FULLY AND PARTIALLY INTERPOLATED ADAPTIVE VOLTERRA FILTERS

Eduardo L. O. Batista, Orlando J. Tobias, and Rui Seara

LINSE – Circuits and Signal Processing Laboratory
Department of Electrical Engineering
Federal University of Santa Catarina
88040-900 – Florianópolis – SC – Brazil
E-mails: {dudu, orlando, seara}@linse.ufsc.br

ABSTRACT

This paper presents two simplified algorithms to implement adaptive Volterra filters. The central idea of the proposed approach is the use of sparse adaptive filters to reduce the number of coefficients to be adapted, which is the major drawback of adaptive Volterra filters. Such filters are obtained by removing some coefficients and then recreating them through an interpolating scheme. A second interpolated structure exploits also the block-size structure of the Volterra filter for coefficient reduction. Numerical simulation results are shown to ratify the good MSE behavior of the proposed structures.

1. INTRODUCTION

The use of a linear adaptive approach to deal with nonlinear filtering problems is common practice. This is due to the fact that simple and well-established tools for such filters are widely available. In contrast to these problems, nonlinear adaptive filters lead to better results, despite its higher computational complexity. However, due to the significant increase of the processing capacity of the modern digital signal processors (DSPs), the implementation of nonlinear adaptive structures is now feasible. In this context, Volterra adaptive filters have become an interesting tool for nonlinear applications, such as active control [1], acoustic echo canceling [2], identification and reduction of distortions of loudspeaker systems [3], and satellite-channel equalization [4]. Nevertheless, adaptive Volterra filters present a serious drawback: the total number of coefficients increases exponentially with the required memory size, restraining in certain cases its application. Some alternative approaches have been proposed to overcome such a drawback. Frequency domain Volterra filters [5], Volterra delay filters [6], and simplified Volterra filters [7] are examples of structures presenting a lower complexity.

The present paper proposes two reduced-complexity structures for implementing adaptive Volterra filters. The main idea of the proposed approach is to use a sparse adaptive Volterra filter along with an interpolator filter, such that this set behaves (almost) as a full adaptive filter. This interpolated approach has been applied successfully to linear filters [8], [9]. Here, we use either this technique on all or just on higher order blocks, which require a larger number of coefficients. This procedure gives rise to the fully and partially interpolated adaptive Volterra filters, respectively. Simulation results including linear adaptive, standard Volterra, and simplified Volterra [7] filters as well as the proposed structures are shown for performance comparison purposes.

2. VOLTERRA FILTER

The input-output relationship of a causal and discrete Volterra filter is given by [10]

$$y(n) = \sum_{m_1=0}^{N-1} h_1(m_1)\,x(n-m_1) + \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} h_2(m_1,m_2)\,x(n-m_1)x(n-m_2)\ldots + \sum_{m_1=0}^{N-1}\cdots\sum_{m_P=m_{P-1}}^{N-1} h_P(m_1,\cdots,m_P)\,x(n-m_1)\cdots x(n-m_P), \quad (1)$$

where $x(n)$ and $y(n)$ represent the input and output signals, respectively, $h_p(m_1,\ldots,m_p)$ denotes the pth-order coefficient, N is the memory size, and P is the filter order. From (1), this filter can be seen as the composition of a first order linear block, represented by h_1, followed by nonlinear blocks $h_2, h_3,\ldots,h_P$. By denoting the output of a pth-order block as $y_p(n)$, we can rewrite (1) as [1]

$$y(n) = \sum_{p=1}^{P} y_p(n), \quad (2)$$

with $y_p(n)$ given by

$$y_p(n) = \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1}\cdots\sum_{m_p=m_{p-1}}^{N-1} h_p(m_1,m_2,\ldots,m_p)\times\prod_{k=1}^{p} x(n-m_k). \quad (3)$$

Such decomposition is interesting for implementation purposes, permitting to establish strategies for complexity reduction. Thus, we can manipulate the whole filter or just the coefficient-demanding blocks.

The output of the first-order (linear) block of the Volterra filter can be expressed as

$$y_1(n) = \mathbf{h}_1^{\mathrm{T}}\mathbf{x}_1(n), \quad (4)$$

where $\mathbf{x}_1(n)$ denotes the first order input vector and $\mathbf{h}_1$ is the coefficient vector. For the second-order block, the input-output relationship is given by

$$y_2(n) = \mathbf{h}_2^{\mathrm{T}}\mathbf{x}_2(n), \quad (5)$$

where

$$\mathbf{x}_2(n) = [x^2(n)\ \ x(n)x(n-1)\ \ \cdots\ \ x(n)x(n-N+1)\ \ x^2(n-1)\ \ \cdots\ \ x^2(n-N+1)]^{\mathrm{T}}, \quad (6)$$

$$\mathbf{h}_2 = [h_2(0,0)\ \ h_2(0,1)\ \ \cdots\ \ h_2(0,N-1)\ \ h_2(1,1)\ \ \cdots\ \ h_2(N-1,N-1)]^{\mathrm{T}}. \quad (7)$$

By generalizing for the remaining blocks, we obtain the pth-order block output given by

$$y_p(n) = \mathbf{h}_p^{\mathrm{T}}\mathbf{x}_p(n). \quad (8)$$

By defining the Volterra input and coefficient vectors as

$$\mathbf{x}_{\mathrm{V}}(n) = [\mathbf{x}_1^{\mathrm{T}}(n),\ \mathbf{x}_2^{\mathrm{T}}(n),\ldots,\ \mathbf{x}_P^{\mathrm{T}}(n)]^{\mathrm{T}} \quad (9)$$

and

$$\mathbf{h}_{\mathrm{V}} = [\mathbf{h}_1^{\mathrm{T}}, \mathbf{h}_2^{\mathrm{T}}, \ldots, \mathbf{h}_P^{\mathrm{T}}]^{\mathrm{T}}, \quad (10)$$

the Volterra filter input-output relationship (1) can now be rewritten as

$$y(n) = \mathbf{h}_{\mathrm{V}}^{\mathrm{T}} \mathbf{x}_{\mathrm{V}}(n). \quad (11)$$

The computational burden of Volterra filters increases exponentially when the memory size is increased. We can note from (6) and (7), that the second-order block presents one coefficient for each of the second-order cross products of the input signal samples, resulting in a larger coefficient number than that required by the first-order block. For higher order blocks, this fact becomes more critical, since the coefficient number needed is proportional to the *p*th-order combinations of the input signal samples, with *p* representing the block order. Thus, depending on the memory size, the use of Volterra filters may be unfeasible. For complex adaptive algorithms, such as RLS [11] and affine projection [12], the use of Volterra filters becomes even more restricted. By denoting as $D_p(N)$ the coefficient number for each *p*th-order block of a Volterra filter with memory size given by N, we get [1]

$$D_p(N) = \frac{(N+p-1)!}{(N-1)!\,p!}. \quad (12)$$

Then, the total number of coefficients $D_{\mathrm{V}}(N,P)$ of a Volterra filter, with order P and a memory size N, is given by the sum of coefficients of all the involved blocks, resulting in

$$D_{\mathrm{V}}(N,P) = \frac{(N+P)!}{N!P!} - 1. \quad (13)$$

3. FULLY INTERPOLATED ADAPTIVE VOLTERRA FILTER

By reducing the memory size of a Volterra filter using sparse filters, we can achieve a substantial reduction in the coefficient number. The filter sparsity is a function of the interpolation factor, denoted by L. Thus, for instance, the sparse first-order input vector $\mathbf{x}_{1\mathrm{s}}(n)$ is obtained by removing $(L-1)$ samples from each *L* consecutive ones of the original input vector [8], resulting in

$$\mathbf{x}_{1\mathrm{s}}(n) = \left\{x(n)\ \ x(n-L)\ \ x(n-2L)\ \ \cdots\ \ x[n-(N_{\mathrm{s}}-1)L]\right\}^{\mathrm{T}}, \quad (14)$$

with N_{s} denoting the memory size of the sparse filter, given by

$$N_{\mathrm{s}} = \left\lfloor \frac{N-1}{L} \right\rfloor + 1, \quad (15)$$

where $\lfloor \cdot \rfloor$ represents the truncation operation.

The corresponding sparse first-order coefficient vector is expressed as

$$h_{1\mathrm{s}} = \{h_1(0)\ \ h_1(L)\ \ h_1(2L)\ \ \cdots\ \ h_1[(N_{\mathrm{s}}-1)L]\}^{\mathrm{T}}. \quad (16)$$

The remaining higher-order input and coefficient vectors are obtained from the first-order ones as presented in Section 2.

However, by using a sparse filter we are considering a suboptimal structure with significant loss of performance. An input-signal interpolator filter is then used to reduce the effect of the removed samples. This procedure leads to the interpolated Volterra filter, depicted in Fig. 1. In this figure $\mathbf{h}_{\mathrm{Vi}}$ is the sparse Volterra filter and $\mathbf{I}$ denotes the impulse response of the interpolator filter, represented by an *M*-coefficient FIR filter, defined by $\mathbf{I} = [i_0\ i_1 \cdots i_{M-1}]^{\mathrm{T}}$. The input signal and its interpolated version are represented by $x(n)$ and $x_{\mathrm{i}}(n)$, respectively, where

$$x_{\mathrm{i}}(n) = \sum_{j=0}^{M-1} i_j x(n-j). \quad (17)$$

The signal $z(n)$ is a measurement noise, uncorrelated with any other signal in the system. The sparse filter output signal and the error signal are represented by $y(n)$ and $e(n)$, respectively (see Fig. 1). Hence, the first-order input vector for the sparse-interpolated Volterra filter is given by

$$\mathbf{x}_{1\mathrm{i}}(n) = \{x_{\mathrm{i}}(n)\ \ x_{\mathrm{i}}(n-L)\ \ x_{\mathrm{i}}(n-2L)\ \ \cdots\ \ x_{\mathrm{i}}[n-(N_{\mathrm{s}}-1)L]\}^{\mathrm{T}}. \quad (18)$$

By generating the higher-order input vectors from (18), the overall interpolated input vector denoted by $\mathbf{x}_{\mathrm{Vi}}(n)$ is constructed in the same way as (9).

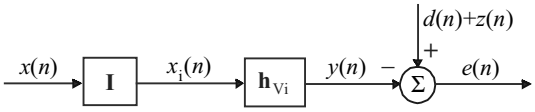

Fig. 1. Block diagram of the interpolated Volterra filter.

The entire structure of the interpolated Volterra filter can be viewed as a full Volterra filter with coefficient constraints. Considering it on the adaptive context, we have a constrained Volterra filter with the need to adapt only the coefficients from the sparse one, which is very interesting because of the Volterra complexity drawback. The adaptive algorithm then performs the adjustment of the coefficients towards the optimum value. Thus, the coefficient update equation for the adaptive LMS interpolated Volterra filter is given by

$$\mathbf{h}_{\mathrm{Vi}}(n+1) = \mathbf{h}_{\mathrm{Vi}}(n) + 2\mu e(n)\mathbf{x}_{\mathrm{Vi}}(n), \quad (19)$$

where $\mathbf{h}_{\mathrm{Vi}}(n)$ is the coefficient vector corresponding to the interpolated input vector $\mathbf{x}_{\mathrm{Vi}}(n)$. Table 1 compares the number of coefficients to be adapted by both the standard and the interpolated approaches, highlighting the complexity difference between these structures.

Table 1. Number of coefficients: standard *versus* interpolated implementation

N	P	L	N_{s}	$D_{\mathrm{V}}(N,P)$	$D_{\mathrm{V}}(N_{\mathrm{s}},P)$	Reduction (%)
3	2	2	2	9	5	44.44%
10	2	2	5	65	20	69.23%
25	2	2	13	350	104	70.29%
15	3	2	8	815	164	79.88%
30	3	2	15	5455	815	85.06%
30	3	4	8	5455	164	96.99%

It is well known that the interpolated filters do not have good performance for modeling plants with weak correlation between its coefficients [8], [9]. For this case, a better solution is the use of partially interpolated Volterra filters, which are described in the next section.

4. PARTIALLY INTERPOLATED ADAPTIVE VOLTERRA FILTER

A second structure for implementing adaptive Volterra filters is obtained by using the interpolated approach only on higher order blocks. Fig. 2 depicts a second-order partially interpolated Volterra filter. In this figure, $\mathbf{h}_1$ and $\mathbf{h}_{2\mathrm{i}}$ represent the first-order and the sparse second-order blocks, respectively, with $y_1(n)$ and $y_2(n)$ as their corresponding outputs.

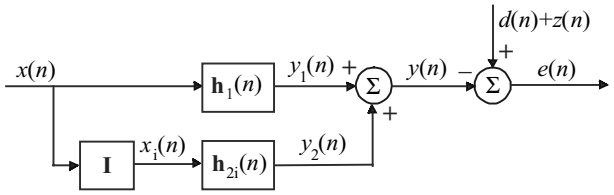

Fig. 2. Block diagram of a second-order partially interpolated Volterra filter.

For the structure of Fig. 2, the filter coefficients are updated by using the LMS algorithm. Thus,

$$\begin{aligned}\mathbf{h}_1(n+1) &= \mathbf{h}_1(n) + 2\mu_1 e(n)\mathbf{x}_1(n),\\ \mathbf{h}_{2i}(n+1) &= \mathbf{h}_{2i}(n) + 2\mu_2 e(n)\mathbf{x}_{2i}(n),\end{aligned} \tag{20}$$

where $\mathbf{x}_1(n)$ and $\mathbf{x}_{2i}(n)$ are the first-order input vector and the second-order sparse input vector, respectively.

The total number of coefficients for this filter is obtained by adding the coefficients of each block according to (12). Note that the interpolated blocks have their memory size reduced as stated by (15). In Table 2, the coefficient number to be adapted is compared considering each structure. It can be noted that the computational savings of the partially interpolated structure are close to the one achieved for the fully interpolated filter.

Table 2. Number of coefficients for $L = 2$

Memory	Order	Coefficient Number		
		Volterra	Fully Interp.	Part. Interp.
3	2	9	5	6
10	2	65	20	25
25	2	350	104	116
50	2	1325	350	375
10	3	285	55	60
25	3	3275	559	571

5. SIMULATION RESULTS

To illustrate the behavior of the proposed approaches, some numerical simulations are presented by considering a system identification problem. The proposed structures are compared with standard Volterra and FIR linear filters for MSE performance. In Examples 1 and 2, such structures are also compared with the simplified implementations from [7]. The latter consists of sparse second-order Volterra filters obtained by setting to zero the coefficients far from the main diagonal [7], resulting in the following input-output relation for the second-order block. Thus,

$$y_2(n) = \sum_{k=0}^{K}\sum_{m=0}^{N-k-1} h_2(m,m+k)x(n-m)x(n-m-k)\,, \tag{21}$$

where $y_2(n)$ is the second-order block output signal, $x(n)$ denotes the input signal, $h_2(m,m+k)$ represents the second-order coefficients, and N is the memory size. In (21), we also have the K factor that determines the amount of coefficients to be set to zero. By choosing $K = N-1$ we have the standard Volterra filter. The step-size values are related to $\mu_{\max}$, maximum allowable value for which the adaptive algorithm converges. For the proposed approaches, an interpolation factor $L = 2$ is used. The interpolator impulse response is given by $\mathbf{I} = [0.25\ 0.5\ 0.25]^{\mathrm{T}}$ and the variance of $z(n)$ is $\sigma_z^2 = 0.001$.

Example 1: For this example, the plant is the length-11 vector $[-0.05\ -0.10\ 0.00\ 0.15\ 0.32\ 0.40\ 0.32\ 0.15\ 0.00\ -0.10\ -0.05]^{\mathrm{T}}$ (linear filter), followed by a memoryless nonlinearity. The desired signal is obtained by $d(n) = y_{\mathrm{f}}(n) + 0.3y_{\mathrm{f}}^2(n)$, where $y_{\mathrm{f}}(n)$ is the linear filter output. The input signal is white, Gaussian with unit variance. The step-size value is $\mu = 0.2\mu_{\max}$, with $\mu_{\max}$ obtained from $\mu_{\max} = 1/\{3\mathrm{tr}[\mathbf{R}]\}$ [11], where $\mathbf{R}$ is the autocorrelation matrix of the input vector. Fig. 3 shows the MSE curves obtained by Monte Carlo simulation (average of 200 independent runs), considering the linear filter (11 coefficients and $\mu_{\max} = 0.03$), standard Volterra filter (77 coefficients and $\mu_{\max} = 0.0033$), fully interpolated Volterra filter (27 coefficients and $\mu_{\max} = 0.05$), and partially interpolated one (32 coefficients and $\mu_{\max} = 0.04$). From this figure we can verify that the partially interpolated structure has a minimum MSE performance close to the standard Volterra implementation. In Fig. 4, the partially interpolated Volterra filter is compared with three different simplified Volterra filter [7] implementations, with $K = 2$ (32 coefficients and $\mu_{\max} = 0.006$), $K = 3$ (41 coefficients and $\mu_{\max} = 0.005$) and $K = 4$ (49 coefficients and $\mu_{\max} = 0.0046$). For these cases, the proposed structure presents a better performance with a smaller coefficient number.

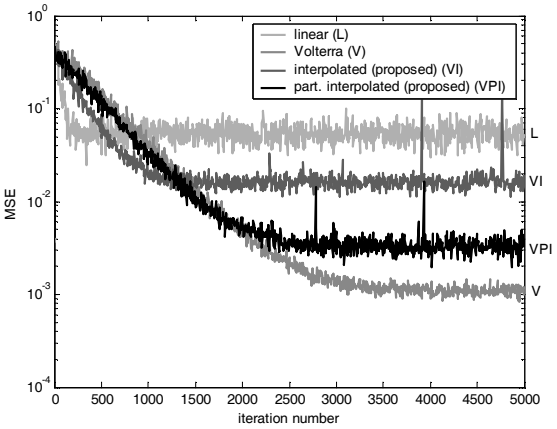

Fig. 3. Example 1. MSE evolution (average of 200 runs).

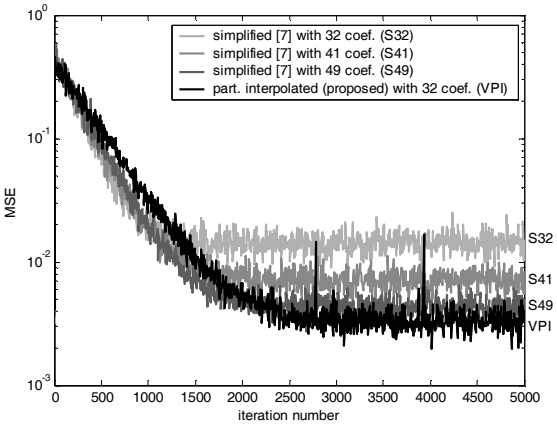

Fig. 4. Example 1. MSE evolution (average of 200 runs).

Example 2: In this example, we use a similar plant to the one from Example 1, but now the linear part is given by $[0.40\ -0.20\ 0.10\ 0.15\ -0.05\ -0.10\ 0.00\ 0.20\ 0.30\ -0.25\ -0.02]^{\mathrm{T}}$. It can be noted that such a plant has coefficients with low cross-correlation, impairing the use of an interpolated solution. The step-size values used are the same as in Example 1, unless for the simplified Volterra filter implementations, which

use $K = 7$ (67 coefficients and $\mu = 0.2\mu_{max} = 0.001$) and $K = 9$ (74 coefficients and $\mu = 0.2\mu_{max} = 0.00068$). Figs. 5 and 6 illustrate the Monte Carlo simulation (average of 200 runs) results. As expected, we can note that the interpolated approach exhibits a worse performance than the linear one, due to the characteristics of the plant used. The partially interpolated structure has slightly better performance than the linear one, according to the conjectures made in Section 4. By comparing the simplified Volterra results with those obtained by the proposed structures shown in Fig. 6, we note that the partially interpolated approach with 32 coefficients has a performance close to the simplified implementation with 67 coefficients.

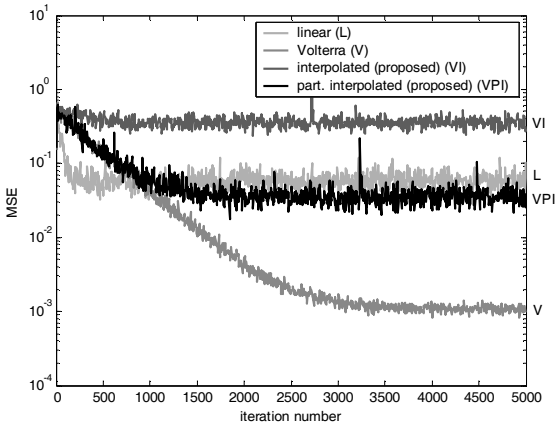

Fig. 5. Example 2. MSE evolution (average of 200 runs).

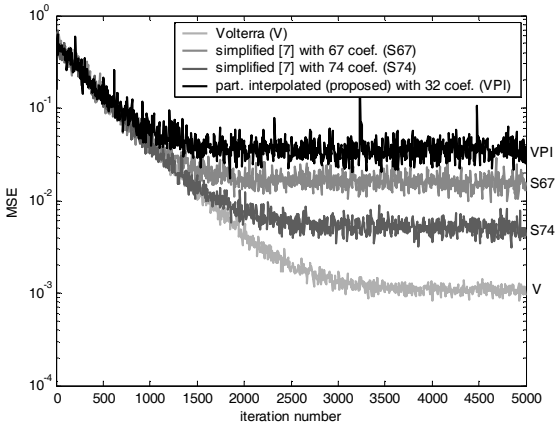

Fig. 6. Example 2. MSE evolution (average of 200 runs).

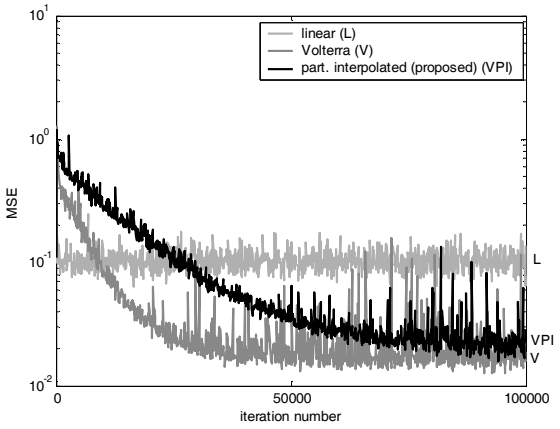

Fig. 7. Example 3. MSE evolution (average of 200 runs).

Example 3: In this example, the plant output is given by $d(n) = 2\tanh[2y_f(n)]$, where $y_f(n)$ is the output of a linear FIR filter with coefficients given by $[0.10\ 0.30\ 0.32\ 0.30\ 0.10]^T$. The adaptive Volterra filters now have first- and third-order blocks and a memory size equal to 5. The input signal is white, Gaussian with variance $\sigma_x^2 = 0.5$. The step-size values are: $\mu = 0.0067$ for the linear filter (5 coefficients), $\mu = 0.0005$ for the standard Volterra filter (40 coefficients), and $\mu = 0.00015$ for the partially interpolated (15 coefficients). Fig. 7 shows the obtained MSE curves. Again, we confirm a satisfactory performance for the partially interpolated Volterra filter.

6. CONCLUSIONS

In this paper, simplified structures for the implementation of adaptive Volterra filters are considered. The use of an interpolated approach permits the implementation of suboptimal solutions leading to significant reductions in the required computational burden. In addition, a satisfactory MSE performance is verified for the proposed structures.

REFERENCES

[1] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. on Signal Processing*, vol. 49, no. 8, pp. 1667-1676, Aug. 2001.

[2] A. Stenger, L. Trautmann, and R. Rabenstein, "Nonlinear acoustic echo cancellation with second order adaptive Volterra filters," *IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Phoenix, USA, vol. 2, Mar. 1999, pp. 15-19.

[3] M. Tsujikawa, T. Shiozaki, Y. Kajikawa, and Y. Nomura, "Identification and elimination of second-order nonlinear distortion of loudspeaker systems using Volterra filters," *IEEE Int. Symp. on Circ. and Systems*, Geneva, Switzerland, vol. 5, May 2000, pp. 28-31.

[4] A. Gutierrez and W. E. Ryan, "Performance of adaptive Volterra equalizers on nonlinear satellite channels," *IEEE Int. Conf. on Communications*, Seattle, USA, vol. 1, June 1995, pp. 19-22.

[5] M. J. Reed and M. O. J. Hawksford, "Efficient implementation of the Volterra filter," *IEE Proc.-Vis. Image Signal Processing*, vol. 147, no. 2, pp. 109-114, Apr. 2000.

[6] L. Tan and J. Jiang, "System modeling using a second-order Volterra delay filter," *IEEE 39th Midwest Symp. on Circuits and Systems,* Ames, USA, vol. 3, Aug. 1996, pp. 18-21.

[7] A. Fermo, A. Carini, and G. L. Sicuranza, "Simplified Volterra filters for acoustic echo cancellation in GSM receivers," *X European Signal Proc. Conf.*, Tampere, Finland, Sep. 2000.

[8] Y. Neuvo, C. Y. Dong, and S. K. Mitra, "Interpolated finite impulse response digital filters," *IEEE Trans. Acoustics, Speech, Signal Process.*, vol. 32, pp. 563-570, Jun. 1984.

[9] O. J. Tobias and R. Seara, "Analytical model for the mean weight behavior of adaptive interpolated-FIR filters using the constrained filtered LMS algorithm," *Proc. IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Lake Louise, Canada, Oct. 2000, pp. 272-277.

[10] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, vol. 8, pp. 10-26, July 1991.

[11] B. Farhang-Boroujeny, *Adaptive Filters Theory and Applications*, John Wiley & Sons Ltd., 1999.

[12] M. Rupp, "A family of filter algorithms with decorrelating properties," *IEEE Trans. on Signal Processing*, vol. 46, no. 3, pp. 771-774, Mar. 1998.