# Nonlinear system modeling and identification using Volterra-PARAFAC models

Gérard Favier, Alain Y. Kibangou, Thomas Bouilloc

INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
*Int. J. Adapt. Control Signal Process.* 2012; **26**:30–53
Published online 21 September 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.1272

# Nonlinear system modeling and identification using Volterra-PARAFAC models

Gérard Favier [1,*,†], Alain Y. Kibangou [2] and Thomas Bouilloc [1]

[1] *Laboratoire I3S, University of Nice Sophia Antipolis, CNRS, Les Algorithmes - Bât. Euclide B, 2000 Route des lucioles, B.P. 121-06903 Sophia Antipolis, Cedex, France*
[2] *Gipsa-Lab, Systems Control Department, University Joseph Fourier, CNRS, 961 rue de la Houille Blanche, B.P. 46-38402 Grenoble, Cedex, France*

SUMMARY

Discrete-time Volterra models are widely used in various application areas. Their usefulness is mainly because of their ability to approximate to an arbitrary precision any fading memory nonlinear system and to their property of linearity with respect to parameters, the kernels coefficients. The main drawback of these models is their parametric complexity implying the need to estimate a huge number of parameters. Considering Volterra kernels of order higher than two as symmetric tensors, we use a parallel factor (PARAFAC) decomposition of the kernels to derive Volterra-PARAFAC models that induce a substantial parametric complexity reduction. We show that these models are equivalent to a set of Wiener models in parallel. We also show that Volterra kernel expansions onto orthonormal basis functions (OBF) can be viewed as Tucker models that we shall call Volterra-OBF-Tucker models. Finally, we propose three adaptive algorithms for identifying Volterra-PARAFAC models when input–output signals are complex-valued: the extended complex Kalman filter, the complex least mean square (CLMS) algorithm and the normalized CLMS algorithm. Some simulation results illustrate the effectiveness of the proposed identification methods.


Received 6 April 2010; Revised 11 May 2011; Accepted 4 July 2011

KEY WORDS: CLMS algorithm; extended complex Kalman filter; nonlinear system identification; nonlinear system modeling; PARAFAC; tensor decompositions; Volterra kernels; Volterra models; Wiener models

## 1. INTRODUCTION

Empirical modeling from measured input–output data is a key issue for many applications. Such a modeling consists in first selecting an appropriate model structure and then estimating the model parameters by processing the input–output signals with an identification method. Most real-life systems are nonlinear in nature so that nonlinear models are often preferable for well representing the system under study. Finite-dimensional discrete-time Volterra models, also called truncated Volterra series expansions or nonrecursive polynomial models, are now widely used in various fields of applications like speech modeling [1], loudspeaker linearization [2, 3], nonlinear control of chemical processes [4–6], active noise control [7], nonlinear acoustic echo cancellation [8], modeling of biological and physiological systems [9, 10], nonlinear communication channel identification and equalization [11–16], and many others.

The usefulness of Volterra models is mainly because of their ability to approximate to an arbitrary accuracy any fading memory nonlinear system [17] and also their property of linearity with respect to parameters, the kernels coefficients. Moreover, they represent a nonlinear extension of the

*Correspondence to: Gérard Favier, Laboratoire I3S, University of Nice Sophia Antipolis, CNRS, Les Algorithmes - Bât. Euclide B, 2000 Route des lucioles, B.P. 121-06903 Sophia Antipolis, Cedex, France.
†E-mail: favier@i3s.unice.fr

very popular finite impulse response (FIR) linear model, with guaranteed stability in the bounded-input bounded-output sense. They are interpretable in terms of multidimensional convolutions which makes easy the derivation of their z-transform and Fourier transform representations, and therefore their frequency-domain interpretation [18]. The main drawback of these models concerns their parametric complexity implying the need to estimate a huge number of parameters. This number increases rapidly with the system nonlinearity order and memory.

There are two main approaches for reducing the number of free parameters in Volterra models. One approach consists in considering block-oriented nonlinear models constituted by a cascade of linear dynamic subsystems and memoryless (static) nonlinearities. The order of the blocks defines the model subclass: Wiener models (L-N: liear-nonlinear) are composed of a linear dynamic subsystem followed by a nonlinear static one, Hammerstein models (N-L: nonlinear-linear) are obtained by reversing the order of these blocks, Wiener-Hammerstein models (L-N-L: linear-nonlinear-linear) are constituted by a static nonlinearity in sandwich between two linear subsystems, whereas Hammerstein-Wiener models (N-L-N: nonlinear-linear-nonlinear) comprise two static nonlinearities sandwiching a linear subsystem. Compared with Volterra models, the block-oriented ones are characterized by a smaller number of parameters but their output is nonlinear with respect to the parameters, which leads to a nonlinear estimation problem. Moreover, this kind of models allows taking some specific nonlinearities into account, as for instance actuator's and sensor's nonlinearities in control applications. Because of their wide relevance, there exist a lot of methods for identifying block-oriented systems. We refer the reader to [4, 19, 20] and references therein for a description of block-oriented model applications and identification methods. Another approach, first suggested in [21], consists in expanding the Volterra kernels onto orthonormal basis functions (OBFs). Such kernel expansions can lead to parsimonious modeling when the basis functions are adequately chosen. The Laguerre and Kautz functions, characterized respectively by a single real-valued parameter, the Laguerre pole, and a pair of complex conjugate poles, have been used for approximating stable linear systems in [22–25]. Such OBFs are well suited to model systems with first-order or second-order dominant dynamics. For more complex dynamics, more general rational basis functions, called generalized OBFs (GOBFs), have been developed in the context of linear system identification [26, 27]. Laguerre expansions of Volterra kernels have been used for identifying biological systems [28] and for nonlinear adaptive control [29]. More recently, Volterra models using OBFs have been proposed for modeling radio frequency power amplifiers [30, 31]. Optimization of the poles of Laguerre–Volterra, Kautz–Volterra, and GOBF–Volterra models has been addressed in [32–37].

Two other alternatives for reducing the Volterra model complexity consist in using either a tensor product basis approximation method [38], or a parallel-cascade realization resulting from the singular value decomposition (SVD) of a matrix representation of the kernel of a homogeneous Volterra system [39, 40].

In the present paper, considering higher-order Volterra kernels as symmetric tensors, we use a parallel factor (PARAFAC) decomposition [41] of these kernels for deriving the Volterra-PARAFAC model. This approach was introduced for the first time in [42] without using the symmetry property of the kernels. Part of the results presented hereafter has been recently addressed in [43] and [44]. Then, three adaptive algorithms are proposed for estimating the parameters of such a model: the extended complex Kalman filter (ECKF), the complex least mean square (CLMS) algorithm, and the normalized CLMS (NCLMS) algorithm.

The rest of this paper is organized as follows. Section 2 introduces notations and basic matrix operations, whereas Section 3 reviews the required mathematical background on tensor operations and models, more particularly on the Tucker and PARAFAC decompositions. In Section 4, after a brief presentation of the Volterra model and its symmetrization, we apply the PARAFAC decomposition to symmetric Volterra kernels to derive the Volterra-PARAFAC model. We show that such a model can be viewed as a parallel cascade model constituted of simplified Wiener paths. In Section 5, we state that Volterra models using kernel expansions onto the GOBFs can be interpreted as Tucker models that we shall call Volterra-GOB-Tucker models. Then, in Section 6, we propose three adaptive methods for identifying Volterra-PARAFAC models: the ECKF, CLMS, and NCLMS algorithms. Some simulation results are presented in Section 7, to illustrate the effectiveness of Volterra-PARAFAC models for reducing the parametric complexity and then the performance



DOI: 10.1002/acs

of the proposed adaptive parameter estimation algorithms. Finally, the conclusions are drawn in Section 8.

## 2. NOTATIONS

$\mathcal{R}$ and $\mathcal{C}$ denote the fields of real and complex numbers, respectively. Scalars, vectors, matrices, and higher-order tensors are written as lower-case $(a, b, \ldots)$, bold lower-case $(\mathbf{a}, \mathbf{b}, \ldots)$, bold upper-case $(\mathbf{A}, \mathbf{B}, \ldots)$, and blackboard $(\mathbb{A}, \mathbb{B} \cdots)$ letters, respectively. $\mathbf{A}^T$, $\mathbf{A}^H$, $\mathbf{A}^*$, and $\mathbf{A}^\dagger$ stand for transpose, transconjugate (Hermitian transpose), complex conjugate, and Moore–Penrose pseudo-inverse of $\mathbf{A}$, respectively. The vector $\mathbf{A}_{i.}$ (resp. $\mathbf{A}_{.j}$) represents the $i^{th}$ row (resp. $j^{th}$ column) of $\mathbf{A}$. The scalars $a_i$, $a_{i,j}$, and $a_{i_1,\cdots,i_N}$ denote, respectively, the $i^{th}$ element of $\mathbf{a}$, the $(i,j)^{th}$ element of $\mathbf{A}$ and the $(i_1,\cdots,i_N)^{th}$ element of $\mathbb{A}$. $\mathbf{I}_N$ is the identity matrix of order $N$, and $\|.\|_F$ is the Frobenius norm. The operator vec(.) forms a vector by stacking the columns of its matrix argument. The outer, Kronecker and Khatri–Rao (column-wise Kronecker) products are denoted by $\circ$, $\otimes$, and $\odot$, respectively.

- For $\mathbf{A} \in \mathcal{C}^{I\times R}$ and $\mathbf{B} \in \mathcal{C}^{J\times R}$:

$$\mathbf{A} \odot \mathbf{B} = (\mathbf{A}_{.1} \otimes \mathbf{B}_{.1} \cdots \mathbf{A}_{.R} \otimes \mathbf{B}_{.R}) \in \mathcal{C}^{IJ\times R} \tag{1}$$

- For $\mathbf{A}^{(n)} \in \mathcal{C}^{I_n\times R_n}$, $n = 1, 2, \cdots N$:

$$\bigotimes_{n=1}^{N} \mathbf{A}^{(n)} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)} \in \mathcal{C}^{I_1\cdots I_N \times R_1 \cdots R_N}$$

$$\bigodot_{n=1}^{N} \mathbf{A}^{(n)} = \mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(N)} \in \mathcal{C}^{I_1\cdots I_N \times R}$$

  when $R_n = R$, $\forall\, n = 1, 2, \cdots N$.

- For $\mathbf{u}^{(n)} \in \mathcal{C}^{I_n\times 1}$, $n = 1, 2, \cdots N$:

$$\bigcirc_{n=1}^{N} \mathbf{u}^{(n)} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \cdots \circ \mathbf{u}^{(N)} \in \mathcal{C}^{I_1\times\cdots\times I_N}$$

  with

$$\left(\bigcirc_{n=1}^{N} \mathbf{u}^{(n)}\right)_{i_1,\cdots,i_N} = \prod_{n=1}^{N} u_{i_n}^{(n)} \tag{2}$$

  In particular, for $\mathbf{u} \in \mathcal{C}^{I\times 1}$, $\mathbf{v} \in \mathcal{C}^{J\times 1}$, and $\mathbf{w} \in \mathcal{C}^{K\times 1}$, $\mathbf{u} \circ \mathbf{v} \circ \mathbf{w}$ is a rank-one third-order tensor such as:

$$\mathbf{u} \circ \mathbf{v} \circ \mathbf{w} \in \mathcal{C}^{I\times J\times K} \Leftrightarrow (\mathbf{u} \circ \mathbf{v} \circ \mathbf{w})_{i,j,k} = u_i v_j w_k.$$

## 3. TENSOR PREREQUISITES

### 3.1. Some definitions

For an $N^{th}$-order tensor $\mathbb{H} \in \mathcal{C}^{I_1\times\cdots\times I_N}$, also called an $N$-way array, of dimensions $I_1 \times \cdots \times I_N$, with entries $h_{i_1,\cdots,i_N} \in \mathcal{C}$ ($i_n = 1, 2, \cdots, I_n$, for $n = 1, 2, \cdots, N$), each index $i_n$ is associated with a way, also called a mode, and $I_n$ is the mode-$n$ dimension. The tensor $\mathbb{H}$ is characterized by $\prod_{n=1}^{N} I_n$ scalar coefficients $h_{i_1,\cdots,i_N}$. A vector is a first-order tensor, whereas a matrix is a second-order one. Tensors of order greater than two are called higher-order tensors. The Frobenius norm of $\mathbb{H}$ is given by:

$$\|\mathbb{H}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} \left|h_{i_1,\cdots,i_n}\right|^2} \tag{3}$$



DOI: 10.1002/acs

The tensor $\mathbb{H}$ is said to be hypercubic if its $N$ dimensions $I_n$ are equal; that is, $I_1 = \cdots = I_N = I$. A hypercubic tensor is said to be symmetric if its elements $h_{i_1,\cdots,i_N}$ do not change under any permutation of their indices. It is diagonal if $h_{i_1,\cdots,i_N} \neq 0$ only for $i_1 = i_2 = \cdots = i_N$. The identity tensor of order N, denoted by $\mathbb{I}_N$ or simply $\mathbb{I}$, is a diagonal hypercubic tensor with all the elements of its main diagonal equal to 1:

$$h_{i_1,\cdots,i_N} = \delta_{i_1,\cdots,i_N}, \quad i_n = 1,\cdots,I \quad \text{for } n = 1,\cdots,N$$

where $\delta$ is the generalized Kronecker symbol:

$$\delta_{i_1,\cdots,i_N} = \begin{cases} 1 & \text{if } i_1 = i_2 = \cdots = i_N \\ 0 & \text{otherwise} \end{cases}$$

By slicing the tensor along its mode-$n$, that is, by fixing the index $i_n$, we get a tensor of order $N-1$ and dimensions $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$, called the $i_n^{th}$ mode-$n$ slice of $\mathbb{H}$ and denoted by $\mathbb{H}_{\cdots,i_n,\cdots}$. The mode-$n$ vectors are the $I_n$ dimensional vectors obtained from $\mathbb{H}$ by varying the index $i_n$, while keeping the other indices fixed.

Different matricizations, also called matrix unfoldings, of $\mathbb{H} \in \mathcal{C}^{I_1 \times \cdots \times I_N}$ can be obtained. For instance, we define the horizontal mode-$n$ matrix unfolding, denoted by $\mathbf{H}_n$, as the $I_n \times I_{n+1} \cdots I_N I_1 \cdots I_{n-1}$ matrix whose columns are the mode-$n$ vectors. The dimension of the mode-$n$ vector space spanned by the mode-$n$ vectors is called the mode-$n$ rank of $\mathbb{H}$.

In the case of a third-order tensor $\mathbb{H} \in \mathcal{C}^{I \times J \times K}$, we have three types of matrix slices denoted by $\mathbf{H}_{i..}$, $\mathbf{H}_{.j.}$, and $\mathbf{H}_{..k}$ of respective dimensions $K \times J$, $I \times K$, and $J \times I$.

By column-wise stacking the matrix slices of the same type, we get the three following horizontal matrix unfoldings:

$$\mathbf{H}_1 = [\mathbf{H}_{.1.} \cdots \mathbf{H}_{.J.}] \in \mathcal{C}^{I \times JK} \tag{4}$$

$$\mathbf{H}_2 = [\mathbf{H}_{..1} \cdots \mathbf{H}_{..K}] \in \mathcal{C}^{J \times KI} \tag{5}$$

$$\mathbf{H}_3 = [\mathbf{H}_{1..} \cdots \mathbf{H}_{I..}] \in \mathcal{C}^{K \times IJ} \tag{6}$$

which implies

$$h_{i,j,k} = (\mathbf{H}_1)_{i,(j-1)K+k} = (\mathbf{H}_2)_{j,(k-1)I+i} = (\mathbf{H}_3)_{k,(i-1)J+j}$$

Other matrix unfoldings are possible. However, it is to be noticed that all the unfolded matrix representations of a given tensor are different from the point of view of tensor elements arrangement but are equivalent in terms of contained information because each one contains all the tensor elements.

The mode-$n$ product of a tensor $\mathbb{H} \in \mathcal{C}^{I_1 \times \cdots \times I_N}$ of order $N$ with a matrix $\mathbf{A} \in \mathcal{C}^{J_n \times I_n}$, denoted by $\mathbb{V} = \mathbb{H} \times_n \mathbf{A}$, gives a tensor of order $N$ and dimensions $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$ such as:

$$v_{i_1,\cdots,i_{n-1},j_n,i_{n+1},\cdots,i_N} = \sum_{i_n=1}^{I_n} h_{i_1,\cdots,i_{n-1},i_n,i_{n+1},\cdots,i_N} a_{j_n,i_n} \tag{7}$$

This mode-$n$ product can be expressed in terms of horizontal mode-$n$ matrix unfoldings of tensors $\mathbb{V}$ and $\mathbb{H}$ as:

$$\mathbf{V}_n = \mathbf{A}\mathbf{H}_n \tag{8}$$

For $\mathbb{H} \in \mathcal{C}^{I_1 \times \cdots \times I_N}$, $\mathbf{A} \in \mathcal{C}^{J_m \times I_m}$, and $\mathbf{B} \in \mathcal{C}^{J_n \times I_n}$, with $m \neq n$, we have the following property:

$$(\mathbb{H} \times_m \mathbf{A}) \times_n \mathbf{B} = (\mathbb{H} \times_n \mathbf{B}) \times_m \mathbf{A} = \mathbb{H} \times_m \mathbf{A} \times_n \mathbf{B} \tag{9}$$



DOI: 10.1002/acs

### 3.2. Tucker and PARAFAC models

For a tensor $\mathbb{H} \in \mathcal{C}^{I_1 \times \cdots \times I_N}$, the Tucker model is defined as follows [45]:

$$h_{i_1,\cdots,i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\cdots,r_N} \prod_{n=1}^{N} a^{(n)}_{i_n,r_n} \tag{10}$$

with $i_n = 1, \cdots, I_n$ for $n = 1, \cdots, N$, where $g_{r_1,\cdots,r_N}$ is an element of the core tensor $\mathbb{G} \in \mathcal{C}^{R_1 \times \cdots \times R_N}$ and $a^{(n)}_{i_n,r_n}$ is an element of the matrix factor $\mathbf{A}^{(n)} \in \mathcal{C}^{I_n \times R_n}$.

Taking the relation (2) into account, the Tucker model can be written in terms of outer products of the columns of its matrix factors:

$$\mathbb{H} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\cdots,r_N} \bigcirc_{n=1}^{N} \mathbf{A}^{(n)}_{.r_n} \tag{11}$$

showing that $\mathbb{H}$ is decomposed into a weighted sum of $\prod_{n=1}^{N} R_n$ outer products.

Using definition (7) and property (9) of the mode-$n$ product, the model (10) can also be written as:

$$\mathbb{H} = \mathbb{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)} \tag{12}$$

which will be concisely noted as

$$\mathbb{H} = \mathbb{G} \times_{n=1}^{N} \mathbf{A}^{(n)} \tag{13}$$

For the horizontal mode-$n$ matrix unfolding of the Tucker model of $\mathbb{H} \in \mathcal{C}^{I_1 \times \cdots \times I_N}$, we have:

$$\mathbf{H}_n = \mathbf{A}^{(n)} \mathbf{G}_n \left[ \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)} \otimes \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \right]^T \tag{14}$$

where $\mathbf{G}_n$ is the horizontal mode-$n$ matrix unfolding of the core tensor $\mathbb{G}$.

When each matrix factor $\mathbf{A}^{(n)}$, for $n = 1, \cdots, N$ with $N \geqslant 3$, is unitary and calculated as the left matrix of the SVD of the horizontal unfolded matrix $\mathbf{H}_n$, the Tucker model can be viewed as an extension of the matrix SVD to the $N^{th}$-order. It is then called higher-order SVD [46].

The PARAFAC model, introduced in [41], corresponds to the particular case of a Tucker model with an identity core tensor ($\mathbb{G} = \mathbb{I}_R = \mathbb{I}$), which implies $R_n = R$, $\forall\, n = 1, 2, \cdots, N$.

Equations (10)–(13) then become:

$$h_{i_1,\cdots,i_N} = \sum_{r=1}^{R} \prod_{n=1}^{N} a^{(n)}_{i_n,r} \tag{15}$$

$$\mathbb{H} = \sum_{r=1}^{R} \left( \bigcirc_{n=1}^{N} \mathbf{A}^{(n)}_{.r} \right) \tag{16}$$

and

$$\mathbb{H} = \mathbb{I}_R \times_{n=1}^{N} \mathbf{A}^{(n)} \tag{17}$$

with the matrix factors $\mathbf{A}^{(n)} \in \mathcal{C}^{I_n \times R}, n = 1, \cdots, N$.

The horizontal unfolded matrix forms associated with the PARAFAC model of $\mathbb{H}$ are given by:

$$\mathbf{H}_n = \mathbf{A}^{(n)} \left[ \mathbf{A}^{(n+1)} \odot \cdots \odot \mathbf{A}^{(N)} \odot \mathbf{A}^{(1)} \odot \cdots \odot \mathbf{A}^{(n-1)} \right]^T \tag{18}$$

For a third-order tensor $\mathbb{H} \in \mathcal{C}^{I \times J \times K}$, with matrix factors $(\mathbf{A}, \mathbf{B}, \mathbf{C})$, the horizontal unfolded matrix forms corresponding to (18) are:

$$\mathbf{H}_1 = \mathbf{A} \, (\mathbf{B} \odot \mathbf{C})^T \tag{19}$$



DOI: 10.1002/acs

$$\mathbf{H}_2 = \mathbf{B}\,(\mathbf{C} \odot \mathbf{A})^T \tag{20}$$

$$\mathbf{H}_3 = \mathbf{C}\,(\mathbf{A} \odot \mathbf{B})^T \tag{21}$$

Equation (16) shows that PARAFAC corresponds to the decomposition of the $N^{th}$-order tensor $\mathbb{H}$ into a sum of $R$ rank-one tensors; that is, a sum of $R$ outer products of $N$ vectors. The minimum number $R$ of rank-one tensors which are needed to generate $\mathbb{H}$ is called the rank of $\mathbb{H}$, denoted by rank($\mathbb{H}$), as first introduced in [47] and then in [48].

So, an $N^{th}$-order rank-one tensor $\mathbb{H}$ is a tensor that is expressible as the outer product of $N$ vectors:

$$h_{i_1,\cdots,i_N} = \prod_{n=1}^{N} a_{i_n}^{(n)}, \qquad i_n = 1,\cdots,I_n \tag{22}$$

where $a_{i_n}^{(n)}$ is an entry of the vector factor $\mathbf{a}^{(n)} \in \mathcal{C}^{I_n \times 1}$.

*Theorem 1 (*[49]*)*
Any symmetric $N^{th}$-order tensor $\mathbb{H} \in \mathcal{C}^{I \times \cdots \times I}$ can always be decomposed as the sum of symmetric outer products of vectors:

$$\mathbb{H} = \sum_{r=1}^{R_S} \underbrace{\mathbf{A}_r \circ \cdots \circ \mathbf{A}_r}_{N \text{ copies}} \tag{23}$$

or, equivalently, in scalar form:

$$h_{i_1,\cdots,i_N} = \sum_{r=1}^{R_S} \prod_{n=1}^{N} a_{i_n,r}, \qquad i_n = 1,\cdots,I \tag{24}$$

Unlike (15)–(17), such symmetric decomposition needs a unique matrix factor $\mathbf{A} \in \mathcal{C}^{I \times R_S}$. This decomposition will be called a symmetric PARAFAC model.

The symmetric rank of $\mathbb{H}$, noted $\text{rank}_S\,(\mathbb{H})$, is defined as the minimum number $R_S$ of symmetric outer products needed to generate $\mathbb{H}$ by means of (23) or (24).

So, any symmetric tensor can always be decomposed as in (15) or (24) with the following inequality between its rank and its symmetric rank [49]:

$$\text{rank}\,(\mathbb{H}) \leqslant \text{rank}_S\,(\mathbb{H}) \tag{25}$$

For an N$^{\text{th}}$-order rank-one symmetric tensor $\mathbb{H} \in \mathcal{C}^{I \times \cdots \times I}$, its symmetric PARAFAC model can be written in terms of a unique vector factor $\mathbf{a} \in \mathcal{C}^{I \times 1}$:

$$\mathbb{H} = \underbrace{\mathbf{a} \circ \cdots \circ \mathbf{a}}_{N \text{ copies}} \tag{26}$$

Under certain conditions, initially established by Harshman [50] and Kruskal [48, 51] for third-order real-valued tensors, then extended to complex-valued tensors in [52] and to $N^{th}$-order tensors in [53], the PARAFAC model is essentially unique, that is, its matrix factors are unique up to column permutation and scaling.

### *3.3. Estimation of PARAFAC parameters*

Identifying the PARAFAC model of a tensor $\mathbb{H}$ consists in estimating its matrix factors $\mathbf{A}^{(n)}$ from its matrix representations $\mathbf{H}_n$. This parameter estimation problem is generally solved in applying the alternating least squares (ALS) algorithm, originally proposed in [41] and [50].

The ALS algorithm, deduced from Equations (19)–(21), is summarized in Table I for a third-order PARAFAC model, where $\tilde{\mathbf{H}}_n$, $n = 1, 2, 3$, is the noisy version of $\mathbf{H}_n$ ($\tilde{\mathbf{H}}_n = \mathbf{H}_n + \mathbf{E}_n$, where $\mathbf{E}_n$



DOI: 10.1002/acs

Table I. Alternating least squares algorithm for a third-order PARAFAC model.

1. Randomly initialize $\mathbf{B}_0$ and $\mathbf{C}_0$ and set $t = 0$.
2. Increment $t$ and compute:
   (a) $\mathbf{A}_t = \tilde{\mathbf{H}}_1 \left[ (\mathbf{B}_{t-1} \odot \mathbf{C}_{t-1})^T \right]^\dagger$
   (b) $\mathbf{B}_t = \tilde{\mathbf{H}}_2 \left[ (\mathbf{C}_{t-1} \odot \mathbf{A}_t)^T \right]^\dagger$
   (c) $\mathbf{C}_t = \tilde{\mathbf{H}}_3 \left[ (\mathbf{A}_t \odot \mathbf{B}_t)^T \right]^\dagger$
3. Return to step (2) until convergence.

Table II. Conditional least squares algorithm for a symmetric third-order PARAFAC model.

1. Randomly initialize $\mathbf{A}_0$ and set $t = 0$.
2. Increment $t$ and compute:
$$\mathbf{A}_t = \tilde{\mathbf{H}}_1 \left[ (\mathbf{A}_{t-1} \odot \mathbf{A}_{t-1})^T \right]^\dagger$$
3. Return to step (2) until convergence.

is the horizontal mode-$n$ matrix unfolding of the additive noise tensor $\mathbb{E}$ having the same dimensions as $\mathbb{H}$).

In the case of a symmetric third-order tensor $\mathbb{H} \in \mathcal{C}^{I \times I \times I}$, Equations (19)–(21) become:

$$\mathbf{H}_1 = \mathbf{H}_2 = \mathbf{H}_3 = \mathbf{A} \, (\mathbf{A} \odot \mathbf{A})^T \in \mathcal{C}^{I \times I^2} \tag{27}$$

and the matrix factor $\mathbf{A}$ can be estimated at iteration $t$ in minimizing the following conditional least squares (CLS) cost function:

$$\min_{\mathbf{A}} \left\| \tilde{\mathbf{H}}_1 - \mathbf{A} \, (\mathbf{A}_{t-1} \odot \mathbf{A}_{t-1})^T \right\|_F^2 \tag{28}$$

Assuming that the matrix factor is full column rank, we get the CLS algorithm summarized in Table II [54].

The convergence test consists in detecting if an estimated parameter variation between two consecutive iterations or the model fit error calculated in using the tensor reconstructed with the estimated parameters, becomes smaller than a predefined threshold. In practice, to improve the convergence, the CLS algorithm that enforces the symmetry to the solution is applied after a transient period during which the classical ALS described in Table I is used for estimating the three matrix factors without enforcing the symmetry. More efficient algorithms like the enhanced line search (ELS) [55] or the Levenberg–Marquardt one [56] can also be used.

## 4. VOLTERRA-PARAFAC MODELS

A $P^{th}$-order Volterra model for a causal, stable, finite memory, time-invariant, SISO system is described by the following input–output relationship:

$$\begin{aligned} y_k &= h_0 + \sum_{p=1}^{P} \sum_{m_1=1}^{M_p} \cdots \sum_{m_p=1}^{M_p} h^{(p)}_{m_1,\cdots,m_p} \prod_{i=1}^{p} u_{k-m_i} \\ &= h_0 + \sum_{p=1}^{P} y_k^{(p)} \end{aligned} \tag{29}$$

where $h_0$ is the offset, $u_k$ and $y_k$ denote respectively the input and output signals, $P$ is the nonlinearity degree of the Volterra model, $M_p$ is the memory of the $p^{th}$-order homogeneous term $y_k^{(p)}$,



DOI: 10.1002/acs

and $h^{(p)}_{m_1,\cdots,m_p}$ is a coefficient of the $p^{th}$-order kernel. This coefficient being characterized by $p$ indices, it can be viewed as an element of a tensor $\mathbb{H}^{(p)} \in \mathcal{K}^{M_p\times\cdots\times M_p}$, of order $p$, with $\mathcal{K} = \mathcal{R}$ or $\mathcal{C}$, depending on whether the kernel coefficients are real-valued or complex-valued. The $p^{th}$-order kernel is characterized by $M_p^p$ coefficients. As each permutation of the indices $m_1,\cdots,m_p$ corresponds to the same product $\prod_{i=1}^{p} u_{k-m_i}$ of delayed inputs, we can sum all the coefficients associated with these permutations to get a symmetric kernel calculated as:

$$h^{(p,sym)}_{m_1,\cdots,m_p} = \frac{1}{p\,!}\sum_{\pi(.)} h^{(p)}_{m_{\pi(1)},\cdots,m_{\pi(p)}} \tag{30}$$

where $(\pi(1),\cdots,\pi(p))$ denotes a permutation of $(1,\cdots,p)$. So, in the sequel, without loss of generality, the Volterra kernels of order $p \geqslant 2$ will be considered in symmetric form. The number of independent coefficients contained in the symmetric $p^{th}$-order kernel is equal to $C_p^{M_p+p-1} = \frac{(M_p+p-1)!}{p!(M_p-1)!}$ (see [18]).

The $p^{th}$-order kernel is said to be separable if it can be expressed as the product of $p$ first-order kernels:

$$h^{(p)}_{m_1,\cdots,m_p} = \Pi_{i=1}^{p} h^{(i)}_{m_i} \tag{31}$$

where $h^{(i)}_{m_i}$ represents the $m_i^{th}$ element of the first-order kernel $\mathbf{h}^{(i)}$. If the separable kernel is symmetric, then the $p$ first-order kernels $\mathbf{h}^{(i)}$ are identical and (31) becomes:

$$h^{(p)}_{m_1,\cdots,m_p} = \Pi_{i=1}^{p} h_{m_i} \tag{32}$$

By comparing (31) with (22), we can conclude that a separable kernel is a rank-one tensor.

From the above discussion, any $p^{th}$-order Volterra kernel can be considered as a symmetric $p^{th}$-order tensor, which leads to the following corollary of Theorem 1:

*Corollary 1*

Any $p^{th}$-order Volterra kernel $h^{(p)}_{m_1,\cdots,m_p}$, with $p \geqslant 2$, can be decomposed using the symmetric PARAFAC model (24), with symmetric rank $r_p$ and matrix factor $\mathbf{A}^{(p)} \in \mathcal{C}^{M_p\times r_p}$:

$$h^{(p)}_{m_1,\cdots,m_p} = \sum_{r=1}^{r_p}\prod_{i=1}^{p} a^{(p)}_{m_i,r}, \quad m_i = 1,\cdots,M_p \quad i = 1,\cdots,p \tag{33}$$

That allows us rewriting the $p^{th}$-order homogeneous term $y_k^{(p)}$ as follows:

$$\begin{aligned} y_k^{(p)} &= \sum_{m_1=1}^{M_p}\cdots\sum_{m_p=1}^{M_p} h^{(p)}_{m_1,\cdots,m_p}\prod_{i=1}^{p} u_{k-m_i} \\ &= \sum_{m_1=1}^{M_p}\cdots\sum_{m_p=1}^{M_p}\left(\sum_{r=1}^{r_p}\prod_{i=1}^{p} a^{(p)}_{m_i,r}\right)\prod_{i=1}^{p} u_{k-m_i} \end{aligned} \tag{34}$$

Re-arranging the order of summation gives:

$$y_k^{(p)} = \sum_{r=1}^{r_p}\prod_{i=1}^{p}\sum_{m_i=1}^{M_p} a^{(p)}_{m_i,r} u_{k-m_i}$$

Defining the linear input regression vector $\mathbf{u}_k^{(p)} = \left[u_{k-1},\ \cdots,u_{k-M_p}\right]^T$, this expression becomes:

$$y_k^{(p)} = \sum_{r=1}^{r_p}\left(\mathbf{u}_k^{(p)^T}\mathbf{A}^{(p)}_{.r}\right)^p \tag{35}$$



DOI: 10.1002/acs

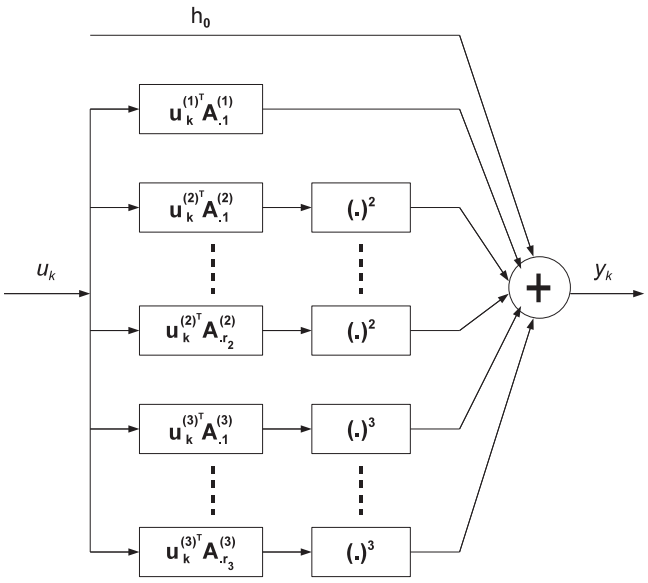


Figure 1. Realization of a cubic Volterra-PARAFAC model as Wiener models in parallel.

The $p^{th}$-order homogeneous term can therefore be carried out in parallelizing $r_p$ Wiener models, each one being associated with a column $\mathbf{A}_{.r}^{(p)} \in \mathcal{C}^{M_p \times 1}$ of the matrix factor $\mathbf{A}^{(p)}$ of the $p^{th}$-order kernel PARAFAC decomposition. Consequently, the Volterra model output (29) is obtained as the sum of the offset term $h_0$, and the outputs of $\sum_{p=1}^{P} r_p$ Wiener models in parallel, as illustrated in Figure 1 for a cubic Volterra-PARAFAC model, where $\mathbf{A}_{.1}^{(1)} = \left[h_1^{(1)}, \cdots, h_{M_1}^{(1)}\right]^T$ and $r_1 = 1$.

It is interesting to notice that such a Volterra-PARAFAC model provides a very attractive modular and parallel architecture for approximating nonlinear systems with a low computational complexity. This parallel Volterra-PARAFAC model is to be compared with the parallel cascade model constituted of Wiener paths and described by the following input–output relationship:

$$y_k = \sum_{n=1}^{N} \mathcal{N}^{(n)} \left( \sum_{m=1}^{M} h_m^{(n)} u_{k-m} \right) \tag{36}$$

where $\mathbf{h}^{(n)}$ and $\mathcal{N}^{(n)}(.)$ are respectively the FIR linear block and the static nonlinearity of the $n^{th}$ path, $n = 1, \cdots, N$. Korenberg showed that such a parallel cascade Wiener (PCW) model, with a finite number $N$ of paths, allows to approximate to an arbitrary accuracy any discrete-time, finite-memory nonlinear system [57]. The Volterra-PARAFAC model is a PCW model for which the FIR linear filters $\mathbf{h}^{(n)}$ are given by the columns of the matrix factors of the kernels PARAFAC decompositions, and the static nonlinearities $\mathcal{N}^{(n)}(.)$ are simple powers $(.)^n$. It is to be noticed that a method based on a joint diagonalization of third-order Volterra kernel slices has been recently proposed for identifying PCW systems [58].

## 5. VOLTERRA-GOB-TUCKER MODELS

Assuming that the $p^{th}$-order Volterra kernel $\mathbb{H}^{(p)}$ is absolutely summable on $[1, \infty)$, it can be expanded on GOB as:

$$h_{m_1,\cdots,m_p}^{(p)} = \sum_{r_1=1}^{\infty} \cdots \sum_{r_p=1}^{\infty} g_{r_1,\cdots,r_p}^{(p)} \prod_{n=1}^{p} \varphi_{r_n}^{(p,n)}(m_n) \tag{37}$$



DOI: 10.1002/acs

where $\varphi_r^{(p,n)}(.)$ is the $r^{th}$ GOBF used for expanding the $p^{th}$-order kernel along its $n^{th}$ mode, and $g^{(p)}_{r_1,\cdots,r_p}$ are the expansion coefficients, also called Fourier coefficients, that are calculated as:

$$g^{(p)}_{r_1,\cdots,r_p} = \sum_{m_1=1}^{\infty} \cdots \sum_{m_p=1}^{\infty} h^{(p)}_{m_1,\cdots,m_p} \prod_{n=1}^{p} \varphi^{(p,n)}_{r_n}(m_n) \tag{38}$$

The GOBF $\varphi_r^{(p,n)}(.)$, in the time domain, is given by the inverse $z$-transform of the following transfer function [27]:

$$\begin{aligned} \Phi_r^{(p,n)}(z) &= Z\left[\varphi_r^{(p,n)}(m)\right] \\ &= \frac{\sqrt{1-\left|\zeta_r^{(p,n)}\right|^2}}{z-\zeta_r^{(p,n)}} \prod_{i=1}^{r-1} \frac{1-\zeta_i^{(p,n)*} z}{z-\zeta_i^{(p,n)}},\ r=2,3,\cdots \end{aligned}$$

and

$$\Phi_1^{(p,n)}(z) = Z\left[\varphi_1^{(p,n)}(m)\right] = \frac{\sqrt{1-\left|\zeta_1^{(p,n)}\right|^2}}{z-\zeta_1^{(p,n)}}$$

The GOBs satisfy the following orthonormality property:

$$\left\langle \varphi_q^{(p,n)}, \varphi_r^{(p,n)} \right\rangle = \sum_{m=0}^{\infty} \varphi_q^{(p,n)}(m) \varphi_r^{(p,n)}(m) = \delta_{q,r}$$

where $\delta_{q,r}$ is the Kronecker delta.

Comparing (37) with (10), we conclude that the series expansion (37) of the kernel $\mathbb{H}^{(p)}$ can be interpreted as an infinite-dimensional Tucker model, with the core tensor $\mathbb{G}^{(p)} \in \mathcal{C}^{\infty\times\cdots\times\infty}$ containing the Fourier coefficients, and with matrix factors $\mathbf{A}^{(p,n)}$ whose coefficients are $a^{(p,n)}_{m_n,r_n} = \varphi^{(p,n)}_{r_n}(m_n)$ and the columns are orthonormal.

In practice, a truncated expansion is used, which amounts to consider only a finite number of GOBF:

$$h^{(p)}_{m_1,\cdots,m_p} = \sum_{r_1=1}^{R_p} \cdots \sum_{r_p=1}^{R_p} g^{(p)}_{r_1,\cdots,r_p} \prod_{n=1}^{p} \varphi^{(p,n)}_{r_n}(m_n) \tag{39}$$

This truncated series expansion is equivalent to a Tucker model with a core tensor $\mathbb{G}^{(p)} \in \mathcal{C}^{R_p\times\cdots\times R_p}$, $R_p$ being the truncation order, and the $r^{th}$ column of the matrix factor $\mathbf{A}^{(p,n)} \in \mathcal{C}^{M_p\times R_p}$, is constituted with the first $M_p$ points of the impulse response of the filter having $\Phi_r^{(p,n)}(z)$ as transfer function. It is to be noticed that the truncation order $R_p$ is strongly dependent on the choice of the poles $\zeta_r^{(p,n)}$, $r=1,\cdots,R_p$, of the GOBF $\varphi_r^{(p,n)}$, $n=1,\cdots,p$.

Substituting (39) into (34) gives the following input–output relationship for the $p^{th}$-order homogeneous Volterra-GOB-model that we shall call Volterra-GOB-Tucker model:

$$\begin{aligned} y_k^{(p)} &= \sum_{r_1=1}^{R_p} \cdots \sum_{r_p=1}^{R_p} g^{(p)}_{r_1,\cdots,r_p} \prod_{n=1}^{p} \left( \sum_{m_n=1}^{M_p} \varphi^{(p,n)}_{r_n}(m_n) u_{k-m_n} \right) \\ &= \sum_{r_1=1}^{R_p} \cdots \sum_{r_p=1}^{R_p} g^{(p)}_{r_1,\cdots,r_p} \prod_{n=1}^{p} \Psi^{(p,n)}_{r_n}(k) \end{aligned} \tag{40}$$



DOI: 10.1002/acs

where:

$$\Psi_{r_n}^{(p,n)}(k) = \sum_{m_n=1}^{M_p} \varphi_{r_n}^{(p,n)}(m_n) u_{k-m_n}$$

is the response of the GOBF $\varphi_{r_n}^{(p,n)}$, viewed as an FIR linear filter with memory $M_p$, to the input signal $u_k$.

A triangular form of the Volterra-GOB-Tucker model (40) is characterized by $C_p^{R_p+p-1}$ free parameters. This model is to be compared with the Volterra-PARAFAC model (35). When the GOB poles are fixed, the Volterra-GOB-Tucker model is linear in its parameters, the Fourier coefficients, that can be estimated by means of the standard least squares (LS) algorithm, whereas the Volterra-PARAFAC model is nonlinear in its parameters, the PARAFAC coefficients, that can be estimated using a nonlinear optimization method like a gradient-based algorithm, or an extended Kalman filter, as described in the next section. However, the selection of the GOB poles is a difficult task, the result of which significantly influences the choice of the truncation order $R_p$, that is, the parametric complexity of the model. For this reason, the Volterra-PARAFAC model appears more attractive, the choice of the kernels ranks $r_p$ being easy to make as it will be illustrated by means of simulation results in Section 7.

## 6. ADAPTIVE PARAMETER ESTIMATION METHODS FOR VOLTERRA-PARAFAC MODELS

Let us assume that the output of the system to be identified is modeled by means of the following Volterra-PARAFAC model deduced from Equations (29) and (35):

$$y_k = h_0 + \sum_{p=1}^{P} \sum_{r=1}^{r_p} \left( \mathbf{u}_k^{(p)^T} \mathbf{A}_{.r}^{(p)} \right)^p \tag{41}$$

where $\mathbf{u}_k^{(p)} = \left[u_{k-1}, \cdots, u_{k-M_p}\right]^T$, $\mathbf{A}^{(p)}$ is the matrix factor of the PARAFAC decomposition of the $p^{th}$-order kernel, for $p \geq 2$, and $r_1 = 1$.

In the sequel, we assume that the Volterra kernels are real valued, with known symmetric rank $r_p$ and memory $M_p$, and we consider complex valued input–output signals to be more general. Indeed, signals take complex values in various fields of application like array processing or mobile communications for which the symbols to transmit are most often PSK (phase shift keying) or QAM (quadrature amplitude modulation) modulated. Three adaptive algorithms are proposed for estimating the parameters of the $P^{th}$-order Volterra-PARAFAC model (41). We first present the ECKF algorithm. Then, we develop the adaptive steepest descent algorithm, also called stochastic gradient algorithm or CLMS algorithm, and finally, we derive its normalized version, the so-called NCLMS algorithm.

### *6.1. The extended complex Kalman filter algorithm*

Defining the parameter vector:

$$\boldsymbol{\theta}^T = \left[ h_0 \, \boldsymbol{\theta}^{(1)^T} \, \cdots \boldsymbol{\theta}^{(P)^T} \right] \tag{42}$$

with

$$\boldsymbol{\theta}^{(1)} = \mathbf{A}_{.1}^{(1)} = \left[ h_1^{(1)}, \cdots, h_{M_1}^{(1)} \right]^T \in \mathcal{C}^{M_1 \times 1} \tag{43}$$

$$\boldsymbol{\theta}^{(p)} = \mathrm{vec}\left( \mathbf{A}^{(p)} \right) \in \mathcal{C}^{M_p r_p \times 1}, \quad \text{for } p = 2, \cdots, P, \tag{44}$$

The input–output relation (41) can be rewritten as:

$$y_k = f(k, \boldsymbol{\theta}) \tag{45}$$



DOI: 10.1002/acs

The idea behind the ECKF algorithm is to apply the Kalman filter at time $k$, after a linearization of $f(k,\boldsymbol{\theta})$ around the last estimate $\hat{\boldsymbol{\theta}}_{k-1}$ calculated at time $k-1$:

$$y_k \approx f(k,\hat{\boldsymbol{\theta}}_{k-1}) + \hat{\mathbf{g}}_k^T \left(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{k-1}\right) \tag{46}$$

where

$$\hat{\mathbf{g}}_k = \mathbf{g}\left(k,\hat{\boldsymbol{\theta}}_{k-1}\right) = \left.\frac{\partial f(k,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}}$$

The gradient of the nonlinear function $f(k,\boldsymbol{\theta})$ with respect to the parameter vector $\boldsymbol{\theta}$ can be calculated analytically by means of the following formulae:

$$\hat{\mathbf{g}}_k = \left[\hat{\varphi}_k^{(0)} \;\; \hat{\boldsymbol{\varphi}}_k^{(1)^T} \;\; \hat{\boldsymbol{\varphi}}_k^{(2)^T} \;\; \cdots \;\; \hat{\boldsymbol{\varphi}}_k^{(P)^T}\right]^T \tag{47}$$

with

$$\hat{\varphi}_k^{(0)} = 1, \qquad \hat{\boldsymbol{\varphi}}_k^{(1)} = \mathbf{u}_k^{(1)} \tag{48}$$

and for $p = 2,\cdots,P$:

$$\hat{\boldsymbol{\varphi}}_k^{(p)} = \left.\frac{\partial f(k,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(p)}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}} = p\hat{\mathbf{v}}_k^{(p)} \otimes \mathbf{u}_k^{(p)} \tag{49}$$

with

$$\hat{\mathbf{v}}_k^{(p)} = \left[\;\left(\hat{\alpha}_k^{(p,1)}\right)^{p-1}, \;\; \cdots \;\; , \left(\hat{\alpha}_k^{(p,r_p)}\right)^{p-1}\;\right]^T \tag{50}$$

and

$$\hat{\alpha}_k^{(p,r)} = \mathbf{u}_k^{(p)^T} \hat{\mathbf{A}}_{.r}^{(p)}(k-1), \text{for } r = 1,\cdots,r_p \tag{51}$$

$\hat{\mathbf{A}}_{.r}^{(p)}(k-1)$ being extracted from the estimated parameter vector $\hat{\boldsymbol{\theta}}_{k-1}$.

Noting that we have:

$$f\left(k,\hat{\boldsymbol{\theta}}_{k-1}\right) = \hat{\boldsymbol{\psi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \tag{52}$$

with

$$\hat{\boldsymbol{\psi}}_k^T = \left[\hat{\varphi}_k^{(0)} \;\; \hat{\boldsymbol{\varphi}}_k^{(1)^T} \;\; \frac{1}{2}\hat{\boldsymbol{\varphi}}_k^{(2)^T} \;\; \cdots \;\; \frac{1}{P}\hat{\boldsymbol{\varphi}}_k^{(P)^T}\right] \tag{53}$$

the innovation process associated with the linearized model equation is given by:

$$\hat{e}_k = s_k - \left[f\left(k,\hat{\boldsymbol{\theta}}_{k-1}\right) + \hat{\mathbf{g}}_k^T \left(\hat{\boldsymbol{\theta}}_{k-1} - \hat{\boldsymbol{\theta}}_{k-1}\right)\right] \tag{54}$$

$$= s_k - \hat{\boldsymbol{\psi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \tag{55}$$

where $s_k$ is the measured output of the system to be identified.

The application of the complex Kalman filter leads to the ECKF algorithm summarized in Table III, where $\lambda$ is a forgetting factor.

*Remarks*

(i) When the PARAFAC parameters are slowly time-varying, we can assume that they satisfy a random walk model defined as:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \mathbf{w}_k \tag{61}$$



DOI: 10.1002/acs

Table III. The extended complex Kalman filter algorithm.

1. Randomly initialize $\hat{\boldsymbol{\theta}}_0$, set $P_0 = \alpha\mathbf{I}$, where $\alpha$ is a scalar, and $k = 0$.
2. Increment $k$ and compute $\hat{\mathbf{g}}_k$ and $\hat{\boldsymbol{\psi}}_k$ using (47)–(51) and (53), respectively.

$$\hat{e}_k = s_k - \hat{\boldsymbol{\psi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \tag{56}$$
$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{G}_k \hat{e}_k \tag{57}$$
$$\Sigma_k = \hat{\mathbf{g}}_k^T \mathbf{P}_{k-1} \hat{\mathbf{g}}_k^* + \sigma_e^2 \tag{58}$$
$$\mathbf{G}_k = \mathbf{P}_{k-1} \hat{\mathbf{g}}_k^* \Sigma_k^{-1} \tag{59}$$
$$\mathbf{P}_k = \lambda^{-1} \left( \mathbf{P}_{k-1} - \mathbf{G}_k \Sigma_k \mathbf{G}_k^H \right) \tag{60}$$

3. Return to Step 2 until $k = K$, where $K$ is the number of input–output data to be processed.

where $\{\mathbf{w}_k\}$ is a white noise sequence with covariance $\sigma_{\mathbf{w}}^2\mathbf{I}$. This model is also called a discrete-time Brownian motion. With this state-space model for the parameter variations, the only modification to make in the ECKF algorithm concerns Equation (60) that becomes:

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{G}_k \Sigma_k \mathbf{G}_k^H + \sigma_{\mathbf{w}}^2 \mathbf{I} \tag{62}$$

(ii) The ECKF algorithm described in Table III can be viewed as an extension of the standard recursive least squares algorithm to the case of the nonlinear measurement equation (45), extension that results from the linearization (46) of this equation. As well known, the extended Kalman filter exhibits good performance in terms of speed of convergence. The main drawback of this algorithm is its computation burden because of the Riccati equation defined by Equations (58)–(60). A simpler adaptive method is obtained in applying the steepest-descent algorithm described in the next section that leads to the CLMS algorithm.

### *6.2. The complex least mean square algorithm*

Let us consider the CLMS algorithm that minimizes the following instantaneous cost function:

$$J(k) = \frac{1}{2}|e_k|^2 = \frac{1}{2}\left(e_{k,R}^2 + e_{k,I}^2\right) \text{ with } e_k = e_{k,R} + j e_{k,I}$$

where $e_{k,R} = s_{k,R} - f_R(k, \boldsymbol{\theta})$ and $e_{k,I} = s_{k,I} - f_I(k, \boldsymbol{\theta})$ are respectively the real and imaginary parts of the complex valued output error $e_k$, $f(k, \boldsymbol{\theta}) = f_R(k, \boldsymbol{\theta}) + j f_I(k, \boldsymbol{\theta})$, and $j^2 = -1$.

The derivation of the cost function $J(k)$ with respect to the parameter vector $\boldsymbol{\theta}^{(p)}$ gives for $p = 0, 1, \cdots, P$:

$$\frac{\partial J(k)}{\partial \boldsymbol{\theta}^{(p)}} = -e_{k,R} \frac{\partial f_R(k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(p)}} - e_{k,I} \frac{\partial f_I(k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(p)}} \tag{63}$$

Applying the steepest-descent algorithm gives the following update equations for the estimated parameters:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_k^{(p)} &= \hat{\boldsymbol{\theta}}_{k-1}^{(p)} - \frac{\mu_p}{2} \frac{\partial \left|e_k^2\right|}{\partial \boldsymbol{\theta}^{(p)}}\bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}} \\ &= \hat{\boldsymbol{\theta}}_{k-1}^{(p)} + \mu_p \left[ e_{k,R} \frac{\partial f_R(k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(p)}} + e_{k,I} \frac{\partial f_I(k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(p)}} \right]\bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}} \end{aligned}$$

where $\mu_p$ is a small, positive, step size that controls the convergence speed and the steady-state properties of the algorithm.

Using the definition (49) of the gradient, we obtain:

$$\hat{\boldsymbol{\theta}}_k^{(p)} = \hat{\boldsymbol{\theta}}_{k-1}^{(p)} + \mu_p \mathcal{R}\left(\hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k^*\right) \tag{64}$$



DOI: 10.1002/acs

Table IV. The complex least mean square algorithm.

1. Randomly initialize $\hat{\boldsymbol{\theta}}_0$ and set $k = 0$.
2. Increment $k$ and compute $\hat{\boldsymbol{\varphi}}_k^{(p)}$ for $p = 0, 1, \cdots, P$, and $\hat{\boldsymbol{\psi}}_k$ using (48)–(51) and (53), respectively.

$$\hat{e}_k = s_k - \hat{\boldsymbol{\psi}}_k^T \hat{\boldsymbol{\theta}}_{k-1}$$

For $p = 0, 1, \cdots, P$:

$$\hat{\boldsymbol{\theta}}_k^{(p)} = \hat{\boldsymbol{\theta}}_{k-1}^{(p)} + \mu_p \mathcal{R}\left(\hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k^*\right) \tag{65}$$

3. Return to Step 2 until $k = K$, where $K$ is the number of input–output data to be processed.

where $\mathcal{R}\left(\hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k^*\right)$ denotes the real part of $\hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k^*$, with $\hat{\boldsymbol{\varphi}}_k^{(p)}$ defined in (48) for $p = 0, 1$, and in (49)–(51) for $p = 2, \cdots, P$, and $\hat{e}_k$ is calculated as in (56).

The equations of the CLMS algorithm are summarized in Table IV.

*Remarks*

When the input–output signals are real valued, the update equation (64) becomes:

$$\hat{\boldsymbol{\theta}}_k^{(p)} = \hat{\boldsymbol{\theta}}_{k-1}^{(p)} + \mu_p \hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k \tag{66}$$

### *6.3. The NCLMS algorithm*

From Equation (49), we have for $p = 2, \cdots, P$:

$$\left\| \hat{\boldsymbol{\varphi}}_k^{(p)} \right\|_2^2 = \hat{\boldsymbol{\varphi}}_k^{(p)H} \hat{\boldsymbol{\varphi}}_k^{(p)} = p^2 \left\| \hat{\mathbf{v}}_k^{(p)} \right\|_2^2 \left\| \mathbf{u}_k^{(p)} \right\|_2^2 \tag{67}$$

The estimated parameter update equations of the NCLMS algorithm are then directly deduced from Equation (64) as:

$$\hat{\boldsymbol{\theta}}_k^{(1)} = \hat{\boldsymbol{\theta}}_{k-1}^{(1)} + \frac{\nu_1}{c_1 + \left\| \mathbf{u}_k^{(1)} \right\|_2^2} \mathcal{R}\left(\mathbf{u}_k^{(1)} \hat{e}_k^*\right) \tag{68}$$

and

$$\hat{\boldsymbol{\theta}}_k^{(p)} = \hat{\boldsymbol{\theta}}_{k-1}^{(p)} + \frac{\nu_p}{c_p + p^2 \left\| \hat{\mathbf{v}}_k^{(p)} \right\|_2^2 \left\| \mathbf{u}_k^{(p)} \right\|_2^2} \mathcal{R}\left(\hat{\boldsymbol{\varphi}}_k^{(p)} \hat{e}_k^*\right) \text{ for } p = 2, \cdots, P, \tag{69}$$

where $c_p$ is a small positive constant introduced for avoiding numerical problems when $\left\| \hat{\boldsymbol{\varphi}}_k^{(p)} \right\|_2$ becomes close to zero as it is the case for a low-value input sequence.

It is important to notice that for the CLMS and NCLMS algorithms, the computation of the estimated PARAFAC parameters of each kernel can be parallelized.

## 7. SIMULATION RESULTS

In this section, we present some simulation results to illustrate the effectiveness of PARAFAC for approximating Volterra models. Two aspects are evaluated: the parametric complexity reduction and the accuracy of the Volterra-PARAFAC model. To draw conclusions not too dependent on the simulated Volterra models and observation conditions, the performance were evaluated by averaging the results over different models and noise sequences.

First, in subsection 7.1, given a noisy third order Volterra kernel, we evaluate the complexity reduction rate and the accuracy of the approximated model obtained using PARAFAC.

Secondly, in subsection 7.2, given noisy input–output measurements, we evaluate the adaptive parameter estimation schemes derived in Section 6.



DOI: 10.1002/acs

In both subsections 7.1 and 7.2, the simulated Volterra kernels exhibit a PARAFAC algebraic structure.

Finally, in subsection 7.3, we compare the Volterra-PARAFAC and Volterra-GOB-Tucker models with the standard Volterra model for representing a simulated polymerization reactor.

### *7.1. Approximation of Volterra kernels using PARAFAC*

Let us consider $M = 100$ different third-order homogeneous Volterra models, with memory $M_3 = 10$. The entries of the $m^{th}$ kernel $\mathbb{H}_m^{(3)}$, of dimensions $10 \times 10 \times 10$, are generated as $h_{m_1,m_2,m_3}^{(m)} = \sum_{r=1}^{r_3} \prod_{i=1}^{3} a_{m_i,r}^{(m)}$, $m_i = 1, \cdots, 10$, $m = 1, \cdots, 100$, the parameters $a_{m_i,r}^{(m)}$ being randomly drawn from a Gaussian distribution with zero-mean and unit variance and the rank $r_3 \in \{3, 4\}$. Given noisy kernels $\mathbb{H}_{m,l}^{(3)} = \mathbb{H}_m^{(3)} + \mathbb{N}_l$, $l = 1, \cdots, L$, $L = 10$, we evaluate the effectiveness of PARAFAC for approximating the Volterra model by means of the CLS and ALS algorithms. Herein, the entries of $\mathbb{N}_l$ were randomly and independently drawn from a Gaussian distribution with zero-mean and a variance adjusted in such a way that the signal to noise ratio (SNR) $= 10 \log_{10} \left( \left\| \mathbb{H}_m^{(3)} \right\|_F^2 / \|\mathbb{N}_l\|_F^2 \right)$ be set to a given value. Let us denote by $\hat{\mathbb{H}}_{m,l}^{(3)}$ the reconstructed Volterra kernel for the $m^{\text{th}}$ model and the $l^{\text{th}}$ noise sequence. The accuracy of the PARAFAC approximation is evaluated by means of the kernel normalized mean square error (NMSE) calculated in decibel as an average over the $M_c = \sum_{m=1}^{M} L_m$ simulations that converged at the most in 150 iterations, $L_m \leqslant L$ being the number of runs that converged for the $m^{\text{th}}$ model:

$$\text{K-NMSE} = 10 \log_{10} \left( \frac{1}{M_c} \sum_{m=1}^{M} \sum_{l_m=1}^{L_m} \frac{\left\| \mathbb{H}_m^{(3)} - \hat{\mathbb{H}}_{m,l_m}^{(3)} \right\|_F^2}{\left\| \mathbb{H}_m^{(3)} \right\|_F^2} \right) \tag{70}$$

The complexity reduction rate (CRR) is computed in % as:

$$\text{CRR} = 100 \frac{N_{VS} - N_{VP}}{N_{VS}} \tag{71}$$

where $N_{VS}$ and $N_{VP}$ that represent the parameter numbers in a symmetrized Volterra kernel and in its symmetric rank-$R$ PARAFAC approximation, respectively, are given by:

$$N_{VS} = C_p^{M_p+p-1}, \qquad N_{VP} = M_p R$$

For instance, for a third-order Volterra kernel ($p = 3$), with memory $M_3 = 10$, we have $N_{VS} = 220$, $N_{VP} = 30$ for $R = 3$, which gives CRR $= 86.4$ %. In Figure 2, the CRR is plotted for $R \in \{1, \cdots, 10\}$, when $M_3 = 10$.

To illustrate the notion of rank of a Volterra kernel, we consider the case of a PCW system modeled by means of Equation (36), with $N$ paths. In [58], it was shown that the kernels of the Volterra model associated with such a PCW system admit a natural symmetric PARAFAC decomposition of rank $N$, whose matrix factor, of dimensions $M \times N$, contains the parameters of the linear subsystem of each path, where $M$ is the greatest memory of all the paths.

The performance of the Volterra-PARAFAC model is evaluated in terms of K-NMSE for different values of the approximation rank $R$. We observed that when the rank is overestimated, the CLS algorithm does not converge. In this case, we used the standard ALS algorithm described in Table I for estimating a nonsymmetric PARAFAC model. When the rank is underestimated and for the approximation rank equal to the true rank, we used a combination of ALS and CLS denoted ALS/CLS; that is, CLS is preceded by 60 ALS iterations.

For four different values of the approximation rank $R \in \{2, 3, 4, 5\}$, the K-NMSE versus SNR is plotted on Figures 3 and 4, for rank-3 and rank-4 Volterra kernels, respectively. From these simulation results, we can conclude that the approximation of a Volterra kernel using a symmetric



DOI: 10.1002/acs

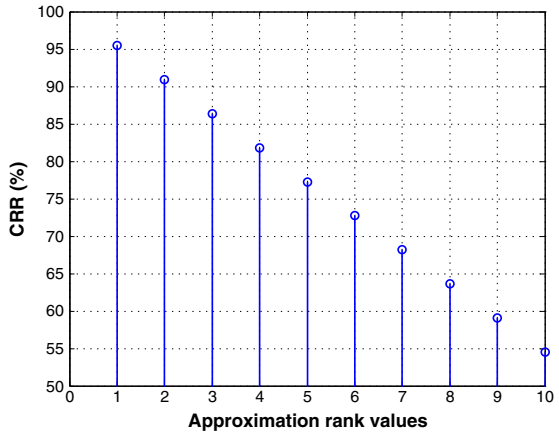


Figure 2. Complexity reduction rate for different approximation rank values when $M_3 = 10$.

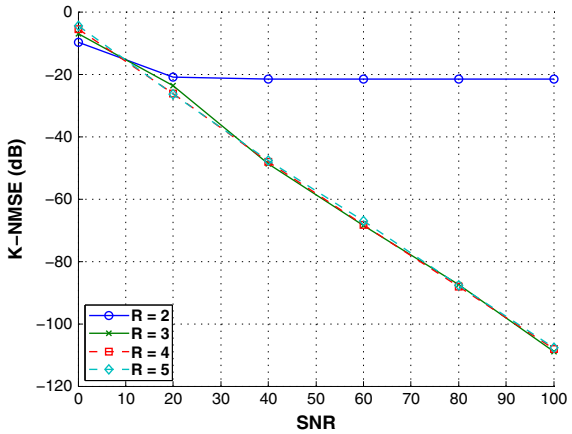


Figure 3. K-NMSE versus SNR obtained with the alternating least squares and alternating least squares/conditional least squares algorithms, for rank-3 kernels and different approximation ranks.

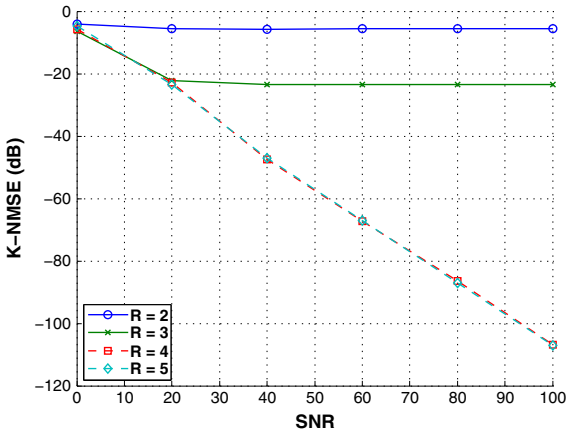


Figure 4. K-NMSE versus SNR obtained with the alternating least squares and alternating least squares/conditional least squares algorithms, for rank-4 kernels and different approximation ranks.

PARAFAC model can be carried out in applying combined ALS and CLS algorithms, in parallel, for different values of the approximation rank $R$. The final estimated value of $R$ can be chosen in such a way that a trade-off between parametric complexity and approximation accuracy be satisfied.



DOI: 10.1002/acs

From these results, we can conclude that:

(i) the Volterra-PARAFAC model allows a drastic complexity reduction that is all greater as the kernel rank is smaller (see Figure 2);
(ii) good fit of the PARAFAC model is reached when the approximation rank is greater than or equal to the true rank (see Figures 3 and 4);
(iii) as expected, for an approximation rank greater than or equal to the true rank, the K-NMSE decreases when the SNR increases (see Figures 3 and 4);

### *7.2. Adaptive parameter estimation using the ECKF, CLMS, and NCLMS algorithms*

In this subsection, we simulated the output of a third-order Volterra system, with a 16-PSK input signal, whose values were uniformly drawn from the alphabet $\{e^{j2\pi\frac{m}{16}} \mid m = 0, \cdots, 15\}$. We consider $M = 15$ Volterra models and $L = 10$ additive complex white Gaussian noises for each model ($m = 1, \cdots, 15$):

$$y_{k,m} = h_0^{(m)} + \sum_{p=1}^{3} \sum_{m_1,\cdots,m_p=1}^{M_p} h_{m_1,\cdots,m_p}^{(p,m)} \prod_{i=1}^{p} u_{k-m_i} = f\left(k, \boldsymbol{\theta}^{(m)}\right)$$

The Volterra kernels were generated as in the previous subsection.

We define the output NMSE at time $k$ as follows:

$$\text{O-NMSE}_k = 10\log_{10}\left(\frac{1}{M_c}\sum_{m=1}^{M}\sum_{l_m=1}^{L_m}\frac{\left\|\mathbf{y}_{k,m} - \hat{\mathbf{y}}_{k,m,l_m}\right\|_2^2}{\left\|\mathbf{y}_{k,m}\right\|_2^2}\right) \tag{72}$$

where $\mathbf{y}_{k,m} = \left[y_{k-\tau+1,m}, \cdots, y_{k,m}\right]^T$ denotes the output vector associated with the $m^{\text{th}}$ simulated model, whereas $\hat{\mathbf{y}}_{k,m,l_m}$ denotes the corresponding vector of reconstructed outputs $\hat{y}_{i,m,l_m} = f(i, \hat{\boldsymbol{\theta}}_{k,m,l_m})$, $i = k-\tau+1, \cdots, k$, of the estimated Volterra-PARAFAC model for the $l_m^{th}$ converged experiment. The length of the sliding window was set to $\tau = 2000$.

In this subsection, we assume that the convergence occured when the O-NMSE reaches its minimum value added to 5 dB. The simulation results given in the sequel were averaged over the experiments that converged within 50,000 iterations.

We evaluate the performance of the adaptive algorithms according to output SNR defined, in dB, as SNR=$10\log_{10}(\|\mathbf{y}_K\|_2^2 / \|\mathbf{b}_K\|_2^2)$, $\mathbf{y}_K$ and $\mathbf{b}_K$ containing, respectively, the noiseless output data and the additive noise.

The adaptation step sizes of the CLMS and NCLMS algorithms are given in Table V. For NCLMS, we set $c_p = 0.001$, $p = 1, 2, 3$.

Figures 5 and 6 show the cumulative density function of the number of iterations for convergence of the three adaptive algorithms, with SNR = 40 dB, for rank-3 and rank-4 kernels, respectively. Similar results were obtained for other SNR values.

Figures 7 and 8 show the O-NMSE versus the iterations for the three adaptive algorithms, SNR= 40 dB, in the cases of rank-3 and rank-4 kernels, respectively.

From these simulation results, we can conclude that ECKF converges more fastly than NCLMS that itself converges much more rapidly than CLMS.

Figures 9 and 10 show the O-NMSE versus SNR with the approximation ranks equal to the true ranks, for the three proposed adaptive estimation methods. As expected, the O-NMSE decreases

Table V. Adaptation step sizes.

| $p$ | 1 | 2 | 3 |
|---|---|---|---|
| $\mu_p$ | $1\cdot10^{-3}$ | $1.5\cdot10^{-4}$ | $8\cdot10^{-5}$ |
| $\nu_p$ | $5\cdot10^{-2}$ | $2\cdot10^{-1}$ | $2\cdot10^{-1}$ |



DOI: 10.1002/acs

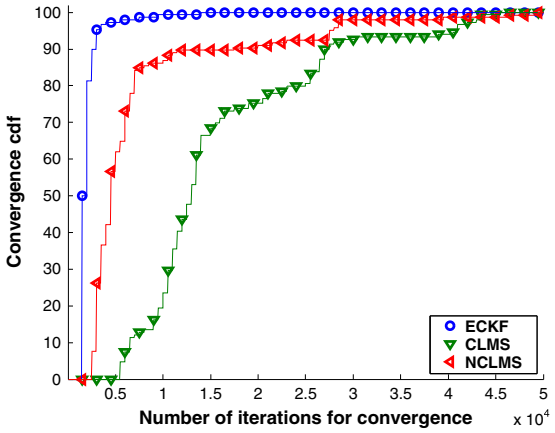


Figure 5. Cumulative density function for the convergence of the three methods, rank-3 kernels, SNR = 40 dB.

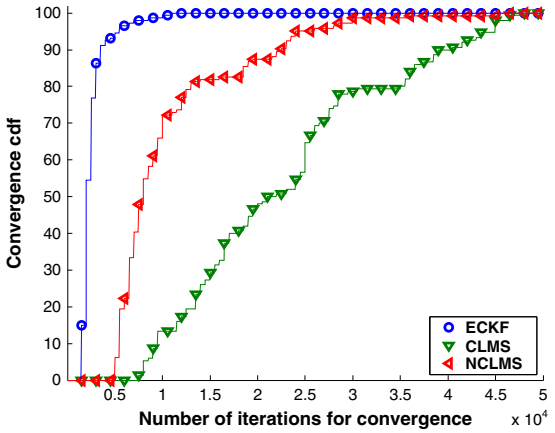


Figure 6. Cumulative density function for the convergence of the three methods, rank-4 kernels, SNR = 40 dB.

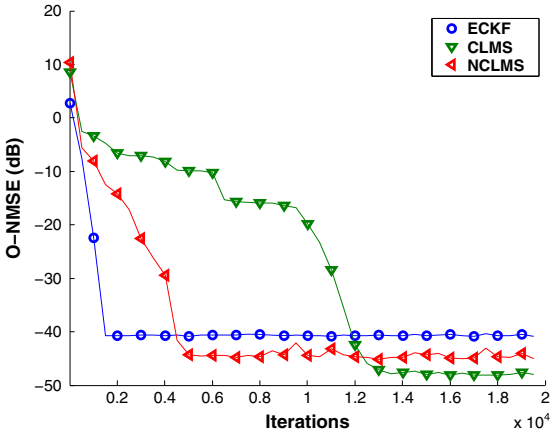


Figure 7. O-NMSE versus iterations for rank-3 kernels, SNR= 40 dB.

when the SNR increases, whatever the method and model we consider. Moreover, it can be observed that the CLMS algorithm gives more precise estimation than NCLMS and ECKF. For instance, in the case of a SNR = 40 dB and for rank-3 kernels, the O-NMSE obtained with



DOI: 10.1002/acs

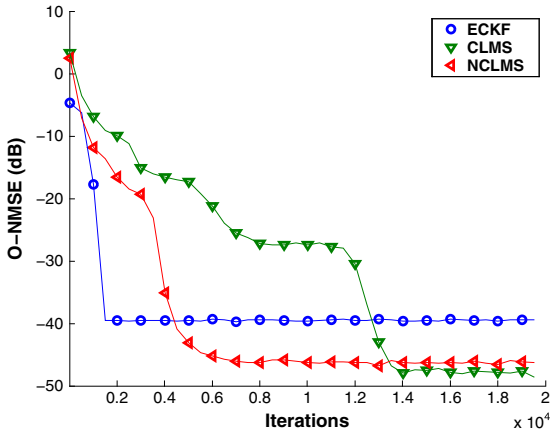


Figure 8. O-NMSE versus iterations for rank-4 kernels, SNR= 40 dB.

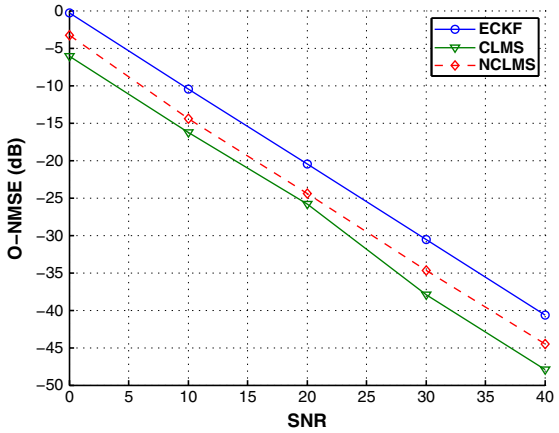


Figure 9. O-NMSE versus SNR for the three adaptive methods, rank-3 kernels.

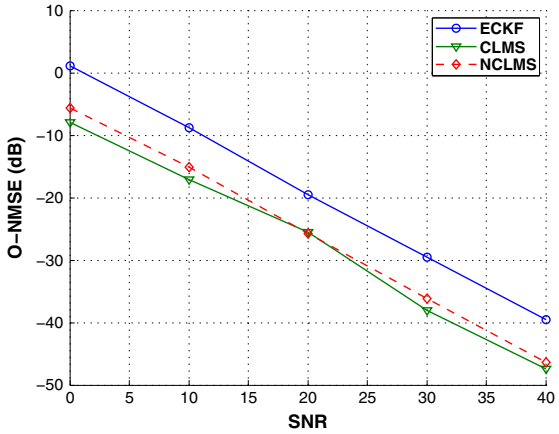


Figure 10. O-NMSE versus SNR for the three adaptive methods, rank-4 kernels.

CLMS is around −47.9 dB, whereas it is around −40.7 dB and −44.5 dB with ECKF and NCLMS, respectively.

We have to notice that the performances of the CLMS and NCLMS algorithms are strongly dependent on the choice of the step sizes.



DOI: 10.1002/acs

### 7.3. Simulated polymerization reactor

In this subsection, we consider the identification of a simulated polymerization reactor [59] whose state-space representation is given by:

$$\begin{aligned}
\dot{x}_1 &= 60 - 10x_1 - 2.45684x_1\sqrt{x_2} \\
\dot{x}_2 &= 80u - 10.1022x_2 \\
\dot{x}_3 &= 0.0024121x_1\sqrt{x_2} + 0.112184x_2 - 10x_3 \\
\dot{x}_4 &= 245.979x_1\sqrt{x_2} - 10x_4 \\
y &= \frac{x_4}{x_3}
\end{aligned}$$

This model is composed of coupled nonlinear differential equations. The reaction system involves the isothermal free radical polymerization of methyl-methacrylate using azo-bis-isobutyronitrile as an initiator and toluene as a solvent. The output $y$ is the number average molecular weight (NAMW) of the polymer and is a measure of the quality of the final product. The NAMW is controlled by manipulating the inlet initiator flow rate which serves as the input $u$. The state variables are the monomer concentration $x_1$, the initiator concentration $x_2$, and the zeroth and first order moments ($x_3$,$x_4$) of the polymer molecular weight distribution. For our simulations, we have made use of the equivalent discrete-time state-space representation given in [60].

The input signal was stepwise constant with a random magnitude uniformly distributed within $[-1, 1]$, kept constant during 10 consecutive samples, the sampling period being $T_e = 0.06$h. The length of the input sequence used for identification was $K = 15000$. For validation purpose, we made use of five input sequences with different length. The validation results shown below are averaged values obtained with these different input sequences (see Table VII). The adopted Volterra model was a third-order one.

In the sequel, we compare three models: Volterra, Volterra-GOB-Tucker and Volterra-PARAFAC, in terms of parametric complexity and O-NMSE evaluated as follows:

$$\text{O-NMSE} = 10\log_{10}\frac{\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2}{\|\mathbf{y}\|_2^2} \tag{73}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote respectively the output vector of the simulated polymerization reactor and that of the estimated model.

The parameters of Volterra and Volterra-GOB-Tucker models were estimated using an NLMS algorithm, whereas an extended Kalman filter was used for estimating those of the Volterra-PARAFAC model.

The selected GOBs, given in Table VI, were obtained after numerous trials, with a memory $M = 20$.

Unlike Volterra-Laguerre and Volterra-Kautz models [32]–[34], to the best of the authors' knowledge, there exists no method for computing optimal poles of Volterra-GOB models. Only suboptimal methods have been proposed in the literature (see for example [35]–[36]).

For the Volterra-PARAFAC model, Figure 11 depicts the O-NMSE versus the model memory, for three values of the approximation rank $R$ chosen identical for the quadratic and cubic kernels. From this figure, we can conclude that the best trade-off between parametric complexity and model precision is obtained with $R = 1$ and $M = 20$.

Table VI. Selected generalized orthonormal basis

| Kernel order | Truncation order | Poles |
|---|---|---|
| 1 | 5 | 0.53, 0.53, 0.53, 0.53, 0.53 |
| 2 | 3 | 0.5, 0.5, 0.5 |
| 3 | 5 | 0.5, 0.5, 0.5, −0.3, 0.5 |



DOI: 10.1002/acs

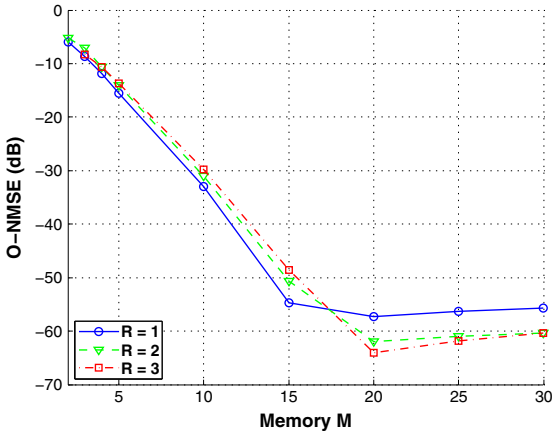


Figure 11. O-NMSE versus model memory for three values of the approximation rank.

In Figure 12, the performance of the three models is compared in terms of O-NMSE for different values of the memory $M$. For the Volterra model, by increasing $M$, the O-NMSE decreases for $M < 15$ and then increases. The minimal O-NMSE is obtained for $M = 15$ that corresponds to 816 parameters. The degradation of the performance for $M > 15$ is because of the increase of the number of parameters to be estimated. Therefore, more data are needed to reach convergence. On the other hand, we can note that increasing the memory $M$ allows improving the O-NMSE of the Volterra-GOB-Tucker model while keeping constant the number of parameters of the model. For the Volterra-PARAFAC model, increasing $M$ above $M = 15$ keeps the O-NMSE quasi constant.

In Table VII, we compare the best configurations of the three considered models in terms of parametric complexity and O-NMSE for both identification and validation phases. From these

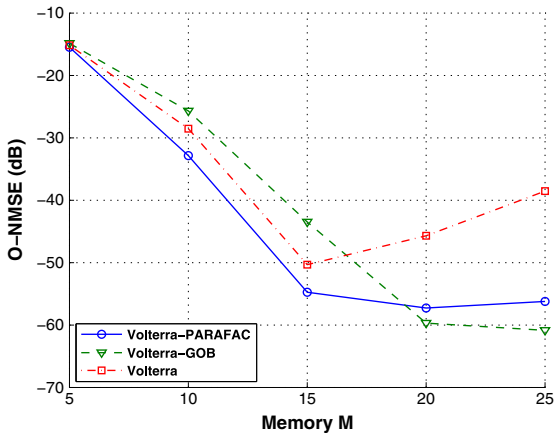


Figure 12. O-NMSE versus model memory for the different models: Volterra, Volterra-GOB-Tucker and Volterra-PARAFAC ($R = 1$).

Table VII. Performance comparison between Volterra, Volterra-GOB-Tucker and Volterra-PARAFAC models.

| | Volterra | Volterra-GOB | Volterra-PARAFAC | |
|---|---|---|---|---|
| | $M = 15$ | $M = 20$ | $M = 15$ | $M = 20$ |
| Number of parameters | 816 | 47 | 46 | 61 |
| O-NMSE identification (dB) | −50.42 | −59.75 | −55.10 | −57.24 |
| O-NMSE validation (dB) | −49.55 | −60.90 | −51.61 | −56.02 |



DOI: 10.1002/acs

simulation results, we can conclude that both the Volterra-GOB-Tucker and Volterra-PARAFAC models provide a significant parametric complexity reduction. For a similar parametric complexity, the performance obtained with the Volterra-GOB-Tucker model is slightly better than that obtained with the Volterra-PARAFAC one. Therefore, we can conclude that both reduced complexity Volterra models are efficient for representing the simulated polymerization reactor with few parameters compared with the standard Volterra model. However, the advantage of using the Volterra-PARAFAC model resides in its simplicity, all the design parameters being estimated by means of an extended Kalman filter without a priori knowledge on the system to be identified, whereas the task of determining the GOB poles is much more complex and needs a priori knowledge on the system dynamics.

## 8. CONCLUSION

In this paper, a new formulation of the Volterra model has been derived using a PARAFAC decomposition of its kernels considered as symmetric tensors. The obtained Volterra-PARAFAC model that can be viewed as a simplified PCW model, allows to significantly reduce the parametric complexity when the rank of the kernels is small with respect to their memory. Three adaptive methods have been proposed for estimating the PARAFAC parameters: the ECKF, the CLMS, and the normalized CLMS. The performance of these algorithms has been compared by means of computer simulations. As expected, the ECKF gives the best results in terms of speed of convergence. Moreover, it is possible to run several ECKFs in parallel, each filter being associated with a different choice of kernel rank, for a joint structure and parameter estimation. On the other hand, the CLMS and NCLMS algorithms allow a parallelization of the computation of the PARAFAC parameters of each kernel. The proposed Volterra-PARAFAC model has been validated with a simulated polymerization reactor and its performance has been compared with that of a Volterra-GOB-Tucker model. Both Volterra models allow a similar parametric complexity reduction but with a much simpler utilization of the Volterra-PARAFAC model.

The theoretical convergence analysis of the proposed adaptive parameter estimation algorithms, in terms of steady state misadjustment and speed of convergence, is a topic for future research. A perspective of this work also consists in developing adaptive estimation algorithms with selective partial updates of the PARAFAC parameters, as it was proposed for FIR linear filters [61, 62]. Another perspective concerns an extension of the Volterra-PARAFAC model for modeling nonlinear communication systems, for which there is not even order homogeneous terms and the homogeneous term of order $(2p+1)$ contains $(p+1)$ nonconjugated inputs and p conjugated inputs [12, 13], which induces a double symmetry in the PARAFAC decomposition of the odd order kernels.

## REFERENCES

1. Mumolo E, Francescato D. Adaptive predictive coding of speech by means of Volterra predictors. *Proceedings of IEEE Winter Workshop on Nonlinear Digital Signal Processing.* Tampere, Finland, 1993; 2.1.4.1–2.1.4.4.
2. Frank WA. An efficient approximation to the quadratic Volterra filter and its application in real-time loudspeaker linearization. *Signal Processing* 1995; **45**(1):97–113.
3. Kajikawa Y. Subband parallel cascade Volterra filter for linearization of loudspeaker systems. *16th European Signal Processing Conference (EUSIPCO)*, Lausanne, Switzerland, 2008.
4. Doyle FJ, III, Pearson RK, Ogunnaike BA. *Identification and Control Using Volterra Models*. Springer-Verlag: London, 2002.
5. Genceli H, Nikolaou M. Design of robust constrained model-predictive controllers with Volterra series. *AIChE Journal* 1995; **41**(9):2098–2107.
6. Maner BR, Doyle FJ, III, Ogunnaike BA, Pearson RK. Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models. *Automatica* 1996; **32**(9):1285–1301.
7. Tan L, Jiang J. Adaptive Volterra filters for active control of nonlinear noise processes. *IEEE Transactions on Signal Processing* 2001; **49**(8):1667–1676.
8. Zeller M, Kellerman W. Fast adaptation of frequency-domain Volterra filters using inherent recursions of iterated coefficient updates. *15th European Signal Processing Conference (EUSIPCO).* Poznan, Poland, 2007; 1605–1609.
9. Korenberg MJ, Hunter IW. The identification of nonlinear biological systems: Volterra kernel approaches. *Annals of Biomedical Engineering* 1996; **24**:250–268.



DOI: 10.1002/acs

10. Marmarelis VZ. *Nonlinear Dynamic Modeling of Physiological Systems*. IEEE Press, John Wiley & Sons: Piscataway, NJ, 2004.
11. Benedetto S, Biglieri E. Nonlinear equalization of digital satellite channels. *IEEE Journal on Selected Areas in Communications* 1983; **SAC-1**:57–62.
12. Benedetto S, Biglieri E, Daffara S. Modeling and performance evaluation of nonlinear satellite links- a Volterra series approach. *IEEE Transactions on Aerospace Electronic Systems* 1979; **AES-15**(4):494–507.
13. Cheng CH, Powers EJ. Optimal Volterra kernel estimation algorithms for a nonlinear communication system for PSK and QAM inputs. *IEEE Transactions on Signal Processing* 2001; **49**(1):147–163.
14. Fernandes CAR, Favier G, Mota JCM. Blind identification of multiuser nonlinear channels using tensor decomposition and precoding. *Signal Processing* 2009; **89**(12):2644–2656.
15. Kibangou AY, Favier G. Blind equalization of nonlinear channels using tensor decompositions with code/space/time diversities. *Signal Processing* 2009; **89**(2):133–143.
16. Mileounis G, Koukoulas P, Kalouptsidis N. Input-output identification of nonlinear channels using PSK, QAM and OFDM inputs. *Signal Processing* 2009; **89**(7):1359–1369.
17. Boyd S, Chua LO. Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems* 1985; **CAS-32**(11):1150–1161.
18. Mathews VJ, Sicuranza G. *Polynomial Signal Processing*. John Wiley & Sons: New York, 2000.
19. Haber R, Keviczky L. *Nonlinear System Identification: Input-Ouput Modeling Approach, Vol. 1: Nonlinear System Parameter Identification*. Kluwer Academic Publishers: Dordrecht, The Netherlands, 1999.
20. Kibangou AY, Favier G. Tensor analysis-based model structure determination and parameter estimation for block-oriented nonlinear systems. *IEEE Journal of Selected Topics in Signal Processing* 2010; **4**(3): 514–525.
21. Wiener N. *Nonlinear Problems in Random Theory*. Wiley: New-York, 1958.
22. Mäkilä PM. Approximation of stable systems by Laguerre filters. *Automatica* 1990; **26**(2):333–345.
23. Wahlberg B. System identification using Laguerre models. *IEEE Transactions on Automatic Control* 1991; **36**(5):551–562.
24. Wahlberg B. System identification using Kautz models. *IEEE Transactions on Automatic Control* 1994; **39**(6): 1276–1282.
25. Wahlberg B, Mäkilä PM. On approximation of stable linear dynamical systems using Laguerre and Kautz functions. *Automatica* 1996; **32**(5):693–708.
26. Heuberger PSC, Van den Hof PMJ, Bosgra OH. A generalized orthonormal basis for linear dynamical systems. *IEEE Transactions on Automatic Control* 1995; **40**(3):451–465.
27. Ninness B, Gustafsson F. A unifying construction of orthonormal bases for system identification. *IEEE Transactions on Automatic Control* 1997; **42**(4):515–521.
28. Marmarelis VZ. Identification of nonlinear biological systems using Laguerre expansions of kernels. *Annals of Biomedical Engineering* 1993; **21**:573–589.
29. Dumont GA, Fu Y. Non-linear adaptive control via Laguerre expansion of Volterra kernels. *International Journal of Adaptive Control and Signal Processing* 1993; **7**:367–382.
30. Isaksson M, Rönnow D. A parameter-reduced Volterra model for dynamic RF power amplifier modeling based on orthonormal basis functions. *International Journal of RF and Microwave Computer-Aided Engineering* 2007; **17**(6). DOI: 10.1002.
31. Zhu A, Brazil TJ. RF power amplifier behavioral modeling using Volterra expansion with Laguerre functions. *Proceedings of the IEEE International Microwave Theory and Techniques Symposium.* Long Beach, vol. WE4D-1, 2005.
32. Campello RJGB, Amaral WC, Favier G. A note on the optimal expansion of Volterra models using Laguerre functions. *Automatica* 2006; **42**(4):689–693.
33. Campello RJGB, Favier G, Amaral WC. Optimal expansions of discrete-time Volterra models using Laguerre functions. *Automatica* 2004; **40**(5):815–822.
34. da Rosa A, Campello RJGB, Amaral WC. Choice of free parameters in expansions of discrete-time Volterra models using Kautz functions. *Automatica* 2007; **43**(6):1084–1091.
35. Hacioglu R, Williamson G. Reduced complexity Volterra models for nonlinear system identification. *EURASIP Journal on Applied Signal Processing* 2001; **4**:257–265.
36. Kibangou AY, Favier G, Hassani MM. Selection of generalized orthonormal bases for second-order Volterra filters. *Signal Processing* 2005; **85**:2371–2385.
37. Kibangou AY, Favier G, Hassani MM. Laguerre-Volterra filters optimization based on Laguerre spectra. *EURASIP Journal on Applied Signal Processing* 2005; **17**:2874–2887.
38. Nowak R, Van Veen BD. Tensor product basis approximations for Volterra filters. *IEEE Transactions on Signal Processing* 1996; **44**(1):36–50.
39. Panicker TM, Mathews VJ. An extended Kalman filter for parallel-cascade truncaded Volterra systems. *Thirty-First Asilomar Conference on Signals, Systems & Computers* 1997; **1**:18–22.
40. Panicker TM, Mathews VJ. Parallel-cascade realizations and approximations of truncated Volterra systems. *IEEE Transactions on Signal Processing* 1998; **46**(10):2829–2831.
41. Harshman RA. Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics* 1970; **16**:1–84.

42. Khouaja A, Favier G. Identification of PARAFAC-Volterra cubic models using an alternating recursive least squares algorithm. *12th European Signal Processing Conference (EUSIPCO).* Vienna, Austria, 2004; 1903–1906.
43. Favier G, Bouilloc T. Identification de modèles de Volterra basée sur la décomposition PARAFAC. *GRETSI Symposium*, Dijon, France, 2009.
44. Favier G, Bouilloc T. Parametric complexity reduction of Volterra models using tensor decompositions. *17th European Signal Processing Conference (EUSIPCO).* Glasgow, Scotland, 2009.
45. Tucker LR. Some mathematical notes on three-mode factor analysis. *Psychometrika* 1966; **31**(3):279–311.
46. de Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications* 2000; **21**(4):1253–1278.
47. Hitchcock FL. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics Cambridge* 1927; **6**(3):164–189.
48. Kruskal JB. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications* 1977; **18**(2):95–138.
49. Comon P, Golub G, Lim LH, Mourrain B. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications* 2008; **30**(3):1254–1279.
50. Harshman RA. Determination and proof of minimum uniqueness conditions for PARAFAC 1. *UCLA Working Papers in Phonetics* 1972; **22**:111–117.
51. Kruskal JB. Rank, decomposition and uniqueness for 3-way and $N$-way arrays. In *Multiway Data Analysis*, Coppi R, Bolasco S (eds). Elsevier: Amsterdam, 1989; 7–18.
52. Sidiropoulos ND, Giannakis GB, Bro R. Blind PARAFAC receivers for DS-CDMA systems. *IEEE Transactions on Signal Processing* 2000; **48**(3):810–823.
53. Sidiropoulos ND, Bro R. On the uniqueness of multilinear decomposition of $N$-way arrays. *Journal of Chemometrics* 2000; **14**:229–239.
54. Fernandes CER, Favier G, Mota JCM. Blind channel identification algorithms based on the PARAFAC decomposition of cumulant tensors: the single and multiuser cases. *Signal Processing* 2008; **88**(6):1382–1401.
55. Rajih M, Comon P, Harshman RA. Enhanced line search: a novel method to accelerate PARAFAC. *SIAM Journal on Matrix Analysis and Applications* 2008; **30**(3):1148–1171.
56. Tomasi G, Bro R. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis* 2006; **50**(7):1700–1734.
57. Korenberg MJ. Parallel cascade identification and kernel estimation for nonlinear systems. *Annals of Biomedical Engineering* 1991; **19**:429–455.
58. Kibangou AY, Favier G. Identification of parallel-cascade Wiener systems using joint diagonalization of third-order Volterra kernel slices. *IEEE Signal Processing Letters* 2009; **16**(3):188–191.
59. Congalidis JP, Richards JR, Ray WH. Feedforward and feedback control of a solution copolymerization reactor. *AIChE Journal* 1989; **35**(6):891–907.
60. Soni AS. Control-relevant system identification using nonlinear Volterra and Volterra-Laguerre models. *Ph.D. Thesis*, University of Pittsburgh, USA, 2006.
61. Dogançay K, Tanrikulu O. Adaptive filtering algorithms with selective partial updates. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing* 2001; **48**(8):762–769.
62. Douglas SC. Adaptive filters employing partial updates. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing* 1997; **44**(3):209–216.



DOI: 10.1002/acs