ADAPTIVE NLMS DIAGONALLY-INTERPOLATED VOLTERRA FILTERS FOR NETWORK ECHO CANCELLATION

Eduardo L. O. Batista *Rui Seara*

Department of Informatics and Statistics Department of Electrical Engineering
LINSE – Circuits and Signal Processing Laboratory
Federal University of Santa Catarina
88040-900 – Florianópolis – SC – Brazil
E-mails: ebatista@inf.ufsc.br, seara@linse.ufsc.br

ABSTRACT

This paper presents a novel approach to implement adaptive Volterra filters with reduced complexity aiming at echo cancellation applications. The proposed approach is based on combining two different strategies in order to reduce the computational burden of adaptive Volterra structures, namely shortening the number of kernel diagonals and using a sparse-interpolated setup. As a result, an efficient diagonally-interpolated Volterra implementation is achieved, offering a controllable trade-off between complexity and performance. Simulation results considering a real-world network echo cancellation problem attest the effectiveness of the proposed approach.

1. INTRODUCTION

A usual problem in many telephone systems is the occurrence of echo signals that, in the case of voice communications, may significantly interfere in the conversation flow. Moreover, echoes have an important impact on the performance of voice over IP (VoIP) [1] and speech recognition systems [2], which have been increasingly used over the last few years. In this context, the use of echo cancellation systems has become essential to guarantee the quality and reliability of telephone communications.

A common approach to perform echo cancellation is the use of linear adaptive filters to model the echo path and generate an echo replica which is subtracted from the received signal [3]. However, such an approach often does not fulfill the performance requirements due to the presence of nonlinear distortions in the echo path [3]. In these cases, the use of nonlinear adaptive filters, such as the Volterra filter, may be necessary to obtain satisfactory performance [3].

Volterra filters have been used successfully in network and acoustic echo cancellation schemes [4]-[6]. In general, these schemes use reduced-complexity Volterra implementations due to the high computational requirements of the standard Volterra filters [4]-[6]. A usual strategy for obtaining such reduced-complexity implementations is to discard from the standard Volterra filter either non-dominant kernels or kernel diagonals [4]-[5]. The drawback of the resulting sparse approaches is that no mechanism is adopted to offset the loss of performance that arises from discarding entire kernels or parts of them. In this framework, sparse-interpolated implementations [7] may be of interest since they use an interpolation scheme to compensate for the loss of performance arising from the strictly-sparse approaches. Thereby, sparse-interpolated structures bear an equivalent non-sparse characteristic despite using sparseness for complexity reduction [7].

In this paper, a novel strategy to efficiently implement adaptive Volterra filters is proposed. The central idea here is to develop a sparse-interpolated scheme which allows the implementation of Volterra structures presenting a reduced number of kernel diagonals, such as the simplified Volterra filters [4] and the power filters [5]. Thus, the computational burden is reduced by shortening the number of diagonals as well as by using both sparseness and interpolation. As a result, a scalable structure termed diagonally-interpolated Volterra filter is achieved. Such an approach leads to a structure of filtering with a changeable trade-off between complexity and performance. Simulation results considering a network echo cancellation problem with signals from a real-world application are presented aiming to verify the effectiveness of the proposed approach.

This paper is organized as follows. Section 2 describes the basics of Volterra filtering, the diagonal coordinate implementation and the sparse-interpolated approach. In Section 3, the diagonally-interpolated Volterra approach is discussed as well as the update of its coefficients by using an adaptive normalized least-mean-square (NLMS) algorithm. Section 4 presents the simulation results. Finally, in Section 5, concluding remarks are presented.

2. BACKGROUND ON STANDARD, DIAGONAL COORDINATE AND SPARSE-INTERPOLATED VOLTERRA FILTERS

In this section, we briefly review the standard Volterra filter as well as its implementations based on exploiting the diagonal coordinate representation and sparse-interpolated structures.

2.1. Standard Volterra Filter

As described in [3], the input-output relationship of the Volterra filter can be written as the sum of the outputs of kernels with distinct orders. Thus,

$$y(n) = \sum_{p=1}^{P} y_p(n) \tag{1}$$

with $y(n)$ denoting the output signal, P, the filter order, and $y_p(n)$, the output of each kernel, which is given by

$$y_p(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} \cdots \sum_{m_p=m_{p-1}}^{N-1} \underline{h}_p(m_1, m_2, \ldots, m_p) \times \prod_{k=1}^{p} x(n-m_k) \tag{2}$$

where $x(n)$ denotes the input signal, N, the memory size, and $\underline{h}_p(m_1, m_2, \ldots, m_p)$, the pth-order filter coefficients. It is important to highlight that (2) corresponds to both the triangular [3] and redundancy-removed [7] Volterra implementations, which are obtained removing the redundant kernel coefficients with no loss of generality. In this work, the focus is on second-order Volterra implementations due to the characteristics of the application used. For this case, the input-output relationship obtained from (1) and (2) is

$$\begin{aligned} y(n) = & \sum_{m_1=0}^{N-1} \underline{h}_1(m_1) x(n-m_1) \\ & + \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} \underline{h}_2(m_1, m_2) x(n-m_1) x(n-m_2). \end{aligned} \tag{3}$$

By defining the first-order coefficient vector as

$$\underline{\mathbf{h}}_1 = [\underline{h}_1(0) \;\; \underline{h}_1(1) \;\; \cdots \;\; \underline{h}_1(N-1)]^{\mathrm{T}} \tag{4}$$

and the first-order input vector by

$$\underline{\mathbf{x}}_1(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}} \tag{5}$$

the first right-hand side (RHS) term of (3) can be rewritten (in a vector form) as

$$y_1(n) = \underline{\mathbf{h}}_1^{\mathrm{T}} \underline{\mathbf{x}}_1(n). \tag{6}$$

Moreover, defining the second-order input vector as

$$\begin{aligned} \underline{\mathbf{x}}_2(n) = [& x^2(n) \;\; x(n)x(n-1) \;\; \cdots \;\; x(n)x(n-N+1) \\ & x^2(n-1) \;\; x(n-1)x(n-2) \;\; \cdots \;\; x^2(n-N+1)]^{\mathrm{T}} \end{aligned} \tag{7}$$

and the second-order coefficient vector by

$$\begin{aligned} \underline{\mathbf{h}}_2 = [& \underline{h}_2(0,0) \;\; \underline{h}_2(0,1) \;\; \cdots \;\; \underline{h}_2(0,N-1) \\ & \underline{h}_2(1,1) \;\; \underline{h}_2(1,2) \;\; \cdots \;\; \underline{h}_2(N-1,N-1)]^{\mathrm{T}} \end{aligned} \tag{8}$$

the second RHS term of (3) becomes

$$y_2(n) = \underline{\mathbf{h}}_2^{\mathrm{T}} \underline{\mathbf{x}}_2(n). \tag{9}$$

The second-order coefficient vector can also be represented in the form of a matrix, considering the indices of each coefficient as row and column coordinates (Cartesian coordinates). Thus, for a case with memory size $N = 3$, we obtain the following coefficient matrix:

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_2(0,0) & \underline{h}_2(0,1) & \underline{h}_2(0,2) \\ 0 & \underline{h}_2(1,1) & \underline{h}_2(1,2) \\ 0 & 0 & \underline{h}_2(2,2) \end{bmatrix}. \tag{10}$$

2.2. Implementations Based on Diagonal Coordinates

Following the same steps as [8], a diagonal coordinate implementation of Volterra filters can be obtained by introducing a change of coordinates in (2). For our case (second-order Volterra filters), the following change of coordinates is introduced in (3): $m_1 = s$ and $m_2 = s + r$. Thus,

$$\begin{aligned} y(n) = & \sum_{s=0}^{N-1} \underline{h}_1(s) x(n-s) \\ & + \sum_{s=0}^{N-1} \sum_{r=0}^{N-1-s} \underline{h}_2(s, s+r)\, x(n-s) x(n-s-r). \end{aligned} \tag{11}$$

Note that the first RHS term of (3) remains unchanged in (11). On the other hand, the modification in the second RHS term allows exchanging the summations to obtain

$$y_2(n) = \sum_{r=0}^{N-1} \sum_{s=0}^{N-1-r} \underline{h}_2(s, s+r)\, x(n-s) x(n-s-r). \tag{12}$$

Now, defining a new partial second-order input vector as

$$\begin{aligned} \mathbf{u}_{2,r}(n) = [& x(n)x(n-r) \;\; x(n-1)x(n-1-r) \;\; \cdots \\ & x(n-N+1+r)x(n-N+1)]^{\mathrm{T}} \end{aligned} \tag{13}$$

and the corresponding coefficient vector by

$$\underline{\mathbf{h}}_{2,r} = [\underline{h}_2(0,r) \;\; \underline{h}_2(1,1+r) \;\; \cdots \;\; \underline{h}_2(N-1-r, N-1)]^{\mathrm{T}} \tag{14}$$

(12) can be rewritten as

$$y_2(n) = \sum_{r=0}^{N-1} \underline{\mathbf{h}}_{2,r}^{\mathrm{T}} \mathbf{u}_{2,r}(n). \tag{15}$$

Note that (13) is composed of delayed versions of $x(n)x(n-r)$, whereas (14) is formed by elements of a diagonal of the coefficient matrix $\underline{\mathbf{H}}_2$ [see (10)]. Thus, each product $\underline{\mathbf{h}}_{2,r}^{\mathrm{T}} \mathbf{u}_{2,r}(n)$, in (15), corresponds to filtering an input signal $x(n)x(n-r)$ through an FIR filter whose coefficients are obtained from one of the diagonals of the coefficient matrix. As a consequence, the diagonal coordinate implementation of a second-order Volterra filter results in a parallel structure of FIR filters, as illustrated in Figure 1.

The main advantage of the diagonal coordinate implementation, as compared with the redundancy-removed, is that some of the branches can be disregarded aiming to obtain a structure with reduced complexity. This feature is especially interesting in applications in which the coefficients of the main diagonals of the kernels are the most important ones [4]-[5]. Examples of reduced-complexity Volterra implementations exploiting this characteristic are the simplified Volterra implementation [4], in which the last branches of each kernel are disregarded, and the power filters [5], in which only the main branch (or diagonal) from each kernel is preserved.

2.3. Sparse-Interpolated Volterra Filters

Another way for implementing Volterra filters with reduced complexity is the sparse-interpolated approach [7].

Such a scheme uses an input interpolator $\mathbf{g}$ cascaded with a sparse Volterra filter $\mathbf{h}_{\mathrm{Vs}}$ having a reduced number of coefficients, which is illustrated in Figure 2. The interpolator is a linear FIR filter with memory size M and coefficient vector given by

$$\mathbf{g} = [g(0) \;\; g(1) \;\; \cdots \;\; g(M-1)]^{\mathrm{T}}. \tag{16}$$

Thus, the signal in the output of the interpolator is

$$\tilde{x}(n) = \mathbf{g}^{\mathrm{T}} \mathbf{x}_{\mathrm{M}}(n) \tag{17}$$

where $\mathbf{x}_{\mathrm{M}}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-M+1)]^{\mathrm{T}}$.

Additionally, in Figure 2, $\hat{y}(n)$ represents the output signal of the sparse-interpolated Volterra filter. The memory size of the interpolator is a function of the sparseness (or interpolation) factor L [7]. Then,

$$M = 1 + 2(L-1) = 2L-1. \tag{18}$$

The first-order coefficient vector of $\mathbf{h}_{\mathrm{Vs}}$ is obtained by setting to zero $L-1$ of each L coefficients in (4) [7]. Thus, we have

$$\underline{\mathbf{h}}_{1\mathrm{s}} = \{\underline{h}_1(0) \;\; 0 \;\cdots\; \underline{h}_1(L) \;\; 0 \;\cdots\; \underline{h}_1[(N_{\mathrm{s}}-1)L] \;\; 0 \;\cdots\; 0\}^{\mathrm{T}} \tag{19}$$

with the corresponding interpolated input vector given by

$$\tilde{\mathbf{x}}_1(n) = [\tilde{x}(n) \;\; \tilde{x}(n-1) \;\; \cdots \;\; \tilde{x}(n-N+1)]^{\mathrm{T}}. \tag{20}$$

The number of nonzero coefficients in (19) is

$$N_{\mathrm{s}} = \lfloor (N-1)/L \rfloor + 1 \tag{21}$$

where $\lfloor \cdot \rfloor$ represents the truncation operation. For higher order kernels, the sparse coefficient vectors are obtained by zeroing the coefficients which present at least one index not multiple of L [7]. For instance, the coefficient vector for a second-order kernel with memory size $N = 3$ is

$$\underline{\mathbf{h}}_{2\mathrm{s}} = [\underline{h}_2(0,0) \;\; 0 \;\; \underline{h}_2(0,2) \;\; 0 \;\; 0 \;\; \underline{h}_2(2,2)]^{\mathrm{T}} \tag{22}$$

while the corresponding interpolated input vector is

$$\begin{aligned} \underline{\tilde{\mathbf{x}}}_2(n) = [&\tilde{x}^2(n) \;\; \tilde{x}(n)\tilde{x}(n-1) \;\; \tilde{x}(n)\tilde{x}(n-2) \\ &\tilde{x}^2(n-1) \;\; \tilde{x}(n-1)\tilde{x}(n-2) \;\; \tilde{x}^2(n-2)]^{\mathrm{T}}. \end{aligned} \tag{23}$$

Similarly to (10), (22) can be represented in a matrix form as

$$\underline{\mathbf{H}}_{2\mathrm{s}} = \begin{bmatrix} \underline{h}_2(0,0) & 0 & \underline{h}_2(0,2) \\ 0 & 0 & 0 \\ 0 & 0 & \underline{h}_2(2,2) \end{bmatrix}. \tag{24}$$

3. DIAGONALLY-INTERPOLATED VOLTERRA FILTERS

As shown in [7], despite the use of sparseness for complexity reduction, sparse-interpolated Volterra filters exhibit an overall equivalent structure that is not sparse. A precondition to obtain such a non-sparse equivalent structure is to locate the interpolator at the input of the sparse-interpolated Volterra structure [7]. In this way, the zeroed coefficients from the main diagonal and upper triangular part of (24) are recreated by an interpolation process that takes place between coefficients in the same diagonal as well as between coefficients from different adjacent diagonals, as described in detail in [7]. On the other hand, if the interpolator is moved to the output (see Figure 3), a sparse equivalent structure is obtained. To show such a characteristic, we consider the case of a second-order sparse-interpolated Volterra filter with memory size $N = 3$ and sparseness factor $L = 2$. By using the diagonal coordinate representation to implement the sparse Volterra filter, the structure presented in Figure 4 is obtained. Due to the sparseness of $\mathbf{h}_{\mathrm{Vs}}$, the branches of such a structure are also sparse. Thus, the coefficient vector for the first branch is given by (19) and the coefficient vectors for each of the remaining branches are [see (14)]

$$\underline{\mathbf{h}}_{2\mathrm{s},r} = [\underline{h}_2(0,r) \;\; 0 \;\; \underline{h}_2(2,2+r) \;\; 0 \;\; \cdots \;\; \underline{h}_2(N-1-r,N-1)]^{\mathrm{T}}. \tag{25}$$

Furthermore, one branch ($\underline{\mathbf{h}}_{2\mathrm{s},1}$, shown by dashed lines in Figure 4) is removed from the structure since all its coefficients present at least one index not multiple of L. Now, moving the interpolator to the left-hand side (LHS) of all summations in the structure of Figure 4, one observes that the interpolation is performed on each branch and, consequently, along the diagonals of the second-order coefficient matrix. As a consequence, no interpolation is carried out between the coefficients from the adjacent branches $\underline{\mathbf{h}}_{2\mathrm{s},0}$ and $\underline{\mathbf{h}}_{2\mathrm{s},2}$ to recreate $\underline{\mathbf{h}}_{2\mathrm{s},1}$. Thus, the diagonal corresponding to $\underline{\mathbf{h}}_{2\mathrm{s},1}$ remains zeroed, resulting in a sparse equivalent structure for the sparse-interpolated Volterra filter with the interpolator at the output.

To obtain an equivalent structure that is not sparse, we need to modify the characteristics of the sparse filter from the sparse-interpolated structure using the interpolator at the output. Thus, instead of zeroing the coefficients with at least one index not multiple of L, the coefficients with only the first index not multiple of L have to be set to zero. Consequently, none of the branches in the structure of Figure 4 is removed, being all of them sparse [coefficient vectors given by (25)]. Moreover, to improve the performance of the interpolation process, one can move the interpolator to the LHS of all summations in the structure of Figure 4, and use different sets of coefficients for each interpolator of each branch. Thus, the interpolation in any branch can be adjusted independently, leading to a more efficient interpolation process. The resulting structure, termed diagonally-interpolated Volterra filter, is shown in Figure 5. It is important to highlight that, despite the input nonlinearities, all branches of the structure in Figure 5 are composed of a linear sparse FIR filter (memory size N and sparseness factor L) cascaded with an interpolator with memory size $M = 2L-1$ [see (18)].

The drawback of the structure of Figure 5, as compared with those of Figures 2, 3, and 4, is a higher computational burden due to its less-sparse nature as well

as the use of multiple interpolators. However, in the structure of Figure 5, a reduction (truncation) on the number of branches can also be easily carried out in a similar way as used to obtain both the simplified Volterra filters [4] and the power filters [5]. Thus, the computational complexity can be considerably reduced in applications in which only the main diagonals of the kernels are considered. In addition, the diagonally-interpolated structure can be scaled up and down by either using or not the sparse-interpolated approach in each of the branches. For instance, we can set the first two branches (linear and main diagonal of the second-order kernel) as standard branches and the remaining ones as sparse-interpolated to enhance the filter for performance, at the expense of a small increase in complexity. Thus, a better trade-off between complexity and performance can be achieved.

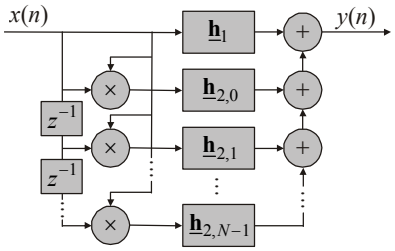

Figure 1 – Block diagram of a diagonal coordinate setup of a second-order Volterra filter.

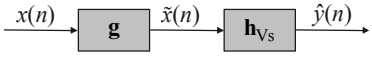

Figure 2 – Block diagram of a sparse-interpolated Volterra filter.

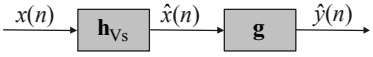

Figure 3 – Block diagram of a sparse-interpolated Volterra filter with the interpolator at the output.

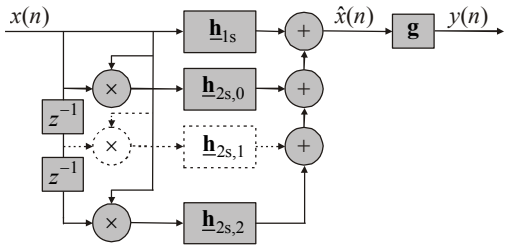

Figure 4 – Block diagram of a second-order sparse-interpolated Volterra filter with the interpolator at the output and $N = 3$.

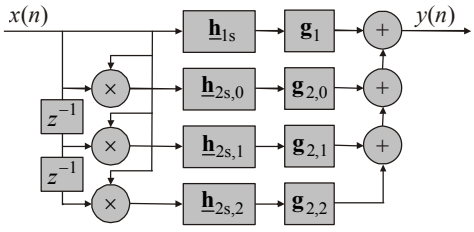

Figure 5 – Block diagram of a second-order diagonally-interpolated Volterra filter with memory size $N = 3$.

3.1 NLMS Coefficient Updating

Since the application considered here involves network echo cancellation with voice signals, the algorithm chosen to update the coefficients of the diagonally-interpolated Volterra implementations is the NLMS. This is due to a relatively small computational burden of such an algorithm as well as its robustness to variations of the input signal power. Note that each branch of the diagonally-interpolated Volterra structure corresponds to either an interpolated FIR (IFIR) filter or the first-order kernel of a sparse-interpolated Volterra filter [7]. Thereby, one can use the algorithm presented in [9] to update the coefficients of the interpolator along with a constrained approach, as that given in [7], to obtain the algorithm for updating the coefficients of the sparse filter. As a result, the update expression for the sparse filter of each second-order branch is

$$\underline{\mathbf{h}}_{2\mathrm{s},r}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{2\mathrm{s},r}(n) + \frac{\alpha_1}{\beta_1 + \psi_1} e(n)\mathbf{P}\tilde{\mathbf{u}}_{2,r}(n) \qquad (26)$$

with

$$e(n) = d(n) - y(n) \qquad (27)$$

where $e(n)$ denotes the error signal, $d(n)$ is the signal with echo, α_1 is the step-size control parameter, ψ_1, a small positive constant (regularization parameter) used for preventing division by zero, and $\mathbf{P}$, a projection matrix due to the sparseness of $\underline{\mathbf{h}}_{2\mathrm{s},r}$ [7]. Moreover, $\tilde{\mathbf{u}}_{2,r}(n)$ is a modified version of (13) obtained by using the signal $x(n)x(n-r)$ filtered by the interpolator $\mathbf{g}_{2,r}$, and β_1, a normalizing term determined by adding the quadratic norms of $\tilde{\mathbf{u}}_{2,r}(n)$ for all considered values of r. On the other hand, to update the interpolator coefficients in each second-order branch, we have

$$\mathbf{g}_{2,r}(n+1) = \mathbf{g}_{2,r}(n) + \frac{\alpha_2}{\beta_2 + \psi_2} e(n)\hat{\mathbf{u}}_{2,r}(n) \qquad (28)$$

where α_2 and ψ_2 are similar to α_1 and ψ_1, respectively, and $\hat{\mathbf{u}}_{2,r}(n)$ is a vector built with M delayed samples obtained from the output of the sparse filter of the rth branch. In addition, β_2 is a normalizing term obtained by adding the quadratic norms of the vectors $\hat{\mathbf{u}}_{2,r}(n)$ from all branches. The update of the coefficients from the first-order branch are carried out similarly to (26) and (28).

4. SIMULATION RESULTS

This section presents numerical simulation results aiming to evaluate the use of the diagonally-interpolated Volterra approach in a network echo cancellation problem with signals obtained from an analog telephone adapter (ATA) used in VoIP systems. Preliminary simulation results, considering some reduced-complexity Volterra implementations, have pointed out a satisfactory performance of the simplified Volterra (SV) filter [4] (with memory size of 51, a first-order branch and 3 second-order branches) as performing echo cancelation with the involved signals. In this way, we assess two different diagonally-interpolated Volterra (DIV) setups, both

presenting memory size of 51: (DIV-A) a DIV filter with a standard first-order branch and 3 sparse-interpolated second-order branches with $L = 3$; and (DIV-B) another DIV filter using two standard branches (first-order and main second-order ones) and 2 additional second-order sparse-interpolated branches using $L = 3$. Furthermore, a linear FIR filter is also used in the simulations. All filters make use of the NLMS algorithm to update the coefficients with step-size control parameters $\alpha_1 = \alpha_2 = 0.5$ and regularization parameters $\psi_1 = \psi_2 = 0.1$. The performance comparison is carried out in terms of echo return loss enhancement (ERLE), defined (in dB) as [4]

$$ERLE = 10\log_{10}\frac{E[d^2(n)]}{E[e^2(n)]} \qquad (29)$$

where $d(n)$ is the signal with echo and $e(n)$, the error signal. The obtained ERLE curves have been smoothed using a moving average filter with memory size $N_{\text{ma}} = 500$. The computational complexity for each of the considered filters is shown in Table 1, in terms of operations per sample and memory use, while the simulation results are presented in Figure 6. From this figure, we observe that the less-complex DIV-A implementation provides a performance close to that from the SV filter (with about 40% less additions, multiplications, and memory use), while the DIV-B implementation presents nearly the same performance as the SV filter (with about 27% less additions and multiplications as well as 26% less memory use). In terms of subjective audio quality, significant differences have not been perceived between the echo-cancelled signals obtained using the SV, DIV-A, and DIV-B implementations. The obtained results illustrate the effectiveness of the proposed approach as well as its flexibility for providing implementations with different trade-offs between complexity and performance.

TABLE 1
COMPLEXITY COMPARISON BETWEEN THE DIFFERENT IMPLEMENTATIONS CONSIDERED IN THE SIMULATIONS

	Additions	Multiplications	Divisions	Memory
FIR	153	154	1	108
SV	600	607	1	412
DIV-A	350	356	2	247
DIV-B	437	443	2	305

5. CONCLUDING REMARKS

This paper presented a novel approach for implementing adaptive Volterra filters with reduced complexity. Such an approach is based on using sparseness and interpolation to obtain computational savings in adaptive Volterra implementations with a reduced number of kernel diagonals. Numerical simulation results for a real-world network echo cancellation problem are shown, corroborating the effectiveness of the proposed approach.

6. ACKNOWLEDGMENT

The authors would like to thank the National Council for Scientific and Technological Development (CNPq) for the financial support of this work.

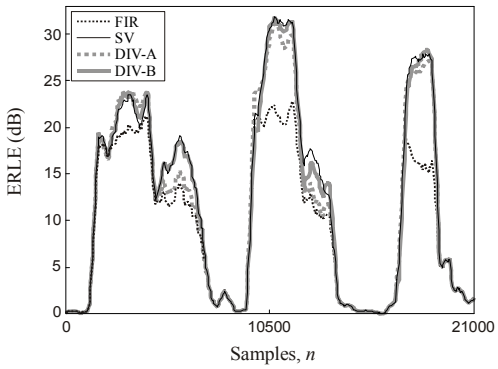

Figure 6. ERLE curves obtained from simulations using real-world network signals of an echo cancellation application. (a) First frame. (b) Second frame.

7. REFERENCES

[1] J. H. James, B. Chen, and L. Garrison, "Implementing VoIP: a voice transmission performance progress report," *IEEE Commun. Mag.*, vol. 42, no. 7, pp. 36-41, July 2004.

[2] X. Huang, A. Acero, and H. W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[3] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: John Wiley & Sons Inc., 2000.

[4] A. Fermo, A. Carini, and G. L. Sicuranza, "Low-complexity nonlinear adaptive filters for acoustic echo cancellation in GSM handset receivers," *Eur. Trans. Telecommun.*, vol. 14, no. 2, pp. 161-169, Mar./Apr. 2003.

[5] F. Kuech, A. Mitnacht, and W. Kellermann, "Nonlinear acoustic echo cancellation using adaptive orthogonalized power filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*. (*ICASSP*), Philadelphia, PA, Mar. 2005, vol. 3, pp. 105-108.

[6] A. Guerin, G. Faucon, and R. Le Bouquin-Jeannes, "Nonlinear acoustic echo cancellation based on Volterra filters," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp.672-684, Nov. 2003.

[7] E. L. O. Batista, O. J. Tobias, and R. Seara, "A sparse-interpolated scheme for implementing adaptive Volterra filters," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2022-2035, Apr. 2010.

[8] G. V. Raz and B. V. Veen, "Baseband Volterra filters for implementing carrier based nonlinearities," *IEEE Trans. Signal Process.*, vol. 46, no. 1, pp.103-114, Jan. 1998.

[9] E. L. O. Batista, O. J. Tobias, and R. Seara, "New insights in adaptive cascaded FIR structure: application to fully adaptive interpolated FIR structures," in *Proc. Eur. Signal Process. Conf.* (*EUSIPCO*), Poznan, Poland, Sep. 2007, pp. 370-374.