A Novel Reduced-Rank Approach for Implementing Volterra Filters

Eduardo Luiz Ortiz Batista and Rui Seara

LINSE – Circuits and Signal Processing Laboratory
Departament of Electrical and Electronics Engineering
Federal University of Santa Catarina
88040-900 - Florianpolis - SC - Brazil
E-mails: ebatista@ieee.org, seara@linse.ufsc.br

***Abstract*—This paper presents a novel reduced-rank approach for implementing Volterra filters with reduced complexity. Such an approach is based on the application of the singular value decomposition to a new form of coefficient matrix obtained by exploiting the representation based on diagonal coordinates of the Volterra kernels. The result is a parallel structure of extended Hammerstein models in which each branch is related to one of the singular values of the coefficient matrix. Then, removing the branches related to the smallest singular values, an effective reduced-complexity Volterra implementation is obtained. Simulation results are presented to confirm the effectiveness of the proposed approach.**

***Index Terms*—Linear-in-the-parameters filters, reduced-rank implementation, Volterra filters.**

I. Introduction

In general, a nonlinear filter is defined simply as a filter for which the superposition principle does not hold [1]. This somewhat broad definition poses a big challenge to researchers and engineers, since it makes the development of a general nonlinear filtering theory a very difficult or almost impossible task. To overcome this problem, the study of nonlinear filters is usually carried out separating such filters into different classes considering some other particular characteristics. In this context, a class of nonlinear filters that has attracted attention over the last decade is that of the linear-in-the-parameters (LIP) filters [1].

A linear dependence between the filter output and the filter coefficients is what characterizes the members of the LIP class. As described in [1], this characteristic is very interesting from the practical point of view, since the least squares methods and adaptive algorithms applied to linear filters can be suitably extended to deal with LIP filters. Some of the members of the LIP class have the additional characteristic of being universal approximators for causal time-invariant finite-memory continuous nonlinear systems, according to the Stone-Weierstrass theorem [1]–[6]. Among these members, the Volterra filter is the precursor and still the most popular [1], [2], [7]–[9].

The aforementioned universality of the Volterra filter comes at the cost of a high number of parameters/coefficients [2], [8]. Moreover, this number of parameters grows exponentially with the memory size, which often leads to a computational cost that forbids the application of the Volterra filter. As a result, a considerable research effort has been devoted to the development of implementations of Volterra filters with reduced computational burden. One interesting approach used for obtaining this type of implementation is based on applying matrix or tensor decompositions to structured representations of Volterra kernels [10]–[15]. In general, this approach results in parallel structures in which each branch has a different level of significance usually related to one of the singular values of a matrix or tensor of coefficients. Then, neglecting the least significant branches (related to the smallest singular values), an effective reduced-rank implementation with reduced computational complexity is obtained.

In this research work, we introduce a novel reduced-rank approach for implementing Volterra filters. The proposed approach is based on the application of the singular value decomposition to a new form of coefficient matrix obtained by exploiting the diagonals of the Volterra kernels. As a result, a parallel structure composed of extended Hammerstein models is obtained and, neglecting the branches of such an structure related to the smallest singular values, an effective reduced-complexity implementation is achieved. Results of a case study involving a Volterra kernel obtained from a real-world echo canceling application are presented to illustrate the applicability of the proposed approach.

The remainder of this paper is organized as follows. In Section II, the Volterra filter, its diagonal-coordinate representation, and its reduced-rank implementations are briefly reviewed. Section III presents the contributions of this paper, including the new strategy to represent the Volterra coefficient matrix as well as the proposed structure for implementing Volterra filters. Sections IV and V present, respectively, simulation results and concluding remarks.

II. Background on Volterra Filters and Reduced-rank Implementations

A truncated Volterra filter is composed of P kernels, each corresponding to a distinct nonlinearity order [2]. The output of such a filter is given by

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

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

where

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

is the output of the pth-order kernel, $x(n)$ is the input signal, and N is the memory size. Moreover, $h_p(m_1, m_2, \ldots, m_p)$ (with $m_1, m_2, \ldots, m_p$ ranging from 0 to N) represent the pth-order coefficients or parameters of the Volterra filter.

Now, defining a first-order coefficient vector as $\mathbf{h}_1 = [h_1(0)\ h_1(1)\ \cdots\ h_1(N-1)]^{\mathrm{T}}$ and a first-order input vector as $\mathbf{x}_1(n) = [x(n)\ x(n-1)\ \cdots\ x(n-N+1)]^{\mathrm{T}}$, the output of the first-order kernel can be written as

$$y_1(n) = \mathbf{h}_1^{\mathrm{T}} \mathbf{x}_1(n). \quad (3)$$

Note that (3) is equivalent to the input-output relationship of an FIR (linear) filter and, thus, the first-order kernel is in fact a linear kernel. The remaining kernels (those with $p \geq 2$) are the nonlinear kernels. In general, the strategies for reducing the complexity of Volterra filters are applied to the nonlinear kernels, since they contain most of the filter coefficients. Thus, for the sake of simplicity, this paper is focused primarily on the implementation of the second-order kernel.

A. Second-order Kernel

The input-output relationship of the second-order kernel is obtained by using $p = 2$ in (2), resulting in

$$y_2(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} h_2(m_1, m_2) x(n-m_1) x(n-m_2). \quad (4)$$

Now, considering the indices m_1 and m_2 of the second-order coefficients $h_2(m_1, m_2)$ as coordinates of a Cartesian space, the following coefficient matrix is defined:

$$\mathbf{H}_2 = \begin{bmatrix} h_2(0,0) & h_2(0,1) & \cdots & h_2(0,N-1) \\ h_2(1,0) & h_2(1,1) & \cdots & h_2(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ h_2(N-1,0) & h_2(N-1,1) & \cdots & h_2(N-1,N-1) \end{bmatrix}. \quad (5)$$

Thus, the output of the second-order kernel can be written as

$$y_2(n) = \mathbf{x}_1^{\mathrm{T}}(n) \mathbf{H}_2 \mathbf{x}_1(n). \quad (6)$$

B. Redundancy-removed Implementation

A more interesting approach for implementing nonlinear kernels is based on the so-called redundancy-removed (also known as triangular) representation [2], [8]. This representation of Volterra kernels is obtained by considering that the coefficients with permutated indices are in fact related to the same cross-product of the input signal [e.g., both $h_2(0,1)$ and $h_2(1,0)$ are multiplied by $x(n)x(n-1)$ for evaluating $y_2(n)$]. Thus, such coefficients can be merged into a single coefficient (as described in [16]) with no loss of generality. As a result, the output of the second-order kernel becomes

$$y_2(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} \underline{h}_2(m_1, m_2) x(n-m_1) x(n-m_2) \quad (7)$$

with $\underline{h}_2(m_1, m_2)$ denoting the coefficients of the redundancy-removed representation. Moreover, defining a redundancy-removed second-order coefficient matrix as

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_2(0,0) & \underline{h}_2(0,1) & \cdots & \underline{h}_2(0,N-1) \\ 0 & \underline{h}_2(1,1) & \cdots & \underline{h}_2(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \underline{h}_2(N-1,N-1) \end{bmatrix} \quad (8)$$

(7) can be rewritten as $y_2(n) = \mathbf{x}_1^{\mathrm{T}}(n) \underline{\mathbf{H}}_2 \mathbf{x}_1(n)$.

C. Implementation Based on Diagonal Coordinates

Another interesting strategy for implementing Volterra filters is based on representing nonlinear kernels using diagonal coordinates [2], [16]. For the case of the second-order kernel, such representation is obtained by introducing the following change of coordinates in (7): $m_1 = s$ and $m_2 = s + r$. In doing so and after exchanging the order of the summations in the resulting expression, one obtains

$$y_2(n) = \sum_{r=0}^{N-1} \sum_{s=0}^{N-1-r} \underline{h}_2(s, s+r) x(n-s) x(n-s-r). \quad (9)$$

Now, defining a partial second-order coefficient vector as

$$\grave{\mathbf{h}}_{2,r} = [\underline{h}_2(0,r)\ \underline{h}_2(1,r-1)\ \cdots\ \underline{h}_2(N-1-r, N-1)]^{\mathrm{T}} \quad (10)$$

and a corresponding input vector as

$$\grave{\mathbf{x}}_{2,r}(n) = [x(n)x(n-r)\ x(n-1)x(n-1-r) \cdots\ x(n-N+1+r)x(n-N+1)]^{\mathrm{T}} \quad (11)$$

(9) can be rewritten as

$$y_2(n) = \sum_{r=0}^{N-1} \grave{\mathbf{h}}_{2,r}^{\mathrm{T}} \grave{\mathbf{x}}_{2,r}(n). \quad (12)$$

As pointed out in [17], (10) is formed by the elements of the rth diagonal of (8), while (11) is composed of delayed versions of $x(n)x(n-r)$. Consequently, each product $\grave{\mathbf{h}}_{2,r}^{\mathrm{T}} \grave{\mathbf{x}}_{2,r}(n)$ in (12) corresponds to the filtering of the signal $x(n)x(n-r)$ by an FIR filter whose coefficients are the elements of the rth diagonal of (8). Thus, the diagonal-coordinate representation leads to the parallel implementation illustrated in Fig. 1.

In principle, the representation based on diagonal coordinates of a Volterra kernel has the same number of parameters as the redundancy-removed representation. However, in many real-world applications, the coefficients close to the main kernel diagonal are the most significant ones [17]–[20]. Thus, effective reduced-complexity Volterra implementations can be obtained simply by neglecting the coefficients far from the main kernel diagonal, which implies removing the blocks in the lower positions of the parallel structure of Fig. 1.

D. Reduced-rank Implementations

As described in [10]–[15], effective structures for implementing Volterra filters with reduced complexity can also be obtained by exploiting matrix or tensor decompositions. Aiming to illustrate the basic characteristics of these structures, we consider here the approach for implementing second-order kernels described in [12]. Such an approach is based on the decomposition of the symmetric matrix $\mathbf{H}_2$ [see (5)] by using the singular value decomposition, which results in

$$\mathbf{H}_2 = \sum_{k=0}^{N-1} \lambda_k \tilde{\mathbf{h}}_{2,k} \tilde{\mathbf{h}}_{2,k}^{\mathrm{T}} \tag{13}$$

where λ_k and $\tilde{\mathbf{h}}_{2,k}$ are, respectively, the kth singular value and the kth singular vector of $\mathbf{H}_2$. Then, substituting (13) into (6) and manipulating the resulting expression, one obtains

$$y_2(n) = \sum_{k=0}^{N-1} \lambda_k [\mathbf{x}_1^{\mathrm{T}}(n) \tilde{\mathbf{h}}_{2,k}]^2. \tag{14}$$

Since $\mathbf{x}_1^{\mathrm{T}}(n)\tilde{\mathbf{h}}_{2,k}$ corresponds to the output of an FIR filter whose coefficients are the elements of $\tilde{\mathbf{h}}_{2,k}$, one notices that (14) corresponds to the parallel structure shown in Fig. 2. Each branch of this parallel structure is composed of an FIR filter with its squared output multiplied by the kth singular value of $\mathbf{H}_2$. Then, removing the branches comprising the smallest singular values, an effective reduced-complexity implementation is obtained. In general terms, the approaches from [10]–[15] lead to parallel structures in which this kind of singular-value-dependent pruning can be carried out to obtain reduced-complexity implementations.

III. Proposed Approach

In this section, a novel approach for implementing Volterra filters with reduced-rank (and thus reduced computational complexity) is discussed. Such an approach is based on defining a new form to represent the coefficient matrix in a way that the dominance of the main kernel diagonal can also be exploited when performing the singular value decomposition. Thus, for the second-order kernel, the idea is to define the coefficient matrix considering the partial second-order coefficient vectors $\grave{\mathbf{h}}_{2,0}, \grave{\mathbf{h}}_{2,1}, \ldots, \grave{\mathbf{h}}_{2,N-1}$ [see (10)] arising from the representation based on diagonal coordinates. This is, however, not a straightforward task, since such vectors do not have the same length (the length of $\grave{\mathbf{h}}_{2,r}$ is in fact $N-r$). To circumvent this problem, the strategy used here is to extend $\grave{\mathbf{h}}_{2,0}, \grave{\mathbf{h}}_{2,1}, \ldots, \grave{\mathbf{h}}_{2,N-1}$ up to length N. As a result, we define the following extended version of $\grave{\mathbf{h}}_{2,r}$:

$$\bar{\mathbf{h}}_{2,r} = [\underline{h}_2(0,r)\ \underline{h}_2(1,r-1)\ \cdots\ \underline{h}_2(N-1,N-1+r)]^{\mathrm{T}}. \tag{15}$$

Now, considering (15), a diagonal-coordinate coefficient matrix can be defined as

$$\bar{\mathbf{H}}_2 = [\bar{\mathbf{h}}_{2,0}\ \bar{\mathbf{h}}_{2,1}\ \cdots\ \bar{\mathbf{h}}_{2,N-1}]. \tag{16}$$

Similarly, extending $\grave{\mathbf{x}}_{2,r}(n)$ up to length N, we get

$$\begin{aligned}\bar{\mathbf{x}}_{2,r}(n) = [&x(n)x(n-r)\ x(n-1)x(n-1-r)\\ &\cdots\ x(n-N+1)x(n-N+1-r)]^{\mathrm{T}}\end{aligned} \tag{17}$$

which allows defining the following diagonal-coordinate input matrix:

$$\bar{\mathbf{X}}_2(n) = [\bar{\mathbf{x}}_{2,0}(n)\ \bar{\mathbf{x}}_{2,1}(n)\ \cdots\ \bar{\mathbf{x}}_{2,N-1}(n)]. \tag{18}$$

Then, considering (16), (18), and the properties of the vectorization operator described in [21], the output of the second-order kernel is written as

$$y_2(n) = \mathrm{vec}[\bar{\mathbf{X}}_2(n)]^{\mathrm{T}} \mathrm{vec}(\bar{\mathbf{H}}_2) = \mathrm{tr}[\bar{\mathbf{X}}_2^{\mathrm{T}}(n)\bar{\mathbf{H}}_2] \tag{19}$$

with $\mathrm{vec}(\cdot)$ representing the matrix vectorization [21] and $\mathrm{tr}(\cdot)$ the matrix trace.

Now, aiming to obtain the proposed reduced-rank implementation, the idea is to apply a singular value decomposition to $\bar{\mathbf{H}}_2$. Thus, one has

$$\bar{\mathbf{H}}_2 = \mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}} \tag{20}$$

where Λ is a diagonal matrix composed of the singular values of $\bar{\mathbf{H}}_2$, $\mathbf{U}$ is a unitary matrix whose columns are the singular vectors of $\bar{\mathbf{H}}_2\bar{\mathbf{H}}_2^{\mathrm{T}}$, and $\mathbf{V}$ is another unitary matrix whose columns are the singular vectors of $\bar{\mathbf{H}}_2^{\mathrm{T}}\bar{\mathbf{H}}_2$. By substituting (20) into (19) and considering the cyclic property of the trace operator [22], we obtain

$$y_2(n) = \mathrm{tr}[\bar{\mathbf{X}}_2^{\mathrm{T}}(n)\mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}}] = \mathrm{tr}[\mathbf{V}^{\mathrm{T}}\bar{\mathbf{X}}_2^{\mathrm{T}}(n)\mathbf{U}\Lambda] \tag{21}$$

which, considering the properties of the vectorization operator [21], is rewritten as

$$y_2(n) = \mathrm{vec}[\bar{\mathbf{X}}_2(n)\mathbf{V}]^{\mathrm{T}} \mathrm{vec}[\mathbf{U}\Lambda]. \tag{22}$$

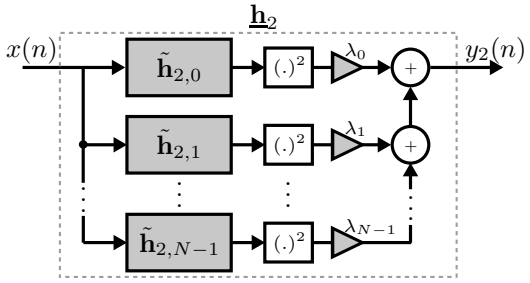

Fig. 2. Block diagram of the implementation of a second-order kernel based on the singular value decomposition.

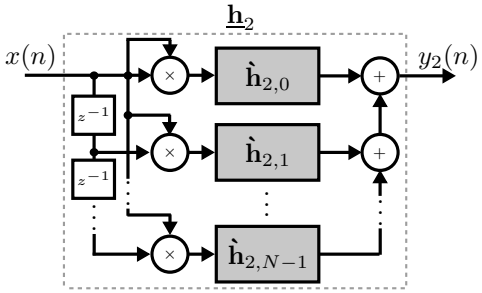

Fig. 1. Block diagram of the implementation based on diagonal coordinates of a second-order Volterra kernel.

The first vectorization operation in the right-hand side (RHS) of (22) can be written as

$$\mathrm{vec}[\bar{\mathbf{X}}_2(n)\mathbf{V}] = \begin{bmatrix} \bar{\mathbf{X}}_2(n)\mathbf{v}_0 \\ \bar{\mathbf{X}}_2(n)\mathbf{v}_1 \\ \vdots \\ \bar{\mathbf{X}}_2(n)\mathbf{v}_{N-1} \end{bmatrix} \quad (23)$$

with $\mathbf{v}_r$ representing the rth column of $\mathbf{V}$. Moreover, since Λ is a diagonal matrix composed of the singular values $\bar{\lambda}_0$ to $\bar{\lambda}_{N-1}$ of $\bar{\mathbf{H}}_2$, the second vectorization operation in the RHS of (22) results in

$$\begin{aligned} \mathrm{vec}[\mathbf{U}\Lambda] &= [\bar{\lambda}_0\mathbf{u}_0^{\mathrm{T}}\ \bar{\lambda}_1\mathbf{u}_1^{\mathrm{T}}\ \cdots\ \bar{\lambda}_{N-1}\mathbf{u}_{N-1}^{\mathrm{T}}]^{\mathrm{T}} \\ &= [\hat{\mathbf{h}}_{2,0}^{\mathrm{T}}\ \hat{\mathbf{h}}_{2,1}^{\mathrm{T}}\ \cdots\ \hat{\mathbf{h}}_{2,N-1}^{\mathrm{T}}]^{\mathrm{T}} \end{aligned} \quad (24)$$

with $\hat{\mathbf{h}}_{2,r} = \bar{\lambda}_r\mathbf{u}_r$ and $\mathbf{u}_r$ representing the rth column of $\mathbf{U}$. Then, considering (23) and (24), (22) is rewritten as

$$y_2(n) = \sum_{k=0}^{N-1} \mathbf{v}_k^{\mathrm{T}}\bar{\mathbf{X}}_2^{\mathrm{T}}(n)\hat{\mathbf{h}}_{2,k}. \quad (25)$$

Now, after analyzing in detail the structure of matrix $\bar{\mathbf{X}}_2(n)$ shown at the bottom of this page, one notices that all rows of such a matrix are delayed versions of the first line. Then, defining a vector

$$\dot{\mathbf{x}}_2(n) = [x^2(n)\ x(n)x(n-1)\ \cdots\ x(n)x(n-N+1)]^{\mathrm{T}} \quad (26)$$

the transpose of $\bar{\mathbf{X}}_2(n)$ can be written as $\bar{\mathbf{X}}_2^{\mathrm{T}}(n) = [\dot{\mathbf{x}}_2(n)\ \dot{\mathbf{x}}_2(n-1)\ \cdots\ \dot{\mathbf{x}}_2(n-N+1)]$, resulting in

$$\begin{aligned} \hat{\mathbf{x}}_{2,k}^{\mathrm{T}}(n) = \mathbf{v}_k^{\mathrm{T}}\bar{\mathbf{X}}_2^{\mathrm{T}}(n) = [\mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n)\ \mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n-1) \\ \cdots\ \mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n-N+1)]. \end{aligned} \quad (27)$$

Finally, substituting (27) into (25), the output of the second-order kernel is rewritten as

$$y_2(n) = \sum_{k=0}^{N-1} \hat{\mathbf{x}}_{2,k}^{\mathrm{T}}(n)\hat{\mathbf{h}}_{2,k}. \quad (28)$$

Note that since $\hat{\mathbf{x}}_{2,k}^{\mathrm{T}}(n)$ is composed of delayed elements of $\mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n)$, we can verify that the kth term of the summation in (28) is in fact the filtering of a signal $\mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n)$ using an FIR filter with coefficient vector $\hat{\mathbf{h}}_{2,k}$. Moreover, since

$$\begin{aligned} f_k(n) = \mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n) = v_{k,0}x^2(n) + v_{k,1}x(n)x(n-1) + \\ \cdots + v_{k,N-1}x(n)x(n-N+1) \end{aligned} \quad (29)$$

one notices that $\mathbf{v}_k^{\mathrm{T}}\dot{\mathbf{x}}_2(n)$ corresponds to the output of a polynomial nonlinearity with coefficients given by the elements $v_{k,0}, v_{k,1}, \ldots, v_{k,N-1}$ of $\mathbf{v}_k$ and variables given by $x(n), x(n)x(n-1), \ldots, x(n)x(n-N+1)$. Thus, we conclude that (28) corresponds to the filtering of $x(n)$ by a structure composed of N parallel *extended* Hammerstein models, as shown in Fig. 3. We use here the *extended* qualifier for these Hammerstein models due to the fact that typical Hammerstein models comprise memoryless nonlinearities, whereas the nonlinearities of the models illustrated in Fig. 3 indeed have memory.

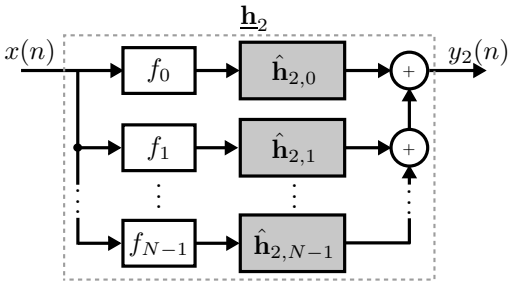

Fig. 3. Proposed parallel Hammerstein implementation of a second-order Volterra kernel.

The proposed structure has some interesting characteristics that are worth being mentioned, namely:

1) *Representation capability*. Due to the extension of coefficient vectors that has resulted in (15), the proposed structure is in fact capable of representing not only a Volterra kernel with memory N, but also parts of the kernel with memory size $2N-1$. This is particularly interesting from the application point of view, since the impact of choosing a reduced memory size is smaller.
2) *Computational complexity*. Considering the second-order kernel, the number of coefficients in each branch of the proposed structure is the same as that of the parallel-cascade structure from [14]. This number is higher than those of the structure from Fig. 2 and of the PARAFAC-Volterra implementation from [15]. However, this higher number of coefficients is compensated by the extended representation capability of the proposed structure.
3) *Diagonal truncation*. The cost for implementing the proposed structure can also be reduced by associating a truncation of the number of diagonals with the aforementioned branch pruning. In this case, the number of columns of $\bar{\mathbf{H}}_2$ and $\bar{\mathbf{X}}_2$ will be reduced, resulting in smaller number of terms in $f_0(n), \ldots, f_{N-1}(n)$.
4) *Extension to higher-order kernels*. The proposed approach can also be used for obtaining implementations of higher-order kernels either by redefining the coefficient and input matrices given by (16) and (18) or by developing a strategy based on tensor decomposition.

$$\bar{\mathbf{X}}_2(n) = \begin{bmatrix} x^2(n) & x(n)x(n-1) & \cdots & x(n)x(n-N+1) \\ x^2(n-1) & x(n-1)x(n-2) & \cdots & x(n-1)x(n-N) \\ \vdots & \vdots & \ddots & \vdots \\ x^2(n-N+1) & x(n-N+1)x(n-N) & \cdots & x(n-N+1)x(n-2N+2) \end{bmatrix}$$

IV. Simulation Results

In this section the effectiveness of the proposed approach for obtaining reduced-complexity implementations of Volterra filters is assessed. To this end, we consider a case study in which different reduced-rank approaches are applied to the implementation of a second-order Volterra kernel. Besides the proposed approach, the other reduced-rank approaches considered here are the one based on the singular value decomposition (SVD-based) from [12] and the parallel-cascade (PC) approach from [14] (without exploiting kernel symmetry). The approach based on the PARAFAC decomposition from [15] is not considered, since for second-order kernels, it leads to a structure similar to that of the aforementioned SVD-based approach. The kernel to be implemented corresponds to the second-order part of an echo path model obtained from a real-world network echo cancellation problem, involving an analog telephone adapter (ATA) used in VoIP systems. This kernel has memory size 25, 325 coefficients, and the cost of its standard implementation is of 674 (multiplication and addition) operations per sample. The effectiveness of the reduced-rank approaches is measured in terms of normalized misalignment, which is defined as $10\log_{10}(||\mathbf{H}_{2k} - \mathbf{H}_{2rr}||_F^2/||\mathbf{H}_{2k}||_F^2)$ with $||\cdot||_F$ representing the Frobenius norm, $\mathbf{H}_{2k}$, the coefficient matrix of the kernel to be implemented, and $\mathbf{H}_{2rr}$, the coefficient matrix obtained by using the reduced-rank approach. For all approaches, the number of branches is varied from 0 to 25 (i.e., from the minimum to the maximum) and, to make a fair comparison, the required number of operations per sample is considered in lieu of the number of branches. Fig. 4 shows the obtained curves of normalized misalignment as a function of number of operations per sample for each reduced-rank approach considered. A dotted line is also included in this figure to indicate the complexity of the standard Volterra (SV) implementation of the kernel considered. This line corresponds to a limit after which it is no longer interesting the use of reduced-rank implementations since they cost more than the standard implementation. From Fig. 4, one notices that, for any level of computational burden below that of the SV implementation, a smaller value of normalized misalignment is obtained by the proposed approach as compared with the other approaches considered.

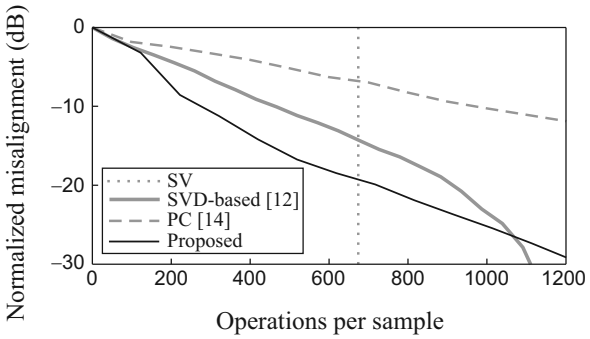

Fig. 4. Computational complexity required by different reduced-rank implementations of a second-order Volterra kernel.

V. Conclusions

In this paper, a novel reduced-rank approach for implementing Volterra filters was presented. This approach is based on first defining a new form of diagonally-oriented coefficient matrix and, then, applying the singular value decomposition to such a matrix. As a result, the Volterra input-output relationship can be rewritten, leading to a parallel implementation composed of extended Hammerstein models. The applicability of the proposed approach was assessed by means of a case study.

References

[1] G. L. Sicuranza and A. Carini, "On a class of nonlinear filters," in *Fest. Honor Jaakko Astola Occas. his 60th Birthd.*, 2009, pp. 115–144.

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

[3] A. Carini and G. L. Sicuranza, "Even mirror Fourier nonlinear filters," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, Vancouver, Canada, May 2013, pp. 5608–5612.

[4] ——, "Fourier nonlinear filters," *Signal Processing*, vol. 94, pp. 183–194, Jan. 2014.

[5] A. Carini, S. Cecchi, L. Romoli, and G. L. Sicuranza, "Legendre nonlinear filters," *Signal Processing*, vol. 109, pp. 84–94, Apr. 2015.

[6] A. Carini and G. L. Sicuranza, "A study about Chebyshev nonlinear filters," *Signal Processing*, vol. 122, pp. 24–32, May 2016.

[7] L. A. Azpicueta-Ruiz, M. Zeller, A. R. Figueiras-Vidal, J. Arenas-Garcia, and W. Kellermann, "Adaptive combination of Volterra kernels and its application to nonlinear acoustic echo cancellation," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 19, no. 1, pp. 97–110, Jan. 2011.

[8] E. L. O. Batista and R. Seara, "On the performance of adaptive pruned Volterra filters," *Signal Processing*, vol. 93, no. 7, pp. 1909–1920, Jul. 2013.

[9] ——, "Advances on adaptive sparse-interpolated filtering," in *Signals and Images: Adv. and Results in Speech, Estim. Compression, Recognition, Filtering, Process.* CRC Press, 2015, ch. 14, pp. 387–420.

[10] H.-H. Chiang, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient implementations of quadratic digital filters," *Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 6, pp. 1511–1528, Dec. 1986.

[11] Y. Lou, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters," *Circuits, Syst. Signal Process.*, vol. 7, no. 2, Jun. 1988.

[12] S. Marsi and G. L. Sicuranza, "On reduced-complexity approximations of quadratic filters," in *Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput.*, vol. 2, Pacific Grove, CA, Nov. 1993, pp. 1026–1030.

[13] R. D. Nowak and B. D. Van Veen, "Tensor product basis approximations for Volterra filters," *IEEE Trans. Signal Process.*, vol. 44, no. 1, pp. 36–50, Jan. 1996.

[14] T. M. Panicker, V. J. Mathews, and G. L. Sicuranza, "Adaptive parallel-cascade truncated Volterra filters," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2664–2673, 1998.

[15] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," *Int. J. Adapt. Control Signal Process.*, vol. 26, no. 1, pp. 30–53, Jan. 2012.

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

[17] E. L. O. Batista and R. Seara, "Adaptive NLMS diagonally-interpolated Volterra filters for network echo cancellation," in *Proc. 19th Eur. Signal Process. Conf.*, Barcelona, Spain, Sep. 2011, pp. 1430–1434.

[18] A. Fermo, A. Carini, and G. L. Sicuranza, "Simplified Volterra filters for acoustic echo cancellation in GSM receivers," in *Proc. 10th Eur. Signal Process. Conf.*, Tampere, Finland, Sep. 2000, pp. 1–4.

[19] ——, "Low-complexity nonlinear adaptive filters for acoustic echo cancellation in GSM handset receivers," *Eur. Trans. Telecommun.*, vol. 14, no. 2, pp. 161–169, Mar. 2003.

[20] F. Kuech and W. Kellermann, "Orthogonalized power filters for nonlinear acoustic echo cancellation," *Signal Processing*, vol. 86, no. 6, pp. 1168–1181, Jun. 2006.

[21] J. W. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Trans. Circuits Syst.*, vol. 25, no. 9, pp. 772–781, Sep. 1978.

[22] D. S. Bernstein, *Matrix Mathematics*, 2nd ed. Princeton, NJ: Princeton University Press, 2009.