady-state tracking capabilities.

The LMS quadratic filter can be analyzed in a manner similar to the analysis of LMS linear filters. Let

$$\mathbf{H(n)} = [h_2(0,0;n), h_2(0,1;n), \ldots, h_2(0,N-1;n), h_2(1,1;n) \ldots\ldots h_2(N-1,N-1;n)]^T . \quad (18)$$

denote the coefficient vector for the adaptive filter at time n. Similarly, define the input vector at a time n as

$$\mathbf{X(n)} = [x^2(n), x(n)x(n-1), \ldots, x(n)x(n-N+1), x^2(n-1), \ldots, x^2(n-N+1)]^T \quad (19)$$

Then, one can show, using several simplifying assumptions commonly employed in the analysis of adaptive filters, that the adaptive filter will converge in the mean squared sense if

$$0 < \mu < \frac{2}{3tr\mathbf{R_{xx}}} \quad (20)$$

where tr (·) denotes the trace of the matrix (·) and R_{xx} is the statistical autocorrelation matrix of the input vector X(n). The above result assumes stationarity of the input signal. One can also show that the speed of convergence is controlled by the eigenvalue spread of the autocorrelation matrix.

The dependence of the adaptive filter on the statistics of the input signal is one major problem associated with LMS adaptive filters. This problem is even greater for the nonlinear case than for the linear case. This is because the eigenvalue spread of the autocorrelation matrix will in general be much larger for the nonlinear filtering problem than for the linear case. There are several approaches that have been employed in the past for partially or completely overcoming the dependence of the quadratic filters on the input signal statistics. We will briefly discuss some of these approaches now.

Variable Step-Size Algorithms [11]

The most common approach for improving the performance of the LMS adaptive filters is that of employing a time-varying convergence factor. Several heuristic algorithms are available in the literature. We now discuss an algorithm that is derived based on more rigorous and systematic considerations. In this approach, the step-size is changed so that the change is guaranteed to bring about a reduction in the squared error. The algorithm can be implemented using the following equations:

$$e(n) = d(n) - \mathbf{H^T(n)X(n)}$$
$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{X^T(n)X(n-1)}e(n-1) \quad (21)$$

and

$$\mathbf{H(n+1)} = \mathbf{H(n)} + \mu(n)\mathbf{X(n)}e(n). \quad (22)$$

In the above equations, ρ is a small positive constant that controls the rate of change of the step-size. Experimental results have indicated much faster convergence behavior for the above adaptive step-size algorithm than the basic LMS algorithm.

Recursive Least Squares (RLS) Quadratic Filters

Most least squares adaptive filters attempt to find the exact solution to the problem of minimizing a deterministic function of the squared error values. For example, the exponentially weighted RLS quadratic filter minimizes the cost function

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \left(d(k) - \mathbf{H^T(n)X(k)}\right)^2 \quad (23)$$

at each time. An interative solution to the problem is given in Table 1. The basic derivation is given in [4].

Table 1: The RLS Adaptive Quadratic Filter

Initialization

$$\mathbf{H(0)} = [0, 0, \ldots, 0]^T$$
$$\mathbf{R^{-1}(0)} = \delta^{-1} I$$
$\delta =$ *a small positive constant*

Algorithm

$$\begin{aligned}
\mathbf{k(n)} &= \frac{\lambda^{-1}\mathbf{R^{-1}(n-1)X(n)}}{1+\lambda^{-1}\mathbf{X^T(n)R^{-1}(n-1)X(n)}} \\
\epsilon(n) &= d(n) - \mathbf{H^T(n-1)X(n)} \\
\mathbf{H(n)} &= \mathbf{H(n-1)} + \mathbf{k(n)}\epsilon(n) \\
\mathbf{R^{-1}(n)} &= \lambda^{-1}\mathbf{R^{-1}(n-1)} - \\
&\qquad \lambda^{-1}\mathbf{k(n)X^T(n)R^{-1}(n-1)} \\
e(n) &= d(n) - \mathbf{H^T(n)X(n)}
\end{aligned} \tag{24}$$

The performance of the adaptive RLS Volterra filters have been analyzed in [7]. The analysis has shown that the convergence behavior and the steady-state properties of the adaptive filter are independent of the signal statistics. Furthermore, experimental comparisons with gradient search and other competing algorithms have demonstrated that the RLS algorithm enjoys many superiorities over them in its capabilities.

One big drawback of the RLS adaptive quadratic filter in Table 1 over its LMS counterparts is that the RLS algorithm requires an order of magnitude more computational complexity than the LMS algorithms. More efficient algorithms have been developed in the recent past. Derivations and performance evaluations can be found in [7].

The fast Volterra filter derivations transform the nonlinear filtering problem into an equivalent multi-channel, but linear filtering problem and then makes use of the theory of fast multichannel filters. The resultant algorithms require $O(N^3)$ arithmetic operations per sample to implement an adaptive RLS quadratic filter with N-sample memory in the system model. This compares with the $O(N^4)$ operations required by the algorithm in Table 1. It should also be pointed out that LMS quadratic filters and its variations require only $O(N^2)$ operations per sample. The increased computational complexity even for the most efficient algorithm is the price we pay for the superior performance capabilities of the RLS Volterra filters.

The early versions of the fast RLS Volterra filters utilized a direct form structure for the system model and had relatively poor numerical properties. Very recently, fast algorithms involving lattice and other non-direct form structures have been developed [20, 21]. Particularly noteworthy is the fact that some of these algorithms are implemented solely using numerically robust Givens' rotations. Extensive experimental results involving finite precision (fixed-point) implementation of these filters have been presented in [20], [21].

5 NONLINEAR LATTICE FILTERS

Lattice filter structures enjoy several advantages over direct form structures. One of the advantages is that lattice filters in general have better numerical properties than direct form filters. They also possess good modularity and parallelizeability. Consequently, such structures are attractive from an implementational point of view. In the context of adaptive filters, if the coefficients of a lattice predictor are appropriately chosen, the predictor performs a Gram-Schmidt orthogonalization of the input signal. This would enable design of gradient-search adaptive filters with better convergence properties than direct form structures. Computationally stable and numerically efficient RLS algorithms can also be developed using lattice filter structures.

A block diagram of a lattice filter structure for a second order Volterra system model with input-output relationship

$$\begin{aligned}
\hat{d}(n) = &\sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) + \\
&\sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} h_2(m_1,m_2)\cdot \\
&x(n-m_1)x(n-m_2)
\end{aligned} \tag{25}$$

is shown in Figure 1. A tutorial derivation of this structure is given in [10]. The key idea in the derivation is that the number of input signals to any stage is one more than the number of input signals to the previous stage. In the figure, the number of lines going into and out of a system component indicates the number of input and output signals, respectively, of that component. If the coefficient matrices $K_1^f, K_1^b, K_2^f, K_2^b$, etc. are appropriately chosen, the error vectors $\underline{b}_0(n), \underline{b}_1(n), \underline{b}_2(n)$, etc. will be orthogonal to each other. Since the number of signals at each stage increases by one, there are a few computations that must be done outside the basic lattice. These computations are also shown in the figure.

Since the error vectors are orthogonal to each other (their elements also span the space spanned by the elements of the original input vector) we can estimate a desired response signal d(n) as a linear combination of the error vectors $b_0(n), b_1(n), \ldots, b_{N-1}(n)$. The coefficient matrices can be individually adapted in an adaptive filter and consequently a gradient adaptive lattice algorithm will have much better convergence behavior than its direct form counterpart. The

structure in Figure 1 is also the basis of several computationally efficient RLS adaptive Volterra filters [20], [21].

One disadvantage of the nonlinear lattice filter structure in Figure 1 is that it is an overparameterized system, i.e., a system that can be described using $O(N^2)$ coefficients using direct form methods require $O(N^3)$ coefficients in the lattice parameterization. No general, exactly parameterized lattice structure is currently available for the Volterra system model. However, such a system is indeed available for Gaussian input signals [5].

6 RECURSIVE NONLINEAR SYSTEMS

Perhaps the biggest disadvantage associated with truncated Volterra series modeling of nonlinear systems is the extremely large number of coefficients such models require. Note that a very general Volterra system model involving P-th order nonlinearity and N-sample memory requires $O(N^P)$ coefficients. Even for moderately large values of N and P, the number of coefficients can become unreasonably large. One approach to dealing with this problem is to look for sparse Volterra filter approximations. Some of the recent developments in this area have been reviewed in Section 2. We now briefly discuss an alternate approach to dealing with the large number of coefficients associated with truncated Volterra system models. This involves the use of recursive nonlinear system models.

The input-output relationship of a general polynomial system model with N-sample memory and up to P-th order nonlinear terms can be described using the following nonlinear difference equation:

$$y(n) = \sum_{i=0}^{P} P_i \{y(n-1), y(n-2), \ldots, y(n-N+1), x(n), \ldots (n-N+1)\} \quad (26)$$

Here, $P_i(\cdot)$ denotes polynomial involving only i-th order of terms in the signal samples within the brackets. The simplest of such recursive nonlinear system models is the bilinear system model whose input-output relationship is given by

$$y(n) = \sum_{i=1}^{N-1} b_i y(n-i) + \sum_{j=0}^{N-1} a_j x(n-j) + \sum_{i=1}^{N-1} \sum_{j=0}^{N-1} c_{ij} x(n-j) y(n-i) \quad (27)$$

It is known that bilinear system models can approximate a very large class of nonlinear systems with arbitrary precision. Much like linear IIR filters can approximate a large number of linear systems with great parsimony in the use of coefficients, bilinear and other recursive nonlinear models can also approximate a large number of nonlinear systems with good efficiency. Consequently, there has been a fair amount of recent effort to model nonlinear systems with polynomial models. We will not go into the details here, but will instead discuss the very important issue of stability of recursive nonlinear systems.

Stability of Recursive Systems

One of the biggest problems associated with recursive nonlinear systems is that almost all such systems are inherently unstable. To see this more clearly, let us rewrite the input-output relationship for a bilinear system as

$$y(n) = \sum_{i=1}^{N-1} \left\{ b_i + \sum_{j=0}^{N-1} c_{ij} x(n-j) \right\} \cdot y(n-i) + \sum_{j=0}^{N-1} a_j x(n-j) \quad (28)$$

It is relatively straightforward to infer from the above equation that one can almost always find bounded input signals that can cause the output of the system to diverge. However, there is also a class of signals for which a given recursive nonlinear system may produce very useful outputs. It is such class of signals and any associated conditions for input-dependent stability that we are interested in.

We will now describe a sufficient condition for guaranteeing that the output of a bilinear system described in (27) whenever the input signals are bounded by some fixed constant M_x.

Theorem [8] *For the bilinear system model represented by (27), let p_1, p_2,..., and p_{N-1} denote the zeros of the polynomial $q^{N-1}\left(1 - \sum_{i=1}^{N-1} b_i q^{-i}\right)$. Given a real, positive number M_x,*

$$\begin{cases} |p_i| < 1, \text{ for } i = 1, 2, \ldots, N-1, \text{ and} \\ M_x \sum_{i=0}^{N-1} \sum_{j=1}^{N-1} |c_{i,j}| < \prod_{i=1}^{N-1} (1 - |p_i|) \end{cases} \quad (29)$$

constitute a sufficient condition for every $x(n)$ bounded by M_x, i.e., $|x(n)| \leq M_x$, to produce a bounded output $y(n)$.

It should be noted that when there is no nonlinearity in the system, the above theorem reduces to the well-known necessary and sufficient condition for the resultant linear time-invariant system to be stable in the bounded-input, bounded-output sense. A detailed derivation of the theorem is available in [6].

Adaptive Bilinear Filters

Adaptive systems to track bilinear or other recursive nonlinear system models can be devised in a relatively straightforward manner. For a tutorial overview of such filters, the reader is referred to [10]. The biggest issue associated with such systems is that of stability. When developing and using such filters, one must make sure

that the system model produced by the adaptive filter is guaranteed to be stable, or if this cannot be achieved, one must equip the adaptive filters with continuous stability monitoring mechanisms. We will now briefly review some of the recent results in this area of nonlinear filtering research.

Extended least squares adaptive polynomial filters are guaranteed to be stable in the sense that the time averaged value of the squared estimation error is always bounded by a finite quantity [8]. However, this result does not imply that the adaptive filter coefficients will not fluctuate wildly for certain type of signals. Variations of exact least squares adaptive filters and gradient search algorithms do not share the property of bounded average squared error value. In [2], this problem has been overcome by appending a Kalman filter to the basic adaptive filter to monitor the stability of the overall system. The Kalman filter senses the onset of instability and modifies the behavior of the basic adaptive filter in such situations. Gradient search adaptive bilinear filters equipped with a projection facility can also be developed. Such systems will continuously check the adaptive filter coefficients to ensure that the conditions of the theorem described earlier are satisfied. If the coefficients do not satisfy the above conditions, the projection mechanism projects the coefficients back into the space of parameters that satisfy the conditions of the theorem and thus avoids instability problems.

All the above-described results are of preliminary nature and much work in the way of performance evaluation and system development needs to be still done.

7 FINAL REMARKS

The trends in the research on Volterra filters are related to the studies on possible simplifications of the general model which is often too complex to deal with. On the other hand, interesting results have been already obtained, especially in 2-D and 3-D applications, with filters having a very limited support, at least for quadratic operators. In this context, the study of sparse Volterra filters is thus very important, and the realizations based on matrix decompositions are effective since the SV decomposition can be usefully applied.

With reference to the design problem, again it is important to consider simple operators whose behaviour is defined according to a proper classification: an example of this kind of work can be found in these Proceedings [3]. Moreover, extensions of the mentioned design techniques to sparse higher-order operators become then feasible.

Another trend in an effort to obtain practical realizations of nonlinear system modeling is that of using recursive system models. In this context, the biggest problem is that of maintaining the stability properties of the realized system. Some exciting results have been obtained in the recent past, but much needs to be still done.

Adaptive algorithms with nonlinear system models have attracted enormous attention in the last decade or so. Many breakthroughs have been achieved, especially in the areas of bringing down the computational complexity and devising numerically stable algorithms. Development of alternate structures for realizing polynomial system models has played a big role in these development. Further reduction in computational complexity is a desired and perhaps even necessary goal for making the adaptive Volterra filters feasible in a large class of applications. One way in which the authors believe this can be achieved is by developing exactly parameterized lattice structures for polynomial system models.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation through Grant No. MIP 8922146, by an IBM Departmental Grant, and by ESPRIT-BRA Project No. 7130.

References

[1] H. H. Chang, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient implementations of quadratic filters", *IEEE Trans. on Acoust., Speech, Signal processing*, Vol. ASSP-34, pp. 1511-1528, December 1986.

[2] F. Fnaiech and L. Ljung, "Recursive identification of bilinear systems," *International Journal of Control*, Vol. 45, No. 2, pp. 453-470, 1987.

[3] P. Fontanot and G. Ramponi, "A polynomial filter for preprocessing of mail address images," *Proc. IEEE Winter Workshop on Nonlinear DSP*, Tampere, Finland, January 1993.

[4] S. Haykin, *Adaptive Filter Theory* (Second Edition), Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[5] T. Koh and E. J. Powers, "An adaptive non-linear digital filter with lattice orthogonalization," *Proc. IEEE Int. Conf. Acoust. Speech, Signal Proc.*, Boston, Massachusetts, April 1983.

[6] J. Lee and V. J. Mathews, "A stability condition for certain bilinear systems," *Proc. IEEE Winter Workshop on Nonlinear DSP*, Tampere, Finland, January 1993.

[7] J. Lee and V. J. Mathews, "A fast, recursive least-squares second-order Volterra filter and its performance analysis," *IEEE Transactions on Signal Processing*, to appear in March 1993.

[8] J. Lee and V. J. Mathews, "On the extended RLS adaptive bilinear filters," to appear in *IEEE Int. Conf. Acoust., Speech, Signal Proc.*, Minneapolis, Minnesota, April 1993.

[9] Y. Lou, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters," *Journal Circuits, Systems, Signal Proc..*, Vol. 7, No. 2, pp. 253-273, 1988.

[10] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, Vol. 8, No. 3, pp. 10-26, July 1991.

[11] V. J. Mathews and Z. Xie, "Stochastic gradient adaptive filters with gradient adaptive step-sizes," *IEEE Int. Conf. Acoust., Speech, Signal Proc.*, Albuquerque, New Mexico, April 1990.

[12] B. G. Mertzios, G. L. Sicuranza, and A. N. Venetsanopoulos, "Efficient realizations of two-dimensional quadratic digital filters," *IEEE Trans. on Acoust., Speech, Signal Processing* Vol. ASSP-37, No. 5, pp. 765-768, May 1989.

[13] B. Picinbono, "Quadratic filters", in *Proc. ICASSP-82*, Paris, France, pp. 298-301, May 1982.

[14] I. Pitas and A. N. Venetsanopoulos, Nonlinear digital filters, Boston/ Dordrecht/ London: *Kluwer Academic Publishers*, 1990.

[15] G. Ramponi, G. L. Sicuranza, and W. Ukovich, "A computational method for the design of 2-D nonlinear Volterra filters," *IEEE Trans. on Circuits and Systems*, Vol. CAS-35, No. 9, pp. 1095-1102, September 1988.

[16] G. Ramponi, "Bi-impulse response design of isotropic quadratic filters," *IEEE Proceedings*, Vol. 78, No. 4, pp. 665-677, April 1990.

[17] G. L. Sicuranza, "Nonlinear digital filter realization by distributed arithmetic", *IEEE Trans. on Acoust., Speech, Signal Processing*, Vol. ASSP-33, No. 4, pp. 939-945, August 1985.

[18] G. L. Sicuranza and G. Ramponi, "Adaptive nonlinear digital filters using distributed arithmetic", *IEEE Trans. on Acoust., Speech, Signal Processing*, Vol. ASSP-34, No. 3, pp. 518-526, June 1986.

[19] G. L. Sicuranza, "Quadratic filters for signal processing," *Proc. IEEE*, Vol. 80, No. 8, pp. 1263-1285, August 1992.

[20] M. A. Syed and V. J. Mathews, "Lattice algorithms for recursive least squares adaptive Volterra filtering," submitted to *IEEE Trans. Circuits and Systems.*

[21] M. A. Syed and V. J. Mathews, "QR-Decomposition based algorithms for adaptive Volterra filtering," to appear in *IEEE Trans. Circuits and Systems.*

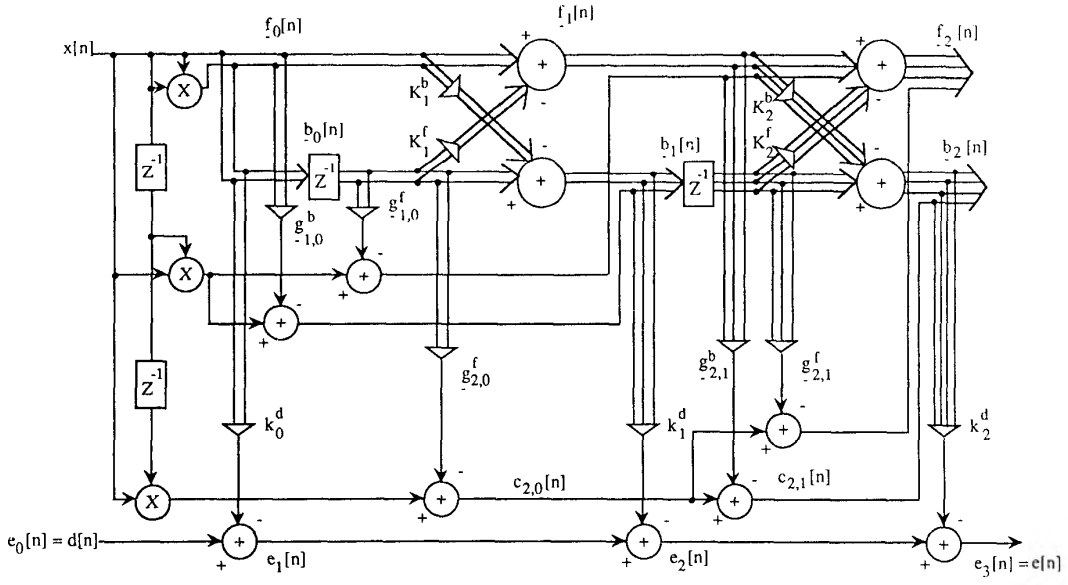

Figure 1. Lattice filter structure for a second order Volterra system.