FULLY ADAPTIVE LMS/NLMS INTERPOLATED VOLTERRA FILTERS WITH REMOVED BOUNDARY EFFECT

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-mail: {dudu, orlando, seara}@linse.ufsc.br

ABSTRACT

Aiming to expand the applicability of adaptive Volterra filters, a large number of reduced complexity implementations have been discussed in the open literature. Contributing to this goal, this paper presents a fully LMS/NLMS adaptive approach for implementing interpolated Volterra filters with removed boundary effect. The main aim here is to combine a fully adaptive interpolated approach, a boundary effect removal procedure, and the LMS/NLMS algorithm to give birth to an algorithm with very good steady state and transient performance. Numerical simulations confirm the effectiveness of the proposed approach.

1. INTRODUCTION

Over the last two decades, adaptive Volterra filters have been used in several nonlinear applications, such as control of nonlinear noise processes [1], acoustic echo canceling [2], compensation of nonlinear effects in OFDM transmitters [3], among others. In this context, both the increasing processing capacity of modern digital signal processors (DSPs) and significant research efforts have decisively contributed to overcome the main problem of implementing digital Volterra filters, which is its inherent computational burden. As a result, the search for Volterra structures with lower computational burden has been strongly demanded. Examples of such structures are the simplified [4], sparse [5], and frequency-domain [6] implementations. Additionally, interpolated Volterra filters also form an important class of structures with reduced implementation complexity [7]. The interpolated approach, which was originally considered for implementing linear FIR filters [8], uses a cascaded filtering structure composed of a sparse filter, with reduced coefficient number, and an interpolator filter, whose purpose is to recreate the zeroed coefficients of the sparse filter [7], [8]. In the Volterra case, the interpolated structure consists of a linear input interpolator followed by a sparse Volterra filter, leading to a structure having considerable reduction in complexity [7]. However, such computational savings come at the expense of increased minimum mean-square error (MSE). Such poor performance can be improved by using a fully adaptive interpolated structure [9] instead of a sparse filter structure [7]. Another point for performance improvement in interpolated structures is to remove the boundary or border effect [10], since it is a considerable source of performance degradation in many cases [10]. In this context, the present paper introduces an interpolated Volterra implementation combining a fully adaptive structure with a boundary effect removal procedure, aiming to improve the performance. In addition, an LMS/NLMS adaptive setup is adopted to enhance the convergence characteristics at the expense of a relatively small computational increment. Through numerical simulations, we verify the performance of the proposed structure.

This paper is organized as follows. Section 2 presents the interpolated Volterra filter and its main characteristics. Section 3 discusses briefly the generalized boundary effect removal procedure for interpolated Volterra filters. In Section 4, the fully adaptive LMS/NLMS interpolated Volterra structure with removed boundary effect is derived. Section 5 presents the results of numerical simulations. Finally, the conclusions of this paper are presented in Section 6.

2. INTERPOLATED VOLTERRA FILTERS

Figure 1 shows the block diagram of an interpolated Volterra filter [7]. Such a filter is composed of an input interpolator filter $\mathbf{g}$, with memory size M and coefficient vector $\mathbf{g} = [g(0)\ g(1)\ \cdots\ g(M-1)]^{\mathrm{T}}$, cascaded with a sparse Volterra filter denoted by $\mathbf{h}_{\mathrm{Vs}}$ with memory size N and order P. The block structure of the Volterra filter is highlighted in the figure by the dashed box, in which each pth-order sparse block is denoted by $\mathbf{h}_{ps}$, having output signals given by $\hat{y}_\ell(n)$ for $\ell = 1, 2, \ldots, P$ and interpolated input vectors by $\tilde{\mathbf{x}}_p(n)$. Additionally, $x(n)$ and $\hat{y}(n)$ represent, respectively, the input and output signals of the interpolated structure.

The first-order sparse coefficient vector is obtained by setting $L-1$ of each L coefficients to zero [7], with L denoting the sparsity or interpolation factor. For instance, by considering the full $N \times 1$ first-order coefficient vector $\mathbf{h}_1 = [h_1(0)\ h_1(1)\ \cdots\ h_1(N-1)]^{\mathrm{T}}$, its corresponding $N \times 1$ sparse vector is given by

$$\mathbf{h}_{1\mathrm{s}} = \{h_1(0)\ 0\ \cdots\ h_1(L)\ 0\ \cdots\ h_1[(N_{\mathrm{s}}-1)L]\ 0\ \cdots\ 0\}^{\mathrm{T}} \qquad (1)$$

with $N_{\mathrm{s}} = \lfloor (N-1)/L \rfloor + 1$, where $\lfloor \cdot \rfloor$ represents the truncation operation, and L is the decimation factor. The first-order input vector is given by

Orlando J. Tobias is also with the Electrical Engineering and Telecom. Dept., Regional University of Blumenau (FURB), Blumenau, SC, Brazil. This work was supported in part by the National Council for Scientific and Technological Development (CNPq).

$$\tilde{\mathbf{x}}_1(n) = [\tilde{x}(n)\ \ \tilde{x}(n-1)\ \ \tilde{x}(n-2)\ \ \cdots\ \ \tilde{x}(n-N+1)]^{\mathrm{T}}. \quad (2)$$

As discussed in [7], (2) can also be expressed as

$$\tilde{\mathbf{x}}_1(n) = \mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n) \quad (3)$$

where $\mathbf{x}_{\mathrm{e}}(n) = [x(n)\ \ x(n-1)\ \ \cdots\ \ x(n-N-M+2)]^{\mathrm{T}}$ is the extended input vector (with $N+M-1$ samples of the input signal), and $\mathbf{G}$ is the $[(N+M-1)\times N]$-dimensional interpolation matrix [7] given by

$$\mathbf{G} = \begin{bmatrix} g(0) & 0 & 0 & \cdots & 0 \\ g(1) & g(0) & 0 & \cdots & 0 \\ g(2) & g(1) & g(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(M-1) & g(M-2) & g(M-3) & \cdots & g(0) \\ 0 & g(M-1) & g(M-2) & \cdots & g(1) \\ 0 & 0 & g(M-1) & \cdots & g(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & g(M-1) \end{bmatrix}. \quad (4)$$

Thus, the output signal of the interpolated first-order block is

$$\hat{y}_1(n) = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}\mathbf{h}_{1\mathrm{s}} = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{h}_{1\mathrm{i}} \quad (5)$$

with $\mathbf{h}_{1\mathrm{i}} = \mathbf{G}\mathbf{h}_{1\mathrm{s}}$. Regarding higher-order blocks, the sparse coefficient vectors are obtained by setting those coefficients having at least one index not multiple of L to zero [7]. The *p*th-order input vectors are obtained recursively from the general form by

$$\tilde{\mathbf{x}}_p(n) = \tilde{\mathbf{x}}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}(n) \quad (6)$$

where $\otimes$ denotes the Kronecker product. Thereby, the output signal of the interpolated second-order block is given by [7]

$$\begin{aligned} \hat{y}_2(n) &= \tilde{\mathbf{x}}_2^{\mathrm{T}}(n)\mathbf{h}_{2\mathrm{s}} = [\mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n) \otimes \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)][\mathbf{G} \otimes \mathbf{G}]\mathbf{h}_{2\mathrm{s}} \\ &= \mathbf{x}_{2\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}_2\mathbf{h}_{2\mathrm{s}} \end{aligned} \quad (7)$$

with $\mathbf{x}_{2\mathrm{e}}(n) = \mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{x}_{\mathrm{e}}(n)$ and $\mathbf{G}_2 = \mathbf{G} \otimes \mathbf{G}$. From (7), we verify that the equivalent vector for the second-order block is $\mathbf{h}_{2\mathrm{i}} = \mathbf{G}_2\mathbf{h}_{2\mathrm{s}}$. Now, generalizing the above expressions for a *p*th-order interpolated block, we have

$$\hat{y}_p(n) = \mathbf{x}_{p\mathrm{e}}(n)\mathbf{G}_p\mathbf{h}_{p\mathrm{s}} = \mathbf{x}_{p\mathrm{e}}(n)\mathbf{h}_{p\mathrm{i}} \quad (8)$$

with $\mathbf{G}_p = \mathbf{G} \otimes \mathbf{G}_{p-1}$ and $\mathbf{h}_{p\mathrm{i}} = \mathbf{G}_p\mathbf{h}_{p\mathrm{s}}$. Moreover, one has $\mathbf{x}_{\mathrm{Ve}}(n) = [\mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\ \ \mathbf{x}_{2\mathrm{e}}^{\mathrm{T}}(n)\ \cdots\ \mathbf{x}_{P\mathrm{e}}^{\mathrm{T}}(n)]^{\mathrm{T}}$ with $\mathbf{x}_{p\mathrm{e}}(n) = \mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{x}_{(p-1)\mathrm{e}}(n)$ and $\mathbf{h}_{\mathrm{Vi}} = [\mathbf{h}_{1\mathrm{s}}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\ \ \mathbf{h}_{2\mathrm{s}}^{\mathrm{T}}\mathbf{G}_2^{\mathrm{T}}\ \cdots\ \mathbf{h}_{P\mathrm{s}}^{\mathrm{T}}\mathbf{G}_P^{\mathrm{T}}]^{\mathrm{T}}$, which results in the output of the interpolated structure given by

$$\hat{y}(n) = \mathbf{x}_{\mathrm{Ve}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vi}}. \quad (9)$$

As described in [7], the purpose of the interpolator filter is to recreate the zeroed coefficients of the sparse coefficient vector in the resulting equivalent coefficient vector. Consequently, the number of coefficients M of the interpolator must be chosen in function of L as

$$M(L) = 1 + 2(L-1) = 2L-1. \quad (10)$$

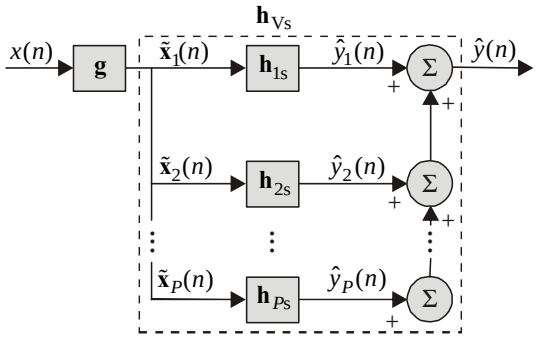

Figure 1 – Block diagram of an interpolated Volterra filter.

For instance, an interpolation factor $L=2$ results in $M=3$ and the interpolator filter is $\mathbf{g} = [g_0\ \ g_1\ \ g_2]^{\mathrm{T}}$. Thus, for a sparse Volterra filter with $N=5$, the first-order coefficient vector is obtained from (1) and, by using $\mathbf{g} = [0.5\ \ 1\ \ 0.5]^{\mathrm{T}}$ (linear interpolator [8]), the first-order equivalent coefficient vector, obtained from $\mathbf{h}_{1\mathrm{i}} = \mathbf{G}_1\mathbf{h}_{1\mathrm{s}}$, is given by

$$\begin{aligned} \mathbf{h}_{1\mathrm{i}} = [&\underline{0.5h_1(0)}\ \ h_1(0)\ \ \boxed{0.5h_1(0)+0.5h_1(2)}\ \ h_1(2) \\ &\boxed{0.5h_1(2)+0.5h_1(4)}\ \ h_1(4)\ \ \underline{0.5h_1(4)}]^{\mathrm{T}}. \end{aligned} \quad (11)$$

Note, from (11), that two facts can be observed: (i) the zeroed coefficients of the sparse filter are recreated by interpolation (boxed ones) and (ii) new coefficients are created (underlined) as a boundary effect [7]. A similar situation occurs for the higher-order blocks [10].

3. GENERALIZED BOUNDARY EFFECT REMOVAL FOR INTERPOLATED VOLTERRA FILTERS

The boundary effect leads to a substantial loss of performance in different cases [7], [10], [11]. In [10], a procedure for removing such an effect in interpolated FIR (IFIR) and interpolated Volterra structures is discussed, considering the case for $L=2$. Recently in [11], the boundary effect removal procedure has been extended to any value of L and for the linear IFIR case. Here, the generalized procedure from [11] is extended to an interpolated Volterra structure using the following generalized transformation matrix:

$$\mathbf{T} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ \underbrace{0 \cdots 0}_{L-1\text{ columns}} & & & \underbrace{0\ 0 \cdots 1}_{N\text{ columns}} & & & & \underbrace{0 \cdots 0}_{L-1\text{ columns}} & & \end{bmatrix} \quad (12)$$

with dimensions $N \times (N+M-1)$. The input vector for the first-order sparse block (2) is replaced by a modified version, given by

$$\tilde{\mathbf{x}}_1'(n) = \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}_1(n) = \mathbf{G}'^{\mathrm{T}}\mathbf{x}_1(n) \quad (13)$$

where $\mathbf{G}' \equiv \mathbf{T}\mathbf{G}$. The input vectors for the nonlinear sparse blocks are generated similarly to (6), but now considering (13), resulting in

$$\tilde{\mathbf{x}}_p'(n) = \tilde{\mathbf{x}}_1'(n) \otimes \tilde{\mathbf{x}}_{p-1}'(n). \quad (14)$$

Finally, defining

$$\tilde{\mathbf{x}}'_{\mathrm{V}}(n)=[\tilde{\mathbf{x}}_1'^{\mathrm{T}}(n)\ \tilde{\mathbf{x}}_2'^{\mathrm{T}}(n)\ \cdots\ \tilde{\mathbf{x}}_P'^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{15}$$

the output of the removed boundary effect interpolated Volterra (RBEIV) filter is given by

$$\hat{y}(n)=\tilde{\mathbf{x}}_{\mathrm{V}}'^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vs}}. \tag{16}$$

Moreover, as described in [11], the implementation of the boundary effect removal procedure adds a small computational burden of about $2L-2$ operations per sample, being negligible since L is generally small.

4. FULLY ADAPTIVE LMS/NLMS RBEIV FILTERS

As discussed in [9], the implementation of fully adaptive interpolated Volterra structures considering adaptive interpolators is not a straightforward task, being more complex if the boundary effect removal is considered. Moreover, because of the high computational burden and the well-known slow convergence behavior of Volterra filters [12], the use of faster and relatively simple algorithms for adaptation, such as the normalized LMS (NLMS) algorithm, is very attractive. In this section, expressions for also adapting the interpolator filter of the RBEIV structure are developed. For the input linear interpolator, the adopted algorithm is the LMS due to the smaller number of coefficients as well as for mathematical simplicity. On the other hand, the sparse Volterra filter is adapted using the NLMS algorithm aiming to improve the overall convergence speed. Then, the goal is to obtain an adaptive Volterra implementation with better convergence rate and steady-state performance, while keeping the computational savings obtained by using an interpolated Volterra structure.

4.1. LMS Interpolator Update

To facilitate the derivation of the LMS update expression for the interpolator in the RBEIV structure, a representation of the input-output relationship of such a structure based on an input matrix is used as in [9]. Thus, (13) can be rewritten as

$$\tilde{\mathbf{x}}_1'(n)=\mathbf{X}_1'^{\mathrm{T}}(n)\,\mathbf{g} \tag{17}$$

where $\mathbf{X}_1'(n)$ is the boundaryless first-order input matrix. Such a matrix has the form of a non-square Hankel matrix [13] with dimensions $M\times N$, having the first column given by

$$\mathbf{X}_1'^{\mathrm{T}}(n)\Big|_{\text{first-column}}=[\underbrace{0\ \cdots\ 0}_{L-1}\ x(n)\ \ x(n-1)\ \cdots\ x(n-M+L)]^{\mathrm{T}} \tag{18}$$

and last row as

$$\mathbf{X}_1'^{\mathrm{T}}(n)\Big|_{\text{last-row}}=[x(n-M+L)\ \cdots\ x(n-N+1)\ \underbrace{0\ \cdots\ 0}_{L-1}]. \tag{19}$$

Vectors (18) and (19) completely define $\mathbf{X}_1'(n)$, since a Hankel matrix has equal elements along any diagonal that slopes from southwest to northeast. From (17), the output of the first-order block of the RBEIV structure is given by

$$\hat{y}_1(n)=\mathbf{g}^{\mathrm{T}}\mathbf{X}_1'(n)\mathbf{h}_{1\mathrm{s}}. \tag{20}$$

By using (14) and (17), the output of the second-order block is

$$\begin{aligned}\hat{y}_2(n)&=\tilde{\mathbf{x}}_2'^{\mathrm{T}}(n)\mathbf{h}_{2\mathrm{s}}\\&=(\mathbf{g}^{\mathrm{T}}\otimes\mathbf{g}^{\mathrm{T}})[\mathbf{X}_1'(n)\otimes\mathbf{X}_1'(n)]\mathbf{h}_{2\mathrm{s}}\\&=\mathbf{g}_2^{\mathrm{T}}\mathbf{X}_2'(n)\mathbf{h}_{2\mathrm{s}}\end{aligned} \tag{21}$$

with $\mathbf{g}_2=\mathbf{g}\otimes\mathbf{g}$ and $\mathbf{X}_2'(n)=\mathbf{X}_1'(n)\otimes\mathbf{X}_1'(n)$. In general, one has

$$\hat{y}_p(n)=\mathbf{g}_p^{\mathrm{T}}\mathbf{X}_p'(n)\mathbf{h}_{p\mathrm{s}} \tag{22}$$

with $\mathbf{g}_p=\mathbf{g}\otimes\mathbf{g}_{p-1}$ and $\mathbf{X}_p'(n)=\mathbf{X}_1'(n)\otimes\mathbf{X}_{p-1}'(n)$. The interpolator filter update using the LMS algorithm is given by

$$\mathbf{g}(n+1)=\mathbf{g}(n)-\mu_{\mathrm{g}}\nabla_{\mathbf{g}}e^2(n) \tag{23}$$

where μ_{g} is the step size, $\nabla_{\mathbf{g}}$ is the gradient w.r.t. interpolator coefficients, and $e(n)=d(n)-\hat{y}(n)$ denotes the error signal with $d(n)$ characterizing the desired signal. From the above definitions and making analogous derivations to [9], the following update expression for the coefficients of the interpolator is obtained:

$$\mathbf{g}(n+1)=\mathbf{g}(n)+2\mu_{\mathrm{g}}e(n)\sum_{p=1}^{P}p\,[\mathbf{X}_1'(n)\otimes\tilde{\mathbf{x}}_{p-1}'^{\mathrm{T}}(n)]\mathbf{h}_{p\mathrm{s}}(n) \tag{24}$$

with $\tilde{\mathbf{x}}_0'(n)=1$ for $p=1$. Since the interpolator is time-varying, $\tilde{\mathbf{x}}_{p-1}'(n)$ in (24) should be determined at each iteration, leading to a larger computational burden. However, such a vector can be approximately obtained by reusing its previous values and assuming slow variation of the interpolator coefficients (similar to [9]). This approach considerably reduces the computational burden at the cost of small convergence issues, which can be easily counterbalanced by properly selecting the value of the step size μ_{g} [9].

4.2. NLMS Sparse Volterra Filter Update

The NLMS algorithm for adapting the sparse Volterra filter from the interpolated structure is obtained by using a constrained optimization approach for minimizing the Euclidean norm of

$$\delta\mathbf{h}_{\mathrm{Vs}}(n+1)=\mathbf{h}_{\mathrm{Vs}}(n+1)-\mathbf{h}_{\mathrm{Vs}}(n) \tag{25}$$

subject to

$$\tilde{\mathbf{x}}_{\mathrm{V}}'^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vs}}(n+1)=d(n) \tag{26}$$

and

$$\mathbf{C}^{\mathrm{T}}\mathbf{h}_{\mathrm{Vs}}(n+1)=\mathbf{f} \tag{27}$$

where, similarly to [8] and [14], $\mathbf{C}$ is a constraint matrix due to the sparsity of $\mathbf{h}_{\mathrm{Vs}}$ and $\mathbf{f}$ is the response vector to the constraints (in this case, a vector of zeros). Then, using the Lagrange multiplier method, the following cost function is obtained:

$$\begin{aligned}J_{\mathbf{h}_{\mathrm{Vs}}}(n)=&\left\|\delta\mathbf{h}_{\mathrm{Vs}}(n+1)\right\|^2+\theta_1\left[d(n)-\mathbf{h}_{\mathrm{Vs}}^{\mathrm{T}}(n+1)\tilde{\mathbf{x}}_{\mathrm{V}}'(n)\right]\\&+\boldsymbol{\theta}_2^{\mathrm{T}}\left[\mathbf{C}^{\mathrm{T}}\mathbf{h}_{\mathrm{Vs}}(n+1)-\mathbf{f}\right]\end{aligned} \tag{28}$$

where θ_1 and $\boldsymbol{\theta}_2$ are the Lagrange multipliers, the former is scalar and the latter a vector. By differentiating (28) with respect to $\mathbf{h}_{\mathrm{Vs}}(n+1)$ and setting the resulting expression equal to zero, one obtains

$$\mathbf{h}_{\mathrm{Vs}}(n+1) = \mathbf{h}_{\mathrm{Vs}}(n) + \frac{1}{2}[\theta_1 \tilde{\mathbf{x}}'_{\mathrm{V}}(n) - \mathbf{C}\boldsymbol{\theta}_2]. \quad (29)$$

By substituting (29) into (27) and considering that $\mathbf{C}^{\mathrm{T}}\mathbf{C} = \mathbf{I}$ (identity matrix), we get

$$\boldsymbol{\theta}_2 = \mathbf{C}^{\mathrm{T}}\left[2\mathbf{h}_{\mathrm{Vs}}(n) + \theta_1 \tilde{\mathbf{x}}'_{\mathrm{V}}(n)\right]. \quad (30)$$

Applying (30) in (29) and substituting the resulting expression into (26), after some mathematical manipulations, we determine

$$\theta_1 = \frac{2e(n)}{\left\|\mathbf{P}\tilde{\mathbf{x}}'_{\mathrm{V}}(n)\right\|^2} \quad (31)$$

where $\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{C}^{\mathrm{T}}$ and $e(n) = d(n) - \hat{y}(n)$ is the error signal. From (29), (30), and (31), and including the control factors α_{Vs} and ψ_{Vs}, the NLMS expression for updating the coefficients of the sparse Volterra filter is

$$\mathbf{h}_{\mathrm{Vs}}(n+1) = \mathbf{P}\mathbf{h}_{\mathrm{Vs}}(n) + \frac{\alpha_{\mathrm{Vs}}}{\left\|\mathbf{P}\tilde{\mathbf{x}}'_{\mathrm{V}}(n)\right\|^2 + \psi_{\mathrm{Vs}}} e(n)\mathbf{P}\tilde{\mathbf{x}}'_{\mathrm{V}}(n). \quad (32)$$

Since $\mathbf{P}$ is a diagonal matrix of ones with its elements zeroed in the diagonal positions corresponding to the elements zeroed in $\mathbf{h}_{\mathrm{Vs}}(n)$, (32) updates only the nonzero coefficients of $\mathbf{h}_{\mathrm{Vs}}(n)$. It is also important to note that the normalization factor $\left\|\mathbf{P}\tilde{\mathbf{x}}'_{\mathrm{V}}(n)\right\|^2$ is obtained considering only part of the elements of $\tilde{\mathbf{x}}'_{\mathrm{V}}(n)$ due to the characteristics of matrix $\mathbf{P}$. Moreover, as discussed for (24), $\tilde{\mathbf{x}}'_{\mathrm{V}}(n)$ from (32) can also be approximated at each iteration by reusing some data, thus reducing its computational burden.

4.3 Computational Complexity

Figure 2 shows the number of operations per sample for implementing different adaptive second-order Volterra filters as a function of the memory size. From this figure, one can note that the proposed LMS/NLMS RBEIV filter presents a computational burden much smaller than the conventional Volterra implementations (both LMS and NLMS Volterra structures), which also is close to the computational burden of the fully adaptive interpolated LMS Volterra (LMS FAIV) implementation [9].

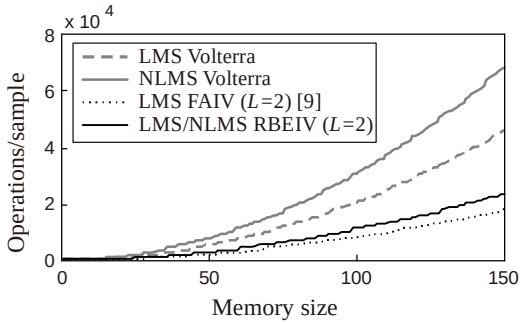

Figure 2 – Computational complexity for different second-order Volterra implementations.

5. SIMULATION RESULTS

In this section, considering a system identification problem [12], comparisons of the fully adaptive LMS/NLMS RBEIV structure with conventional Volterra implementations and also other interpolated Volterra approaches are presented for performance assessment. Such evaluations are accomplished in terms of the MSE obtained from Monte Carlo simulations (average of 100 runs). The simulated structures are second-order implementations of the following adaptive filters: (i) conventional LMS Volterra, (ii) conventional NLMS Volterra, (iii) LMS interpolated Volterra (LMS AIV) [7], (iv) fully adaptive LMS interpolated Volterra (LMS FAIV) [9], and (v) fully adaptive LMS/NLMS RBEIV (proposed here). The sparse Volterra filters from all interpolated implementations present interpolation factor $L = 2$ and the same memory size of the plants to be modeled. The fixed interpolator used by the LMS AIV is given by $\mathbf{g} = [0.5\ 1\ 0.5]^{\mathrm{T}}$, and the adaptive interpolator of the LMS FAIV and LMS/NLMS RBEIV structures is initialized with $\mathbf{g}(0) = [0.5\ 1\ 0.5]^{\mathrm{T}}$. For the filters using the LMS algorithm, the step size is $\mu = \mu_{\max}/2$ ($\mu_{\max}$ is the maximum step-size value for algorithm convergence obtained experimentally), and for the NLMS Volterra filter the control parameters are $\alpha = 0.5$ and $\psi = 10^{-6}$. The parameters used for the LMS/NLMS RBEIV are $\alpha_{\mathrm{Vs}} = 0.5$, $\psi_{\mathrm{Vs}} = 10$, and $\mu_{\mathrm{g}} = \mu_{\mathrm{g}\max}/2$ ($\mu_{\mathrm{g}\max}$ experimentally determined). Moreover, a white Gaussian noise with variance $\sigma_z^2 = 10^{-6}$ is added to the output of the plant.

Example 1: In this example, the plant taken from [10] (Example 1) is a conventional Volterra filter presenting a memory size $N = 11$ and coefficients with decaying exponential values. In Figure 3, the MSE curves obtained from simulations for white Gaussian data with unit variance are shown. In this figure, the convergence rate and steady-state performance of the proposed algorithm in comparison with the other considered algorithms is observed. Furthermore note that in this case, the proposed algorithm presents a performance comparable with the conventional Volterra implementations, demanding smaller computational burden (see Figure 2). To give more insight into the convergence behavior, Figure 4 shows simulation results by using a correlated input signal for the conventional Volterra implementations as well as for the proposed algorithm. Such an input signal is obtained from an AR process given by $x(n) = \beta x(n-1) + \sqrt{1-\beta^2}\,u(n)$, where $u(n)$ is a white Gaussian noise process with unit variance and $\beta = 0.5$. Again very good performance of the proposed algorithm is verified with reduced computational burden.

Example 2: The plant for this example is the conventional Volterra filter with memory size $N = 11$ from [10] (Example 2). The MSE curves obtained by using white Gaussian input data with unit variance are shown in Figure 5. In this figure we again observe a better performance of the proposed algorithm as compared with other adaptive interpolated Volterra implementations. On the other hand, comparing the proposed algorithm with the conventional Volterra ones, we observe better convergence characteristics and worse steady-state performance (the steady-state responses for both conventional Volterra implementations are not completely presented in Figure 5 for scaling reasons). This worse steady-state performance is a direct consequence of the plant characteristics used in this example

[10], presenting both lower correlation between the coefficients and smaller boundary coefficient values than the plant in Example 1. As a rule of thumb, the LMS/NLMS RBEIV exhibits a steady-state response at least equal to that of the LMS FAIV [9], being closer to the conventional Volterra steady state, depending on the correlation level between plant coefficients. In Figure 6, the curves from simulations using a correlated input data obtained in the same way as in Example 1 are presented. Again, we observe satisfactory performance of the proposed reduced-complexity algorithm.

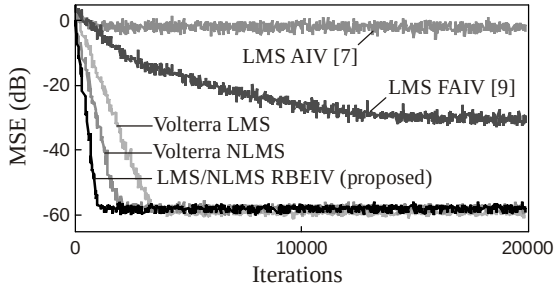

Figure 3 – Example 1. MSE curves for white input data.

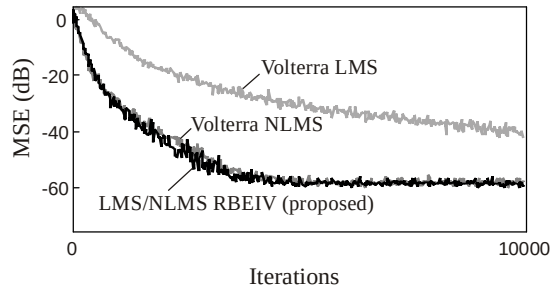

Figure 4 – Example 1. MSE curves for correlated input data.

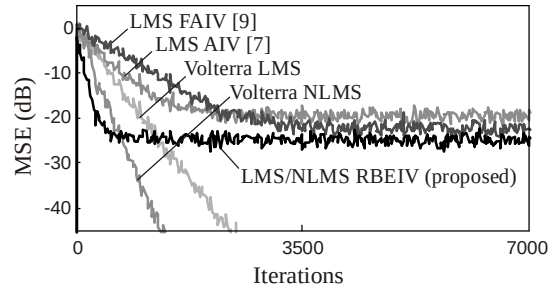

Figure 5 – Example 2. MSE curves for white input data.

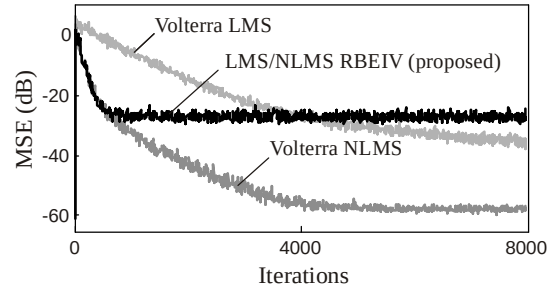

Figure 6 – Example 2. MSE curves for correlated input data.

6. CONCLUSIONS

In this paper, a novel approach for implementing adaptive interpolated Volterra filters is discussed. Such an approach is based on combining the fully adaptation of the interpolated structure, a boundary effect removal procedure, and also an adaptive LMS/NLMS setup. The obtained algorithm outperforms other adaptive interpolated Volterra implementations in terms of transient and steady-state MSE performance. Simulation results attested the effectiveness of the proposed approach.

REFERENCES

[1] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. Signal Process.*, 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," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Phoenix, AZ, Mar. 1999, vol. 2, pp. 877-880.

[3] J. Li and J. Ilow, "Adaptive Volterra predistorters for compensation of non-linear effects with memory in OFDM transmitters," in *Proc. 4th Annual Communication Networks and Services Research Conf.*, Moncton, Canada, May 2006, pp. 1-4.

[4] A. Fermo, A. Carini, and G. L. Sicuranza, "Simplified Volterra filters for acoustic echo cancellation in GSM receivers," in *Proc. Europ. Signal Process. Conf.*, Tampere, Finland, Sep. 2000, pp. 2413-2416.

[5] L. Tan and J. Jiang, "An adaptive technique for modeling second-order Volterra systems with sparse kernels," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 12, pp. 1610-1615, Dec. 1998.

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

[7] E. L. O. Batista, O. J. Tobias, and R. Seara, "A mathematical framework to describe interpolated adaptive Volterra filters," in *Proc. IEEE Int. Telecomm. Symp.*, Fortaleza, Brazil, Sep. 2006, pp. 144-149.

[8] O. J. Tobias and R. Seara, "Analytical model for the first and second moments of an adaptive interpolated FIR filter using the constrained filtered-X LMS algorithm," *IEE Proc.- Vis., Image, Signal Process.*, vol. 148, no. 5, pp. 337-347, Oct. 2001.

[9] E. L. O. Batista, O. J. Tobias, and R. Seara, "A fully LMS adaptive interpolated Volterra structure," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3613-3616.

[10] E. L. O. Batista, O. J. Tobias, and R. Seara, "Border effect removal for IFIR and interpolated Volterra filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Honolulu, HI, Apr. 2007, vol. 3, pp. 1329-1332.

[11] E. L. O. Batista, O. J. Tobias, and R. Seara, "A fully adaptive IFIR filter with removed border effect," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3821-3824.

[12] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley, 2000.

[13] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Sadle River, NJ: Prentice-Hall, 2000.

[14] O. L. Frost III, "An algorithm for linearly constrained adaptive array processing," *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, Aug. 1972.