Batista and Seara *EURASIP Journal on Advances in Signal Processing* (2016) 2016:118
DOI 10.1186/s13634-016-0420-5

EURASIP Journal on Advances in Signal Processing

RESEARCH

Open Access

A reduced-rank approach for implementing higher-order Volterra filters

Eduardo L. O. Batista* and Rui Seara

Abstract

The use of Volterra filters in practical applications is often limited by their high computational burden. To cope with this problem, many strategies for implementing Volterra filters with reduced complexity have been proposed in the open literature. Some of these strategies are based on reduced-rank approaches obtained by defining a matrix of filter coefficients and applying the singular value decomposition to such a matrix. Then, discarding the smaller singular values, effective reduced-complexity Volterra implementations can be obtained. The application of this type of approach to higher-order Volterra filters (considering orders greater than 2) is however not straightforward, which is especially due to some difficulties encountered in the definition of higher-order coefficient matrices. In this context, the present paper is devoted to the development of a novel reduced-rank approach for implementing higher-order Volterra filters. Such an approach is based on a new form of Volterra kernel implementation that allows decomposing higher-order kernels into structures composed only of second-order kernels. Then, applying the singular value decomposition to the coefficient matrices of these second-order kernels, effective implementations for higher-order Volterra filters can be obtained. Simulation results are presented aiming to assess the effectiveness of the proposed approach.

Keywords: Nonlinear filtering, Reduced-rank implementation, Volterra filter

1 Introduction

The first challenge in filtering applications involving nonlinear systems is to choose an adequate model of the nonlinear filter [1]. To meet this challenge, one important filter characteristic that needs to be considered is the trade-off between implementation complexity and approximation capability. The well-known Volterra filter [1] represents one extreme of this trade-off, since its universal approximation capability [2–4] comes at the cost of a high computational complexity (which is due to the large number of coefficients required for the implementation) [1, 5–9]. In this context, one topic that has drawn attention from researchers in the last decades is the development of Volterra implementations having an enhanced trade-off between computational complexity and approximation capability.

Several different approaches have been proposed in the open literature aiming to obtain reduced-complexity implementations of Volterra filters. Some of these approaches are based on sparse Volterra implementations that are obtained by zeroing the less significant filter coefficients [9]. Examples of these implementations are the Volterra delay filter [10], the power filter [11], and general diagonally-pruned implementations [12–14]. Other approaches combine interpolation and sparse implementations for the sake of performance [15–17]. Frequency-domain approaches also have been used for obtaining effective reduced-complexity Volterra implementations [18, 19]. All aforementioned approaches are, in some sense, based on the use of predefined forms of basis vectors for identifying and discarding the less significant coefficients of a Volterra filter. In contrast, the approaches from [20–25] involve the identification of particular basis vectors that can then be exploited aiming to reduce the complexity of a Volterra filter. These approaches are typically based on the definition of coefficient matrices, which are decomposed aiming to obtain the basis vectors. The singular value decomposition is often used for carrying out such a matrix decomposition

*Correspondence: ebatista@ieee.org
LINSE—Circuits and Signal Processing Laboratory, Department of Electrical and Electronics Engineering, Federal University of Santa Catarina, Campus Universitário Trindade, Florianópolis, Brazil

and, as a result, the obtained basis vectors are singular vectors of the coefficient matrix considered. Thus, discarding the singular vectors related to the smaller singular values, effective reduced-complexity reduced-rank Volterra implementations are obtained.

The first reduced-rank approaches used for implementing Volterra filters are focused on second-order Volterra kernels [20–22]. This is due to the fact that the second-order Volterra coefficients have two indices, which makes the definition of a second-order coefficient matrix a straightforward task. For higher-order filters, matrix-based reduced-rank approaches are usually obtained by considering non-trivial definitions of (often rectangular) coefficient matrices [24], which occasionally lead to ineffective reduced-rank implementations. In this context, the present paper is focused on the development of a novel reduced-rank approach for implementing higher-order Volterra filters. This approach is based on a new form of Volterra kernel implementation that allows converting a higher-order Volterra kernel into a structure composed of second-order kernels. Then, applying second-order reduced-rank implementation strategies to such a structure, effective reduced-rank implementations for higher-order Volterra filters can be achieved.

The remainder of this paper is organized as described in the following. Section 2 revisits the Volterra filters, discussing the redundancy-removed and matrix-form representations of kernel input-output relationships. Also, in Section 2, the reduced-rank implementations of Volterra filters are briefly described. Section 3 is dedicated to the contributions of this paper, comprising a new form of kernel implementation and the description of the proposed approach for implementing reduced-rank Volterra filters. Finally, Sections 4 and 5 are dedicated to present the experimental results and the concluding remarks, respectively.

The mathematical notation considered in this paper is based on the standard practice of using lowercase boldface letters for vectors, uppercase boldface letters for matrices, and both italic Roman and Greek letters for scalar quantities. Moreover, superscript T stands for transpose, $\otimes$ represents the Kronecker product, and $||\cdot||_2$ denotes a quadratic norm. Additionally, underbars specify variables related to the redundancy-removed Volterra representation and overbars indicate variables related to the proposed approach.

2 Volterra filters and reduced-rank implementations

A truncated Pth-order Volterra filter is composed of P kernels, each corresponding to a certain order of polynomial nonlinearity [1]. The output $y(n)$ of such a filter is obtained from

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

with $y_p(n)$ representing the output of the pth-order kernel. In its standard form, the input-output relationship of the pth-order kernel is given by

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

with $x(n)$ denoting the input signal and $h_{p(m_1,m_2,\ldots,m_p)}$, the pth-order coefficients. In this work, the standard kernels are assumed to be symmetric, which means that $h_{p(m_1,m_2,\ldots,m_p)} = h_{p(m_2,m_1,\ldots,m_p)} = \cdots = h_{p(m_p,m_{p-1},\ldots,m_1)}$ (i.e., all pth-order coefficients with permutated indices have identical values). This assumption is used without loss of generality, since any standard Volterra kernel can be represented in symmetric form [1, 25].

The first-order kernel of the Volterra filter is a linear kernel whose input-output relationship

$$y_1(n) = \sum_{m_1=0}^{N-1} h_{p(m_1)} x(n-m_1) \tag{3}$$

is that of a standard finite impulse response (FIR) filter with N coefficients. Thus, (3) can be rewritten in a vector form as

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

with

$$\mathbf{h}_1 = [\, h_{1(0)} \quad h_{1(1)} \quad \cdots \quad h_{1(N-1)}]^{\mathrm{T}} \tag{5}$$

and

$$\mathbf{x}_1(n) = [\, x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}}. \tag{6}$$

The other kernels (with $p \geq 2$) are nonlinear kernels whose outputs depend on cross products of samples of the input signal. As described in [1], the input-output relationships of the nonlinear kernels can also be expressed in vector form. Thus, for the pth-order kernel, one has

$$y_p(n) = \mathbf{h}_p^{\mathrm{T}} \mathbf{x}_p(n), \tag{7}$$

where $\mathbf{h}_p$ is the pth-order coefficient vector (composed of coefficients $h_{p(m_1,m_2,\ldots,m_p)}$ with $m_1,\ldots,m_p$ ranging from 0 to $N-1$) and $\mathbf{x}_p(n) = \mathbf{x}_1(n) \otimes \mathbf{x}_{p-1}(n)$ is the pth-order input vector.

Note, from (2) and (7), that the standard pth-order Volterra kernel has one coefficient for each pth-order cross product of input samples, resulting in a number of coefficients given by

$$N_p = N^p. \tag{8}$$

This number increases exponentially with both the memory size and the order of the kernel. As a consequence, the computational cost for implementing a Volterra filter may become prohibitive even in applications involving kernels with relatively small memory size.

2.1 Redundancy-removed implementation

The large number of coefficients required to implement Volterra filters can be reduced by exploiting the redundancy of part of the coefficients of the standard nonlinear kernels [1, 15]. Such redundancy arises from the fact that coefficients with permutated indices (e.g., $h_{2(0,1)}$ and $h_{2(1,0)}$) are multiplied by the same cross product of the input signal (e.g., $x(n)x(n-1)$ in the case of $h_{2(0,1)}$ and $h_{2(1,0)}$) when the kernel output is evaluated. Thus, merging redundant coefficients into a single coefficient, the input-output relationship of the pth-order kernel, given by (2), can be rewritten as

$$y_p(n) = \sum_{m_1=0}^{N-1} \cdots \sum_{m_p=m_{p-1}}^{N-1} \underline{h}_{p(m_1,m_2,\ldots,m_p)} \prod_{k=1}^{p} x(n-m_k) \tag{9}$$

with $\underline{h}_{p(m_1,m_2,\ldots,m_p)}$ denoting the pth-order coefficients of the redundancy-removed kernel. Such representation of the kernel input-output relationship, known as triangular [1] or even redundancy-removed [15] representation, results in a number of coefficients given by

$$\underline{N}_p = \frac{(N+p-1)!}{(N-1)!\,p!}. \tag{10}$$

Moreover, it is important to highlight that the reduction in the number of coefficients from (8) to (10) obtained by using the redundancy-removed implementation comes without loss of generality (i.e., a given kernel can be equivalently implemented by using either the standard or the redundancy-removed implementation).

As in the case of the standard Volterra kernels, the input-output relationship of the redundancy-removed ones can also be represented in vector form. Thereby, we have [9]

$$y_p(n) = \underline{\mathbf{h}}_p^{\mathrm{T}} \underline{\mathbf{x}}_p(n), \tag{11}$$

where $\underline{\mathbf{h}}_p$ is the pth-order redundancy-removed coefficient vector, which is composed of coefficients $\underline{h}_{p(m_1,m_2,\ldots,m_p)}$, and $\underline{\mathbf{x}}_p(n) = \mathbf{L}_p[\underline{\mathbf{x}}_{p-1}(n) \otimes \mathbf{x}_1(n)]$ is the pth-order redundancy-removed input vector with $\mathbf{L}_p$ denoting the pth-order elimination matrix [9]. For instance, considering the second-order kernel, (9) results in

$$y_2(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} \underline{h}_{2(m_1,m_2)} x(n-m_1)x(n-m_2), \tag{12}$$

whereas (11) results in

$$y_2(n) = \underline{\mathbf{h}}_2^{\mathrm{T}} \underline{\mathbf{x}}_2(n) \tag{13}$$

with

$$\begin{aligned}\underline{\mathbf{h}}_2 = [&\underline{h}_{2(0,0)}\ \underline{h}_{2(0,1)}\ \cdots\ \underline{h}_{2(0,N-1)}\\ &\underline{h}_{2(1,1)}\ \cdots\ \underline{h}_{2(N-1,N-1)}]^{\mathrm{T}}\end{aligned} \tag{14}$$

and

$$\begin{aligned}\underline{\mathbf{x}}_2(n) = [&x^2(n)\ \ x(n)x(n-1)\ \cdots\ x(n)x(n-N+1)\\ &x^2(n-1)\ \cdots\ x^2(n-N+1)].\end{aligned} \tag{15}$$

2.2 Matrix-form kernel representation

The input-output relationship of nonlinear Volterra kernels can also be formulated as a function of coefficient matrices instead of coefficient vectors. This type of representation is especially suited for the development of reduced-rank implementations, as will be shown in the next section. For the standard second-order kernel, a matrix-form representation can be obtained by considering m_1 and m_2 as coordinates of a Cartesian space to define the following second-order coefficient matrix:

$$\mathbf{H}_2 = \begin{bmatrix} h_{2(0,0)} & h_{2(0,1)} & \cdots & h_{2(0,N-1)} \\ h_{2(1,0)} & h_{2(1,1)} & \cdots & h_{2(1,N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{2(N-1,0)} & h_{2(N-1,1)} & \cdots & h_{2(N-1,N-1)} \end{bmatrix}. \tag{16}$$

Then, from (16), the input-output relationship of the second-order kernel can be written as

$$y_2(n) = \mathbf{x}_1^{\mathrm{T}}(n)\mathbf{H}_2\mathbf{x}_1(n). \tag{17}$$

In the case of the redundancy-removed second-order kernel, the following coefficient matrix can be defined:

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_{2(0,0)} & \underline{h}_{2(0,1)} & \cdots & \underline{h}_{2(0,N-1)} \\ 0 & \underline{h}_{2(1,1)} & \cdots & \underline{h}_{2(1,N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \underline{h}_{2(N-1,N-1)} \end{bmatrix}. \tag{18}$$

Now, considering (18), (13) can be rewritten as

$$y_2(n) = \mathbf{x}_1^{\mathrm{T}}(n)\underline{\mathbf{H}}_2\mathbf{x}_1(n). \tag{19}$$

The approach presented in [24] generalizes the matrix-form representations from (16) to (19) for higher-order kernels. Such an approach is based on defining an $\underline{N}_{p_1} \times \underline{N}_{p_2}$ pth-order coefficient matrix $\hat{\mathbf{H}}_{p,p_1,p_2}$ with $p = p_1 + p_2$. This matrix contains the $\underline{N}_p$ coefficients of the pth-order redundancy-removed kernel arranged in such a way that the kernel output can be written as

$$y_p(n) = \underline{\mathbf{x}}_{p_1}^{\mathrm{T}}(n)\hat{\mathbf{H}}_{p,p_1,p_2}\underline{\mathbf{x}}_{p_2}(n), \tag{20}$$

where $\underline{\mathbf{x}}_{p_1}(n)$ and $\underline{\mathbf{x}}_{p_2}(n)$ are the redundancy-removed input vectors with orders p_1 and p_2, respectively. It is important to mention that the number of coefficients $\underline{N}_p$ of the pth-order kernel is smaller than the number of entries $(\underline{N}_{p_1} \times \underline{N}_{p_2})$ of $\hat{\mathbf{H}}_{p,p_1,p_2}$. As a result, several elements of $\hat{\mathbf{H}}_{p,p_1,p_2}$ are in fact equal to zero [24].

2.3 Reduced-rank implementations

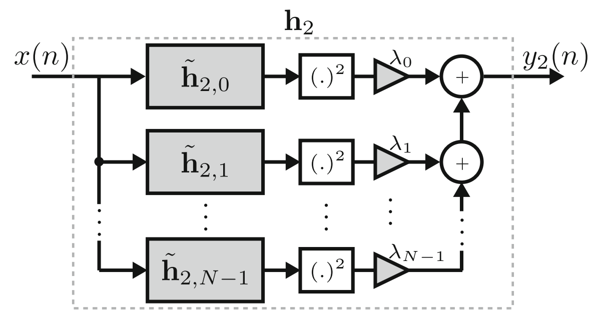
As described in [20–22, 24], approaches based on low-rank approximations can be used for obtaining effective reduced-complexity Volterra implementations. Most of these approaches are based on using the singular value decomposition along with matrix-form representations of Volterra kernels. For instance, the approach used for implementing second-order kernels described in [22] is based on the application of the singular value decomposition to the standard second-order coefficient matrix $\mathbf{H}_2$ given by (16). Since this matrix is symmetric, its left singular vectors are equal to its right singular vectors and, as a result, one obtains

$$\mathbf{H}_2 = \sum_{k=0}^{N-1} \lambda_k \tilde{\mathbf{h}}_{2,k} \tilde{\mathbf{h}}_{2,k}^{\mathrm{T}}, \tag{21}$$

where λ_k and $\tilde{\mathbf{h}}_{2,k}$ are, respectively, the kth singular value and the kth singular vector of $\mathbf{H}_2$. Now, substituting (21) into (17), one gets

$$y_2(n) = \sum_{k=0}^{N-1} \lambda_k [\,\mathbf{x}_1^{\mathrm{T}}(n) \tilde{\mathbf{h}}_{2,k}]^2\,. \tag{22}$$

Note that $\mathbf{x}_1^{\mathrm{T}}(n)\tilde{\mathbf{h}}_{2,k}$ corresponds to the input-output relationship of an FIR filter with coefficient vector given by $\tilde{\mathbf{h}}_{2,k}$. Thus, (22) is in fact the input-output relationship of the structure shown in Fig. 1, which is a parallel structure of N FIR filters with their squared outputs multiplied by the singular values of $\mathbf{H}_2$. Moreover, since the singular vectors $\tilde{\mathbf{h}}_{2,k}$ (with $k = 0, \ldots, N-1$) are unit vectors, the branches of the structure from Fig. 1 involving small values of λ_k can be disregarded, resulting in a reduction of computational complexity with low impact on the implementation precision.



$\mathbf{h}_2$
$x(n)$
$y_2(n)$
$\tilde{\mathbf{h}}_{2,0}$
$\tilde{\mathbf{h}}_{2,1}$
$\tilde{\mathbf{h}}_{2,N-1}$
$(.)^2$
λ_0
λ_1
λ_{N-1}
+

Fig. 1 Block diagram of the implementation of a second-order kernel based on the singular value decomposition

The implementation depicted in Fig. 1 is based on the standard matrix-form representation of the input-output relationship of the second-order kernel. The redundancy-removed matrix-form representation given by (19) can also be used for obtaining a reduced-rank implementation. In this case, since the left and right singular vectors are not the same (due to the fact that $\underline{\mathbf{H}}_2$ is not symmetric), the resulting input-output relationship is

$$y_2(n) = \sum_{k=0}^{N-1} \lambda_k [\,\mathbf{x}_1^{\mathrm{T}}(n) \tilde{\mathbf{h}}_{2\mathrm{l},k}]\,[\,\mathbf{x}_1^{\mathrm{T}}(n) \tilde{\mathbf{h}}_{2\mathrm{r},k}] \tag{23}$$

with $\tilde{\mathbf{h}}_{2\mathrm{l},k}$ representing the kth left singular vector of $\mathbf{H}_2$ and $\tilde{\mathbf{h}}_{2\mathrm{r},k}$ denoting the kth right singular vector. By comparing (22) and (23), one can notice that the latter (which is based on the redundancy-removed implementation) results in a structure with higher computational cost than the structure resulting from the former (based on the standard representation). Thus, one verifies that the standard representation is in fact more advantageous than the less costly redundancy-removed one for obtaining this type of reduced-rank implementations of second-order kernels.

In the case of higher-order kernels (with $p \geq 3$), one appealing approach for obtaining reduced-rank implementations is the one used in [24] to obtain the so-called parallel-cascade Volterra structures. Such an approach is based on the application of the singular value decomposition to the coefficient matrix of the general matrix-form kernel representation given by (20). For instance, considering the case of a third-order kernel, (20) becomes

$$y_3(n) = \underline{\mathbf{x}}_1^{\mathrm{T}}(n) \hat{\mathbf{H}}_{3,1,2} \underline{\mathbf{x}}_2(n) \tag{24}$$

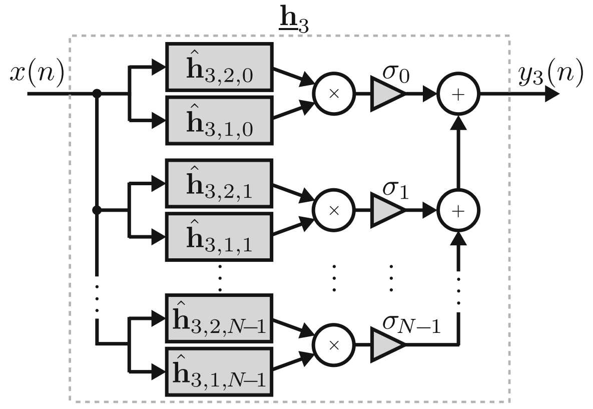
with $\underline{\mathbf{x}}_1(n) = \mathbf{x}_1(n)$. By applying the singular value decomposition to $\hat{\mathbf{H}}_{3,1,2}$, one obtains

$$\hat{\mathbf{H}}_{3,1,2} = \sum_{k=0}^{N-1} \sigma_k \hat{\mathbf{h}}_{3,1,k} \hat{\mathbf{h}}_{3,2,k}^{\mathrm{T}}, \tag{25}$$

where σ_k is the kth singular value of $\hat{\mathbf{H}}_{3,1,2}$, $\hat{\mathbf{h}}_{3,1,k}$ is the kth left singular vector, and $\hat{\mathbf{h}}_{3,2,k}$ is the kth right singular vector. Then, substituting (25) into (24) and manipulating the resulting expression, one gets

$$y_3(n) = \sum_{k=0}^{N-1} \sigma_k [\,\hat{\mathbf{h}}_{3,1,k}^{\mathrm{T}} \underline{\mathbf{x}}_1(n)]\,[\,\hat{\mathbf{h}}_{3,2,k}^{\mathrm{T}} \underline{\mathbf{x}}_2(n)]\,. \tag{26}$$

From (4), one can notice that $\hat{\mathbf{h}}_{3,1,k}^{\mathrm{T}} \underline{\mathbf{x}}_1(n)$ corresponds to the filtering of the input signal by a first-order kernel (FIR filter) with coefficient vector $\hat{\mathbf{h}}_{3,1,k}$. Similarly, considering (13), one notices that $\hat{\mathbf{h}}_{3,2,k}^{\mathrm{T}} \underline{\mathbf{x}}_2(n)$ corresponds to the filtering of the input signal by a second-order kernel with coefficient vector $\hat{\mathbf{h}}_{3,2,k}$. Thus, (26) in fact corresponds to the structure depicted in Fig. 2, which is composed of a set



$\underline{\mathbf{h}}_3$
$x(n)$
$y_3(n)$
$\hat{\mathbf{h}}_{3,2,0}$
$\hat{\mathbf{h}}_{3,1,0}$
$\hat{\mathbf{h}}_{3,2,1}$
$\hat{\mathbf{h}}_{3,1,1}$
$\hat{\mathbf{h}}_{3,2,N-1}$
$\hat{\mathbf{h}}_{3,1,N-1}$
σ_0
σ_1
σ_{N-1}

Fig. 2 Block diagram of the parallel-cascade implementation of a third-order kernel

of N branches, with the output of the kth branch given by the product of the outputs of two kernels (a first-order kernel and a second-order one), weighted by the kth singular value σ_k of $\hat{\mathbf{H}}_{3,1,2}$. As in the case of the structure of Fig. 1, reduced-complexity reduced-rank Volterra implementations can be obtained from the parallel-cascade structure of Fig. 2, removing the branches related to the smallest singular values of $\hat{\mathbf{H}}_{3,1,2}$. In addition, as mentioned in [24], the second-order blocks of the structure of Fig. 2 can be further decomposed by using reduced-rank approaches, which allows a more detailed singular-value-dependent kernel pruning. However, this pruning is not a straightforward task due to the hierarchical nature of the resulting structure (i.e., it involves reduced-rank decompositions with different levels of importance).

3 Novel reduced-rank approach for implementing Volterra filters

This section is devoted to the development of a novel reduced-rank approach for implementing Volterra filters. To this end, a new strategy for implementing higher-order kernels using a parallel structure composed of lower-order kernels is first discussed. Then, such a strategy is exploited along with the singular value decomposition to obtain the proposed reduced-rank implementation approach.

3.1 Kernel implementation redesigned

In this section, the aim is to develop a new form of Volterra kernel implementation that allows factorizing higher-order kernels in terms of lower-order ones. For the second-order kernel, such a type of implementation is obtained by rewriting the second-order redundancy-removed input-output relationship as

$$y_2(n) = \sum_{m_1=0}^{N-1} x(n-m_1) \sum_{m_2=m_1}^{N-1} \underline{h}_{2(m_1,m_2)} x(n-m_2). \quad (27)$$

The rightmost summation term in (27) corresponds to the input-output relationship of a first-order kernel (an FIR filter) with memory size $N - m_1$, coefficient vector

$$\bar{\mathbf{h}}_{2,m_1} = [\, \underline{h}_{2(m_1,m_1)} \;\; \underline{h}_{2(m_1,m_1+1)} \;\; \cdots \;\; \underline{h}_{2(m_1,N-1)}]^{\mathrm{T}} \quad (28)$$

and input vector

$$\bar{\mathbf{x}}_{2,m_1}(n) = [\, x(n-m_1) \;\; x(n-m_1-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}}. \quad (29)$$

Thus, (27) can be rewritten as

$$y_2(n) = \sum_{m_1=0}^{N-1} x(n-m_1) \bar{\mathbf{h}}_{2,m_1}^{\mathrm{T}} \bar{\mathbf{x}}_{2,m_1}(n). \quad (30)$$

Note that, in (30), the output of the second-order kernel is evaluated by summing the outputs of N first-order kernels multiplied by delayed samples of the input signal. Therefore, (30) can be seen as a decomposition of the second-order kernel into a parallel structure composed of first-order kernels.

In the case of the third-order kernel, the redundancy-removed input-output relationship can be written as

$$y_3(n) = \sum_{m_1=0}^{N-1} x(n-m_1) \times \sum_{m_2=m_1}^{N-1} \sum_{m_3=m_2}^{N-1} \underline{h}_{3(m_1,m_2,m_3)} x(n-m_2) x(n-m_3). \quad (31)$$

Now, considering (12), one can notice that the double summation in (31) corresponds to the output of a second-order kernel with coefficient vector

$$\bar{\mathbf{h}}_{3,m_1} = [\underline{h}_{3(m_1,m_1,m_1)} \;\; \underline{h}_{3(m_1,m_1,m_1+1)} \;\; \cdots \;\; \underline{h}_{3(m_1,m_1,N-1)} \;\; \underline{h}_{3(m_1,m_1+1,m_1+1)} \;\; \cdots \;\; \underline{h}_{3(m_1,N-1,N-1)}]^{\mathrm{T}} \quad (32)$$

and input vector

$$\bar{\mathbf{x}}_{3,m_1}(n) = [\, x^2(n-m_1) \;\; x(n-m_1)x(n-m_1-1) \;\; \cdots \;\; x(n-m_1)x(n-N+1) \;\; x^2(n-m_1-1) \;\; \cdots \;\; x^2(n-N+1)]^{\mathrm{T}}. \quad (33)$$

Consequently, from (13), (31) can be rewritten as

$$y_3(n) = \sum_{m_1=0}^{N-1} x(n-m_1) \bar{\mathbf{h}}_{3,m_1}^{\mathrm{T}} \bar{\mathbf{x}}_{3,m_1}(n) \quad (34)$$

which corresponds to the decomposition of the third-order kernel into a structure composed of second-order kernels.

Similarly to (30) and (34), for the pth-order kernel, the following input-output relationship can be obtained:

$$y_p(n) = \sum_{m_1=0}^{N-1} x(n-m_1)\bar{\mathbf{h}}_{p,m_1}^{\mathrm{T}} \bar{\mathbf{x}}_{p,m_1}(n), \tag{35}$$

where the product $\bar{\mathbf{h}}_{p,m_1}^{\mathrm{T}} \bar{\mathbf{x}}_{p,m_1}(n)$ corresponds to the input-output relationship of a $(p-1)$th-order kernel with memory size $N - m_1$. Thus, from (35), we can infer that any pth-order kernel can be decomposed into N kernels with order $p-1$, as illustrated in Fig. 3.

3.2 Proposed approach

Aiming to develop the proposed reduced-rank approach for implementing Volterra filters, we first consider that the kernel implementation strategy introduced in Section 3.1 can be used to decompose any higher-order kernel (with $p \geq 3$) into a parallel structure composed exclusively of second-order kernels. This decomposition is straightforward for the third-order kernel, since, by using $p = 3$ in (35), one obtains a structure composed of N second-order kernels, as described by (31)–(34). In the case of the fourth-order kernel, (35) can be used first for obtaining a structure composed of N third-order kernels and then for decomposing each of these third-order kernels into second-order kernels. As a result, a structure composed of N^2 second-order kernels is obtained for the implementation of the fourth-order kernel. Following this rationale, one can notice that, by using (35) to carry out successive kernel decompositions, an implementation composed of $N^{(p-2)}$ second-order kernels can always be obtained for a pth-order kernel.

Now, the idea behind the proposed reduced-rank approach for implementing Volterra filters is to exploit the fact that any pth-order kernel can be decomposed into a parallel structure of second-order kernels as previously described. Taking into account this fact, a reduced-rank implementation for the pth-order kernel can be obtained by applying, to each second-order kernel resulting from the kernel decomposition, the strategy used for obtaining the reduced-rank implementation of Fig. 1 (see Section 2.3). Thus, one obtains a structure composed of $N^{(p-2)}$ blocks, each having the form of the structure of Fig. 1. Then, disregarding the branches of these blocks related to the smaller singular values, a reduced-complexity reduced-rank kernel implementation is obtained. In this context, the proposed approach can be summarized as follows:

i) Exploit the strategy described in Section 3.1 (see Fig. 3) to obtain an implementation of the pth-order kernel of interest in the form of a parallel structure composed of second-order kernels.
ii) Use the standard matrix-form representation (see Section 2.2) to represent all second-order kernels that compose the kernel of interest.
iii) Obtain reduced-rank implementations of such second-order kernels by using the singular value decomposition as described in Section 2.3.
iv) Remove (prune) the branches of the resulting structure related to the smaller singular values of the involved second-order coefficient matrices.

The proposed reduced-rank approach for implementing Volterra filters consists of the application of these four steps to all kernels, which allows obtaining effective reduced-complexity Volterra filter implementations.

4 Simulation results

This section aims to assess the effectiveness of the proposed reduced-rank approach for obtaining reduced-complexity implementations of Volterra filters. To this end, the proposed approach is compared with the parallel-cascade (PC) one from [24] in the context of the implementation of third-order and forth-order kernels whose coefficients are known in advance. The effectiveness of these approaches is assessed in terms of normalized misalignment [26], which is defined as

$$\mathcal{M} = 10\log_{10}\left(\frac{||\underline{\mathbf{h}}_{\mathrm{k}} - \underline{\mathbf{h}}_{\mathrm{rr}}||_2^2}{||\underline{\mathbf{h}}_{\mathrm{k}}||_2^2}\right) \tag{36}$$

with $\underline{\mathbf{h}}_{\mathrm{k}}$ denoting the coefficient vector of the kernel to be implemented, and $\underline{\mathbf{h}}_{\mathrm{rr}}$, the coefficient vector obtained by using the reduced-rank approach. A hierarchical branch-level pruning is applied to the parallel structures obtained by using the approaches considered, which means that one branch is removed at a time, with the branches related to the smallest singular values removed first. After the removal of each branch, both the normalized misalignment and the number of required arithmetic operations are evaluated, resulting in the curves used here

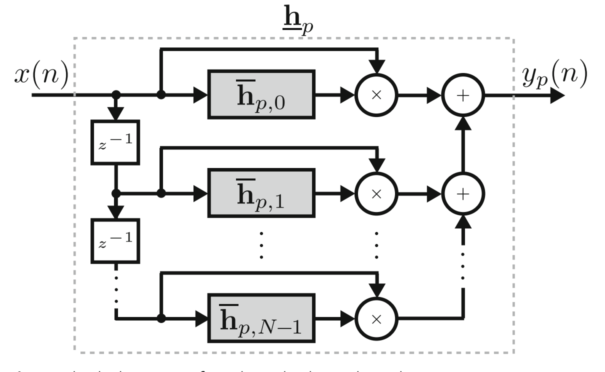


Fig. 3 Block diagram of a pth-order kernel implementation using a parallel structure composed of $(p-1)$th-order kernels

for comparing the different approaches. Markers are presented along these curves, indicating each evaluated pair of normalized misalignment and number of operations per sample.

4.1 Example 1: modeling of a third-order kernel

The third-order kernel considered in this example is obtained from a system modeling experiment in which a diode limiter circuit used in guitar distortion pedals [27, 28] is modeled using an LMS-adapted Volterra filter (with memory size $N_a = 10$, sampling rate of $f_s = 44.1$ kHz, and white input signal). The PC and the proposed reduced-rank approaches are used here for implementing the third-order kernel of the Volterra model obtained in such an experiment. As a result, the curves of normalized misalignment as a function of the number of operations per sample shown in Fig. 4 are obtained. A vertical dotted line pointing out the number of operations per sample required by the corresponding redundancy-removed Volterra implementation (around 495 operations) is also included in this figure aiming to establish a limit after which it is no longer interesting to use reduced-rank implementations. One can notice from Fig. 4 that the proposed approach outperforms the PC one for the case considered here; since, a smaller normalized misalignment is obtained for any given number of operations per sample. For instance, if a misalignment below −15 dB is desired, a PC-based implementation would require at least 280 operations per sample (i.e., a reduction of about 43% with respect to the number of coefficients of the redundancy-removed Volterra kernel), whereas an implementation based on the proposed approach would require only 139 operations per sample (a reduction of almost 72%).

4.2 Example 2: modeling of a fourth-order kernel

The kernel considered in this example is similar to the fourth-order satellite-system model used in [24]. Such a model is obtained by using a cascade of a Butterworth low-pass filter with a memoryless fourth-power nonlinearity and a Chebyshev low-pass filter. For the sake of simplicity, we consider a version of this model in which the Butterworth and Chebyshev IIR filters are replaced by FIR versions obtained by truncating their impulse responses to 30 samples. Then, a fourth-order kernel with memory size 59 is obtained, which is again truncated to memory size 30. As a result, this kernel has a total of $\underline{N}_4 = 40,920$ coefficients in its redundancy-removed representation, requiring around 86,800 operations per sample for its implementation. It is important to mention that this kernel admits a symmetric representation for its coefficient matrix $\hat{\mathbf{H}}_{4,2,2}$. Consequently, the branches of PC-based structures will be composed of a single second-order kernel with its squared output multiplied by one of the singular values of $\hat{\mathbf{H}}_{4,2,2}$. The results obtained for this example are shown in Fig. 5. For a better visualization (due to the high density of the obtained points), we have used one marker for each 30 points in the curve of the proposed approach. From such results, one notices that the reduced-rank approaches are capable of modeling the considered fourth-order kernel with very good accuracy. For instance, −60 dB of misalignment is obtained through the proposed approach with around 12,520 operations per sample, which corresponds to almost 86% of complexity reduction. Moreover, Fig. 5 also shows that the proposed approach outperforms the PC approach in terms of trade-off between performance and complexity either for the range of computational cost from 0 to almost 27,000 operations per sample or for the range of misalignment from 0 to about −120 dB.

5 Conclusions

In this paper, a novel reduced-rank approach for implementing higher-order Volterra filters was introduced. Such an approach is based on a new form of kernel implementation that allows converting any higher-order kernel into a structure composed exclusively of second-order kernels. Then, exploiting the singular value

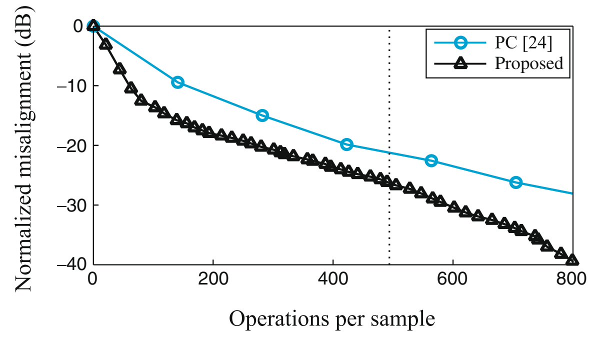


Fig. 4 Number of coefficients required by different reduced-rank implementations of a third-order kernel

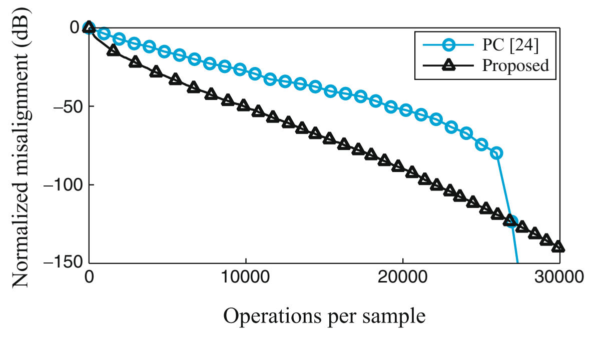


Fig. 5 Number of coefficients required by different reduced-rank implementations of a fourth-order kernel

decomposition along with the coefficient matrices of these second-order kernels, a reduced-rank implementation for the higher-order Volterra filter can be achieved. Numerical simulation results were shown, confirming the effectiveness of the proposed approach in obtaining reduced-complexity implementations of Volterra filters.

Acknowledgements
The authors are thankful to Gabriel Celso Kulevicz da Silva for providing the experimental data used in Example 1 of Section 4. Thanks are also due to the editors and the anonymous reviewers for their valuable and constructive comments and suggestions, from which the revision of this paper has benefited significantly.

Competing interests
The authors declare that they have no competing interests.

Received: 28 June 2016 Accepted: 2 November 2016

References
1. VJ Mathews, GL Sicuranza, *Polynomial Signal Processing*. (John Wiley & Sons, Inc., New York, 2000), p. 452
2. S Boyd, LO Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series. IEEE Trans. Circ. Syst. **CAS-32**(11), 1150–1161 (1985)
3. IW Sandberg, R+ fading memory and extensions of input-output maps. IEEE Trans. Circ. Syst. I, Reg. Papers. **49**(11), 1586–1591 (2002)
4. S Orcini, Improving the approximation ability of Volterra series identified with a cross-correlation method. Nonlinear Dyn. **78**(4), 2861–2869 (2014)
5. A Carini, GL Sicuranza, in *IEEE Int. Conf. Acoust. Speech Signal Process*. Even mirror Fourier nonlinear filters (IEEE, Vancouver, 2013), pp. 5608–5612
6. A Carini, GL Sicuranza, Fourier nonlinear filters. Signal Process. **94**, 183–194 (2014)
7. A Carini, S Cecchi, L Romoli, GL Sicuranza, Legendre nonlinear filters. Signal Process. **109**, 84–94 (2015)
8. A Carini, GL Sicuranza, A study about Chebyshev nonlinear filters. Signal Process. **122**, 24–32 (2016)
9. ELO Batista, R Seara, On the performance of adaptive pruned Volterra filters. Signal Process. **93**(7), 1909–1920 (2013)
10. L Tan, J Jiang, An adaptive technique for modeling second-order Volterra systems with sparse kernels. IEEE Trans. Circ. Syst. II: Analog Digit. Sig. Proc. **45**(12), 1610–1615 (1998)
11. F Kuech, W Kellermann, Orthogonalized power filters for nonlinear acoustic echo cancellation. Signal Process. **86**(6), 1168–1181 (2006)
12. A Fermo, A Carini, GL Sicuranza, Low-complexity nonlinear adaptive filters for acoustic echo cancellation in GSM handset receivers. Eur. Trans. Telecommun. **14**(2), 161–169 (2003)
13. M Zeller, W Kellermann, in *Proc. Int. Work. Acoustic Echo and Noise Control (IWAENC)*. Coefficient pruning for higher-order diagonals of Volterra filters representing Wiener-Hammerstein models, (Seattle, 2008)
14. M Zeller, LA Azpicueta-Ruiz, J Arenas-Garcia, W Kellermann, Adaptive Volterra filters with evolutionary quadratic kernels using a combination scheme for memory control. IEEE Trans. Signal Process. **59**(4), 1449–1464 (2011)
15. ELO Batista, OJ Tobias, R Seara, A sparse-interpolated scheme for implementing adaptive Volterra filters. IEEE Trans. Signal Process. **58**(4), 2022–2035 (2010)
16. ELO Batista, R Seara, in *Proc. 19th Eur. Signal Process. Conf.* Adaptive NLMS diagonally-interpolated Volterra filters for network echo cancellation (IEEE/EURASIP, Barcelona, 2011), pp. 1430–1434
17. ELO Batista, R Seara, A fully LMS/NLMS adaptive scheme applied to sparse-interpolated Volterra filters with removed boundary effect. Signal Process. **92**(10), 2381–2393 (2012)
18. MJ Reed, MOJ Hawksford, Efficient implementation of the Volterra filter. IEE Proc.-Vis. Image Signal Process. **147**(2), 109–114 (2000)
19. F Kuech, W Kellerman, Partitioned block frequency-domain adaptive second-order Volterra filter. IEEE Trans. Signal Process. **53**(2), 564–575 (2005)
20. H-H Chiang, CL Nikias, AN Venetsanopoulos, Efficient implementations of quadratic digital filters. Trans. Acoust., Speech, Signal Process. **34**(6), 1511–1528 (1986)
21. Y Lou, CL Nikias, AN Venetsanopoulos, Efficient VLSI array processing structures for adaptive quadratic digital filters. Circuits, Syst. Signal Process. **7**(2), 253–273 (1988)
22. S Marsi, GL Sicuranza, in *Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput, vol. 2*. On reduced-complexity approximations of quadratic filters (IEEE, Pacific Grove, 1993), pp. 1026–1030
23. RD Nowak, BD Van Veen, Tensor product basis approximations for Volterra filters. IEEE Trans. Signal Process. **44**(1), 36–50 (1996)
24. TM Panicker, VJ Mathews, GL Sicuranza, Adaptive parallel-cascade truncated Volterra filters. IEEE Trans. Signal Process. **46**(10), 2664–2673 (1998)
25. G Favier, AY Kibangou, T Bouilloc, Nonlinear system modeling and identification using Volterra-PARAFAC models. Int. J. Adapt. Control Signal Process. **26**(1), 30–53 (2012)
26. J Benesty, C Paleologu, S Ciochina, *Sparse Adaptive Filters for Echo Cancellation*. (Morgan and Claypool Publishers, San Rafael, CA, 2010), p. 124
27. DT Yeh, JS Abel, JO Smith, in *Proc. 10th Conf. on Digital Audio Effects (DAFx-07)*. Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations, (Bordeaux, France, 2007)
28. GCK da Silva, Contribuições para Implementação Digital de um Pedal de Efeito de Áudio do Tipo Overdrive (in Portuguese). Master's thesis, Federal University of Santa Catarina, Florianópolis, Brazil, 2016