Efficient Computation of Bilinear Approximations and Volterra Models of Nonlinear Systems

Phillip Mark Seymour Burt, José Henrique de Morais Goulart

Efficient Computation of Bilinear Approximations and Volterra Models of Nonlinear Systems

Phillip Mark Seymour Burt, *Member, IEEE*, and José Henrique de Morais Goulart

***Abstract*—Bilinear and Volterra models are important when dealing with nonlinear systems which arise in several signal processing applications. The former can approximate a large class of systems affine in the input with relatively low parametric complexity. Such an approximate bilinear model can be derived by means of Carleman bilinearization (CB). Then, a Volterra model can be computed from it, having the advantage of being linear in the parameters, but often involving a large number of them. In this paper we develop efficient routines for CB and for computing the Volterra kernels of a bilinear system. We argue that they are useful for studying a class of systems for which a reference physical model is known. In particular, the so-derived kernels allow assessing the suitability of a Volterra filter and of other alternatives for modeling the system of interest. Techniques exploiting sparsity and low rank of involved matrices are proposed for alleviating computing cost. Several examples are given along the paper to illustrate their use, based on existing physical models of loudspeakers.**

***Index Terms*—Nonlinear processing, Volterra filter, Carleman Bilinearization, Loudspeaker modeling**

I. Introduction

THE modeling and identification of nonlinear systems has historically been a well-studied subject [1]–[11], to which many contributions have also been given recently [12]–[17]. In signal processing, the interest in using nonlinear instead of linear models comes frequently from their capability of better representing certain physical systems that underlie problems such as acoustic echo cancellation [18], spectral regrowth analysis [19], equalization [20], active noise control [14], [21] and linearization [22]–[24]. This is also true in other areas, such as estimation [25], industrial plant control [15], power systems [26] and sensoring [27]. Nonlinear models are often decisive to attain performance levels not met with linear models—as, for instance, in acoustic echo cancellation for a miniaturized loudspeaker with high power input [18].

P. M. S. Burt is with Escola Politécnica, University of São Paulo, CEP 05508-010 São Paulo, SP, Brazil (email: phillip@lcs.poli.usp.br).

J. H. de M. Goulart is with Univ. Grenoble Alpes, CNRS, GIPSA-lab, F-38000 Grenoble, France (email: jose-henrique.de-morais-goulart@gipsa-lab.fr). During his Ph.D., he was supported by CNPq-Brazil, under the program Ciência sem Fronteiras. He is now supported by the European Research Council under the European Programme FP7/2007-2013, Grant AdG-2013-320594 "DECODA."

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material consists of report [65]. Contact phillip@lcs.poli.usp.br for further questions.

In many cases, a nonlinear continuous-time state-space description of a component that is central to the problem in question can be obtained from physical considerations. We refer to it then as a *physical model* (PM). For instance, PMs of loudspeakers [23], high-frequency transistors [28], [29], ADC sampling gates [30], optical devices [31], power systems [26], MEMS sensors [27], robotic structures [32] and chemical reactors [33] can be mentioned. Only typical values of the PM parameters may be known initially, so, most likely, they have to be refined using an estimation procedure.

More generally, a nonlinear system can be represented by a Volterra model, which is based on the Volterra series functional expansion [2], [4]–[7], [11]. Since the seminal work of Wiener [1] the Volterra model has been applied to many different problems [18]–[22], [34], [35] more conveniently than a state-space model. In particular, it is quite simple to pass from time domain to frequency domain (see Section II-A) and from continuous-time to discrete-time (see Section V-B). Following time discretization and truncation of order and memory length, simulations can be carried out efficiently, especially when varying only the scale of a given input signal (see Section VI-D). Also, the output is linear in the parameters, which makes their estimation easier, and the model is always stable. On the negative side, the parameter count grows rapidly with the chosen order and memory length.

For a nonlinear system belonging to a large class (of the so called linear-analytic systems), expressions can be obtained for the Volterra model in terms of the Taylor expansions of its state-space functions [4], [9]. Taking a linear-analytic PM with typical parameters, we term as a *reference Volterra model* the Volterra model given by those expressions.

As discussed in more detail later, such a reference model can be used, for instance, 1) to realistically assess the structural requirements of a Volterra or related approach to signal processing problems such as echo cancellation and linearization, 2) as an initial point for parameter estimation and as a tool in the validation and evaluation of estimation algorithms themselves, and 3) to efficiently perform simulations, nonlinear distortion calculations, and fault diagnosis of systems.

Regarding the first point in particular, even when the number of parameters is too large for the real-time use of the model, it can aid in the choice of appropriate structural characteristics (as, *e.g.*, memory length and order) for other (possibly less general) classes of models which are not necessarily linear in their parameters) and may also supply suitable initial conditions for their adaptation. This strategy can be applied, for instance, to conventional block-oriented structures (BOS) [16], BOS based on multilinear algebra concepts [13], [15], [36],

sparse-interpolated VFs[1] [37], VFs with kernels containing only coefficients within some defined distance to their main diagonals [17] and even NARMAX models [7, Sec. 3.4]. In general, any nonlinear model whose relationship with the Volterra model is well understood can be assessed.

Another way of modeling a nonlinear system is by means of a state-space bilinear approximation. Bilinear systems, in particular their stability, are better understood than general nonlinear systems [4], [6], [38]–[43]. Also, their discretization in time is similar to that of linear systems [41] and their identification is well-studied [44]–[47]. After time discretization, the possible advantages over the Volterra approach are the smaller number of parameters and realizable infinite memory.

Carleman bilinearization (CB) [4], [9], [40], [41], [48] is a widely applied method for approximating a linear-analytic system by a bilinear system having exactly the same associated Volterra model up to a prescribed order [25], [26], [49], [50]. In fact, because of this property, CB is also used as the first step in obtaining the Volterra model of a linear-analytic system [4], [9], [48]. Applying CB to a physical model with typical parameters provides a *reference bilinear model*, most of its uses being analogous to those of a reference Volterra model. An otherwise specific use, which is discussed later, would be as an intermediate step to lower-order bilinear approximations of a nonlinear system.

Although CB and the formula for the Volterra kernels of a bilinear system are well known, the actual computation of such bilinearization and kernels can be very computationally intensive and has received little attention in the literature. In this paper, we carry out an analysis of such computations and develop a set of efficient routines that greatly reduce their load. For concreteness, implementation aspects are discussed considering the use of Matlab and C routines.

We emphasize that this approach relies on the key assumption that a PM of the system of interest is available, which, however, does not imply that its direct use for signal processing purposes is convenient (see Section II-C). The approach also relies on the assumption that a truncated Volterra series representation of the model is desired for practical purposes. As discussed above, these include simulation and nonlinear distortion analysis, as well as the development of Volterra-related models having lower computational complexity.

To the best of our knowledge, [12] and this work are the first ones to explicitly describe efficient routines for CB implementation and for kernel computation of the derived bilinear model. In [12] we addressed mainly the CB step and computed the kernels via the frequency domain. The Taylor expansion prior to CB was not addressed in that work. Here, we show that all three steps are important regarding computational complexity and develop more efficient computational procedures for each one. More specifically, the Taylor expansion uses literal parameters and exploits sparsity, the CB exploits sparsity and kernel computation exploits recursiveness in the time domain and a low-rank decomposition. As part of the extensive numerical results we present, it follows that, in relation to [12], computing a bilinear model can be more than 30 times faster (see Section III-H) and computing kernels can be more than 300 times faster (see Section IV-E). The numerical computation of kernels from state-space equations was also addressed in [51], [52] but relying on a more complex and less flexible method, as discussed in Section IV-E.

This work is organized as follows. In Section II, Volterra and state-space models are reviewed, together with Carleman bilinearization (CB). From Sections III-B to III-G, techniques for the efficient computation of CB are presented and in Section III-H overall results are compared. In Sections IV and V, an efficient computation of the Volterra kernels of a bilinear system is presented and their related discrete-time kernels are obtained. Section VI discusses application in signal processing. Finally, Section VII presents our conclusions.

Notational conventions: We denote scalars by italic letters (p, P), vectors by boldface lowercase letters ($\mathbf{x}, \mathbf{w}$), matrices by boldface uppercase letters ($\mathbf{F}, \mathbf{G}$), tuples by boldface italic letters, as $\boldsymbol{k} \triangleq (k_1, \ldots, k_p)$, with $\boldsymbol{k}T = (k_1T, \ldots, k_pT)$, and $p-$dimensional volume differentials as $d\boldsymbol{\tau} \triangleq d\tau_1 \ldots d\tau_p$. $\mathbf{0}$ stands either for the null vector or for the null matrix (which is always clear from the context) and $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. The operator vec$(\cdot)$ joins the transposed columns of its matrix argument, yielding a row vector. The Kronecker and the Hadamard products are denoted by $\otimes$ and $\odot$, respectively. For f differentiable with respect to $x_1, \ldots, x_m$, we define $[x_1 \ldots x_m]_{\partial f} \triangleq [\frac{\partial f}{\partial x_1} \ldots \frac{\partial f}{\partial x_m}]$. Though not usual, this allows a considerable gain of space with little loss of intelligibility.

II. Carleman bilinearization for Volterra modeling

A. The Volterra model

The Volterra representation of a causal continuous-time system consists of an input/output relation of the form [2]

$$y(t) = \sum_{p=1}^{\infty} y_p(t), \qquad y_p(t) = \int_{\mathbb{R}_+^p} h_p(\boldsymbol{\tau}) \prod_{i=1}^{p} u(t-\tau_i) d\boldsymbol{\tau}, \quad (1)$$

in which $y_p(t)$ results from the multidimensional convolution of the input $u(t)$ with the pth-order *Volterra kernel* $h_p(\boldsymbol{\tau})$. For uniqueness and without loss of generality, $h_p(\boldsymbol{\tau})$ can be assumed triangular, *i.e.* $h_p(\boldsymbol{\tau}) = 0$ if not $\tau_1 \leq \ldots \leq \tau_p$, or symmetric, *i.e.* invariant under permutation of $\tau_1, \ldots, \tau_p$ [2], [4], [6]. The Volterra kernels may be seen as generalizations of the impulse response of a linear time-invariant system and constitute a fundamental description of the nonlinear dynamics of a system. Their frequency-domain counterparts, known as *generalized frequency response functions* (GFRFs), are obtained via the multidimensional Fourier transform

$$H_p(j\boldsymbol{\Omega}) = \int_{\mathbb{R}_+^p} h_p^{(s)}(\boldsymbol{\tau}) e^{-j(\Omega_1\tau_1 + \ldots + \Omega_p\tau_p)} d\boldsymbol{\tau}, \quad (2)$$

where $h_p^{(s)}$ are the symmetrical kernels. GFRFs are convenient, for instance, to assess typical nonlinear phenomena such as harmonic and intermodulation distortion [53].

[1] In the context of signal processing, the term *Volterra filter* (VF) is often used in connection with the discrete-time version of the truncated Volterra series.

In digital signal processing, the *Volterra filter* (VF) is a discrete-time truncated (to order P and memory lengths N_p) version of (1):

$$y(n) = \sum_{p=1}^{P} \sum_{k_1=0}^{N_p-1} \cdots \sum_{k_p=k_{p-1}}^{N_p-1} v_p(\boldsymbol{k}) \prod_{i=1}^{p} u(n-k_i), \quad (3)$$

where the *discrete-time kernels* v_p are in the triangular form, thereby requiring the minimal number of parameters [6]. The GFRFs are analogous to (2) and are obtained efficiently with a multidimensional fast Fourier transform.

The VF is a finite sum and thus always stable and is also linear in the model parameters $v_p(\boldsymbol{k})$, favoring their adaptation by standard least-squares or least mean squares (LMS) algorithms. In addition, the generality of VFs can be shown either by applying the Stone-Weierstrass theorem to the approximation of input/output mappings with finite memory [14], or by relating them to discrete-time systems with fading memory [11]. On the other hand, a serious drawback is that the number of parameters $\sum_{p=1}^{P} \binom{N_p+p-1}{p}$ grows very rapidly with P or N_p when $P > 1$ and $N_p > 1$.

B. Linear-analytic and bilinear systems

The large class of *linear-analytic* (nonlinear) systems [4], [9], [40], [42], [54] has analytic state-space functions in which the input enters linearly (or more precisely, in an affine manner). For single input and output and $\mathbf{x}(t) \in \mathbb{R}^m$, this has the form

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))u(t), \quad (4)$$

$$y(t) = \lambda(\mathbf{x}(t)), \quad (5)$$

where $\mathbf{f} = [f_1 \dots f_m]^T$, with $f_i(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}$, is analytic in some open region $X \subset \mathbb{R}^m$, and likewise for $\mathbf{g}$ and λ.

A more tractable subset of such class is that of *bilinear systems* [4], [6], [38]–[43], for which $\mathbf{f}(\mathbf{x}) = \mathbf{F}\mathbf{x}$, $\mathbf{g}(\mathbf{x}) = \mathbf{G}\mathbf{x}+\mathbf{b}$ and $\lambda(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$. In particular, under mild assumptions their triangular Volterra kernels are given by (*e.g.*, [4, Sec. 3.1])

$$h_p(\boldsymbol{\tau}) = \mathbf{c}^T e^{\mathbf{F}\tau_1} \mathbf{G} e^{\mathbf{F}(\tau_2-\tau_1)} \dots \mathbf{G} e^{\mathbf{F}(\tau_p-\tau_{p-1})} \mathbf{b}. \quad (6)$$

C. Problems with nonlinear state-space models in practice

Physical models in a nonlinear state-space form such as (4)-(5) are often simple and have small dimension m. However, as discussed below, the direct use of such models does not necessarily imply a low computation complexity in signal processing procedures. This motivates computing Volterra (and Volterra-related) models and bilinear approximations from those physical models and possibly using them instead in signal processing procedures.

1) Discrete-time implementation or simulation: To the best of our knowledge, differently from the linear case, there is no general solution to the problem of obtaining a discrete-time state-space equivalent of (4)-(5) that gives $y_d(n) = y(nT)$ for a prescribed T. [7], [55] aim at obtaining a discrete-time state-space model whose kernels up to a prescribed order P are sampled versions of the continuous-time kernels. However, stability of the obtained discrete-time model is not assured and the development is limited to $P = 2$ and to the second-degree polynomial case in which $f_i(\mathbf{x}) = \sum_{j=1}^{m}\sum_{k=1}^{m} a_{i,j,k}x_jx_k + \sum_{j=1}^{m} b_{i,j}x_j$. Even in this simple case, the development (based on partial fraction expansions of generalized transfer functions) is quite laborious. Furthermore, it is unclear if an extension to higher-degree polynomial functions or to higher values of P is feasible. Otherwise, an approximate discrete-time implementation or simulation of (4)-(5) can be obtained by numerical integration in a relatively straightforward manner [56], [57] . However, even for simple continuous-time models, computational complexity can be much higher than what might be expected. So-called explicit integration methods are not assuredly stable and may require a very small integration step for stability and accuracy, the output being then decimated to the desired sampling rate. On the other hand, implicit methods of first and second orders are stable but require solving a nonlinear system of equations at each step and may still require a small step for accuracy.

2) Frequency domain analysis: In applications, one is often concerned with the spectral analysis of some nonlinear system. For many systems of practical relevance, this task can be performed by inspecting GFRFs, since they provide a complete picture of their spectral behaviour. This is the case, in particular, for linear-analytic systems. However, differently from the linear case, the GFRFs do not follow directly from the state-space model. Instead, their computation must resort to some more involved method such as the growing exponential approach [4] or the generalized harmonic balance method [7].

3) Identification: The complexity of identifying nonlinear state-space models can be assessed in [58] and the references therein. The authors employ the Expectation Maximization (EM) algorithm with a particle smoother instead of a standard linear smoother and exemplify its performance for a nonlinear stochastic model with dimension $m = 1$ and 6 parameters.

D. Carleman bilinearization

Carleman bilinearization (CB) [4], [9], [40], [41], [48] provides a bilinear approximation of a linear-analytic system. Though other techniques exist for this purpose, CB is effectively a constructive method, differently from [59][2].

In the following we drop the time argument in (4)–(5) for notational simplicity and define $\mathbf{x}_p = \mathbf{x} \otimes \mathbf{x}_{p-1}$ for $p > 0$ and $\mathbf{x}_0 = 1$, which gives $\mathbf{x}_1 = \mathbf{x} = [x_1\ x_2 \dots x_m]^T$, $\mathbf{x}_2 = [x_1^2\ x_1x_2 \dots x_1x_m\ x_2x_1 \dots x_2x_m \dots x_m^2]^T$ and so forth. From (4) we can then write

$$\mathbf{x}' \overset{P}{=} \sum_{p=1}^{P} \mathbf{F}_{1,p}\mathbf{x}_p + \sum_{p=0}^{P-1} \mathbf{G}_{1,p}\mathbf{x}_p u, \quad (7)$$

where $\overset{P}{=}$ denotes equality up to the Pth-order terms of Taylor expansions (with respect to $x_1, \dots, x_m$ and u) around the origin and $\mathbf{F}_{1,p}$ and $\mathbf{G}_{1,p} \in \mathbb{R}^{m \times m^p}$ contain the coefficients of the Taylor expansions with respect to $x_1, \dots, x_m$ of $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$, respectively. With no loss of generality [4], we assume

[2]Computing the bilinear approximation of [59] requires the family of solutions to $\mathbf{x}' = \mathbf{f}(\mathbf{x})$ and $\mathbf{x}' = \mathbf{g}(\mathbf{x})$.

that the system is initially at rest (*i.e*, $\mathbf{x}(0) = \mathbf{0}$) and that the solution for $u(t) \equiv 0$ is $\mathbf{x}(t) \equiv \mathbf{0}$. Thus, $\mathbf{F}_{1,0} \equiv 0$.

Now, from (7) and the recursion $(\mathbf{x} \otimes \mathbf{x}_{p-1})' = \mathbf{x}' \otimes \mathbf{x}_{p-1} + \mathbf{x} \otimes \mathbf{x}'_{p-1}$ we can write for $q \geq 2$ the differential equations [4]

$$\mathbf{x}'_q \overset{P}{=} \sum_{p=1}^{P-q+1} \mathbf{F}_{q,p+q-1}\mathbf{x}_{p+q-1} + \sum_{p=0}^{P-q} \mathbf{G}_{q,p+q-1}\mathbf{x}_{p+q-1}u \quad (8)$$

where, using $\mathbf{I}_{m^l} = \mathbf{I}_m \otimes \mathbf{I}_{m^{l-1}} = \mathbf{I}_{m^{l-1}} \otimes \mathbf{I}_{m^l}$ (with $\mathbf{I}_0 = 1$), $\mathbf{F}_{q,p}$ can be expressed as

$$\mathbf{F}_{q,p} = \sum_{l=0}^{q-1} \mathbf{I}_{m^l} \otimes \mathbf{F}_{1,p-q+1} \otimes \mathbf{I}_{m^{q-l-1}} \quad (9)$$

and $\mathbf{G}_{q,p}$ is given by an identical formula.

From (7), (8) and (5) and with the extended state vector

$$\mathbf{x}_\otimes = [\mathbf{x}^T \;\; \mathbf{x}_2^T \;\; \mathbf{x}_3^T \;\; \dots \;\; \mathbf{x}_P^T]^T \in \mathbb{R}^{M_P}, \quad (10)$$

where $M_P = \sum_{p=1}^{P} m^p$, we can write

$$\mathbf{x}'_\otimes \overset{P}{=} \mathbf{F}\mathbf{x}_\otimes + \mathbf{G}\mathbf{x}_\otimes u + \mathbf{b}u, \quad (11)$$

where $\mathbf{b} = [\mathbf{G}_{1,0}^T \; 0 \dots 0]^T$ and

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} & \dots & \mathbf{F}_{1,P} \\ \mathbf{0} & \mathbf{F}_{2,2} & \dots & \mathbf{F}_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{F}_{P,P} \end{bmatrix}, \mathbf{G} = \begin{bmatrix} \mathbf{G}_{1,1} & \mathbf{G}_{1,2} & \dots & \mathbf{G}_{1,P-1} & \mathbf{0} \\ \mathbf{G}_{2,1} & \mathbf{G}_{2,2} & \dots & \mathbf{G}_{2,P-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{3,2} & \dots & \mathbf{G}_{3,P-1} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{G}_{P,P-1} & \mathbf{0} \end{bmatrix}. \quad (12)$$

With $\mathbf{c}$ containing the coeffcients of the Pth order Taylor expansion of (5), we define now the bilinear system

$$\begin{aligned} \mathbf{v}'(t) &= \mathbf{F}\mathbf{v}(t) + \mathbf{G}\mathbf{v}(t)u(t) + \mathbf{b}u(t), \\ y(t) &= \mathbf{c}^T\mathbf{v}(t), \end{aligned}$$

which is an approximation of the original system (4)–(5). Importantly, its associated Volterra kernels of orders up to P are identical to those associated with (4)–(5) [4], [60].

III. Efficient Carleman bilinearization

In this section, we develop computational techniques for the efficient implementation of Carleman bilinearization, taking practical aspects into account[3]. Of particular interest is the possibility of performing the computation for multiple sets of PM parameter values (termed mPM in the following) without having to repeat all steps. This can be used, for instance, for measuring sensitivities in relation to PM parameters.

The performance of the proposed techniques will be exemplified in the course of the presentation with the setups described in Section III-A. The reported execution times are for a 2.67 GHz Intel Core I5 desktop computer with 16 GB of RAM.

[3]URL of the developed package: [61].

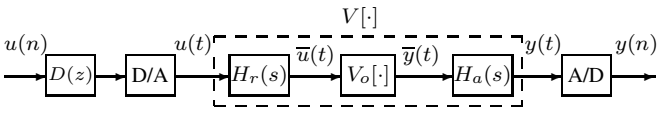

Fig. 1. Overall discrete-time setup.

A. Setup examples

Two setups are considered throughout the paper as examples: (A), the cascade of a zero-order hold (ZOH) D/A converter, a nonlinear loudspeaker modeled by the PM given in [62] and an A/D converter; and (B), the cascade of an ideal impulsive D/A converter, a 4th-order low-pass Butterworth reconstruction filter, same loudspeaker, an anti-aliasing filter identical to the reconstruction filter and an A/D converter.

Both setups are special cases of the general structure shown in Fig. 1, composed of the discrete-time filter $D(z)$, the ideal impulsive D/A converter, the overall system V and the A/D converter. In its turn, V is the cascade of the reconstruction filter H_r, the nonlinear system V_o and the anti-aliasing filter H_a. As discussed in Section V-B, it is samples of the Volterra kernels of V that need to be computed.

In setup (A), $D(z) = 1 - z^{-1}$, $H_r(s) = 1/s$ and $H_a(s) \equiv 1$. As follows from the discussion in Section V-C, the combination of such $D(z)$ and H_r with the ideal D/A converter is equivalent to a ZOH D/A converter and is convenient for calculating the discrete-time kernels of the setup. V_o is the aforementioned loudspeaker model, which has a state-space vector of dimension 3. With H_r and H_a then, the dimension of the state-space vector $\mathbf{x}(t)$ of V is $m = 4$. In setup (B), V_o is the same, $D(z) \equiv 1$, and $H_r(s)$ and $H_a(s)$ are 4th-order filters. Therefore, the dimension of $\mathbf{x}(t)$ is $m = 11$.

We note that the PM adopted in our examples is capable of closely reproducing the output (cone excursion) of real-world loudspeakers [23], [62].

B. Efficient Taylor expansion with literal PM parameters

We start with the computation of $\mathbf{F}_{1,p}$ and $\mathbf{G}_{1,p}$ in (7). Let the Jacobian matrix of a function $\phi(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}^n$ be denoted by $\mathbf{J}_\phi$. In particular, for $f_i(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}$ in (4), $\mathbf{J}_{f_i} = [x_1 \dots x_m]_{\partial f_i}$. Defining $\nabla_1 = \mathbf{J}_{f_i}$, the ith row of $\mathbf{F}_{1,1}$ is $\mathbf{f}_{i,1} = \nabla_1(\mathbf{0})$ and, thus, the ith row of $\mathbf{F}_{1,2}$ is $\mathbf{f}_{i,2} = \nabla_2(\mathbf{0})/2$, with $\nabla_2 = \text{vec}(\mathbf{J}_{\nabla_1})$. For instance, with $m = 3$,

$$\nabla_2 = \text{vec}\left(\begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 \\ x_1x_2 & x_2^2 & x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 \end{bmatrix}_{\partial f_i} \right)$$
$$= \begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_2 & x_2^2 & x_2x_3 & \dots & x_3^2 \end{bmatrix}_{\partial f_i}.$$

It is easy to verify that this can be generalized as follows:

Algorithm 1: The ith rows $\mathbf{f}_{i,p}$ of the Taylor expansion coefficient blocks $\mathbf{F}_{1,p}$ are recursively computed by

$$\begin{aligned} &\nabla_p = \text{vec}(\mathbf{J}_{\nabla_{p-1}}), \;\; p = 1, 2, \dots, P, \;\; \nabla_0 = f_i(\mathbf{x}) \\ &\mathbf{f}_{i,p} = \frac{1}{p!}\nabla_p(\mathbf{0}), \end{aligned}$$

and likewise for $\mathbf{G}_{1,p}$. □

TABLE I
EXECUTION TIMES (SEC) OF TAYLOR EXPANSION.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
	literal expansion / numerical evaluation			
(A)	0.15 / 0.0041	0.31 / 0.012	0.68 / 0.036	1.5 / 0.11
(B)	0.35 / 0.015	1.1 / 0.11	4.5 / 1.1	31 / 13
	numerical expansion			
(A)	0.083	0.18	0.46	1.2
(B)	0.20	0.65	3.8	33

TABLE II
FRACTION OF NONZERO COMPONENTS OF $\{\mathbf{F}_{1,p}\}$.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
(A)	0.23	0.12	0.055	0.023
(B)	0.044	5.9×10^{-3}	8.0×10^{-4}	1.0×10^{-4}

The algorithm above can be automated quite directly with any symbolic computation toolbox that provides a function to compute the Jacobian. We implemented the algorithm in Matlab, using the `jacobian` routine. The input functions f_i are of symbolic type, with literal (as opposed to numerical) PM parameters. It turns out that a large speed increase follows from converting ∇_p from symbolic to string type and computing $\nabla_p(\mathbf{0})$ in string type with the aid of a MEX function written in C, which replaces the literal variables x_i by zero. For instance, for setup (B) described above and $P=3$ speed was increased by a factor of 13. The evaluation of $\nabla_p(\mathbf{0})/p!$ with numerical PM parameters provides then the outputs $\mathbf{f}_{i,p}$.

Execution times for setups (A) and (B) are in the upper part of Table I[4]. In most cases, the final numerical evaluation step is more than 10 times faster than the previous literal Taylor expansion step. This is very convenient in the mPM context (see the beginning of this section), as only the final step needs to be repeated. Otherwise, it can be somewhat faster to start off with numerical values of the parameters instead of literal values, as can be seen in the lower part of Table I.

C. Using sparsity in Taylor expansion

When $f_i(\mathbf{x})$ and $g_i(\mathbf{x})$ depend only on a small fraction of the state vector elements x_i, many of the expansion coefficients are zero so $\mathbf{F}_{1,p}$ and $\mathbf{G}_{1,p}$ become quite sparse as p grows, which is exemplified in Table II. Sparsity can be used to speed up the numerical evaluation step, avoiding the unnecessary evaluation of null elements. For this, $\nabla_p(\mathbf{0})$ is converted to a sparse representation by a MEX function. The resulting execution times are in Table III, which compared to Table I show a 168 times faster numerical evaluation step for setup (B) and $P=5$, while the literal expansion step was not excessively delayed by the additional sparse conversion.

D. Taylor expansion with non-redundant form

There is redundancy in (7) since all elements $x_{j_1}x_{j_2}\dots x_{j_p}$ of $\mathbf{x}_p$ associated with permutations of a given $\{j_1,j_2,\dots,j_p\}$ could be added together. Carrying this out leads to a non-redundant alternative to (7), in which each $\bar{\mathbf{x}}_p$ has the minimal number of elements $\bar{m}_p=\binom{m+p-1}{p}$. For instance, for $m=3$ and $p=2$,

$$\begin{aligned}\mathbf{x}_2 &= [x_1^2\ x_1x_2\ x_1x_3\ x_2x_1\ x_2^2\ x_2x_3\ x_3x_1\ x_3x_2\ x_3^2]^T\\ \bar{\mathbf{x}}_2 &= [x_1^2\ x_1x_2\ x_1x_3\ x_2^2\ x_2x_3\ x_3^2]^T.\end{aligned}$$

We note that $\bar{m}_p$ grows much more slowly with m and p than m^p, which is desirable for implementation. However, the recursiveness of Algorithm 1 would seemingly be lost. It is surprising then that roughly the same algorithm can be applied here to the non-redundant form of the expansion

$$\mathbf{x}' \overset{P}{=} \sum_{p=1}^{P}\bar{\mathbf{F}}_{1,p}\bar{\mathbf{x}}_p + \sum_{p=0}^{P-1}\bar{\mathbf{G}}_{1,p}\bar{\mathbf{x}}_p u. \tag{13}$$

Indeed, the ith row of $\bar{\mathbf{F}}_{1,2}$ is $\bar{\mathbf{f}}_{i,2}=\sigma_2\odot\nabla_2(\mathbf{0})/2$, for a certain σ_2 and with $\nabla_2=\overline{\text{vec}}(\mathbf{J}_{\nabla_1})$ retaining from the ith column of $\mathbf{J}_{\nabla_1}$ only the elements without derivatives with respect to $x_1,...,x_{i-1}$. For instance, with $m=3$,

$$\nabla_2=\overline{\text{vec}}\left(\left[\begin{array}{ccc}\boxed{x_1^2} & x_1x_2 & x_1x_3\\ x_1x_2 & \boxed{x_2^2} & x_2x_3\\ x_1x_3 & x_2x_3 & \boxed{x_3^2}\end{array}\right]_{\partial f}\right)$$
$$=\left[\begin{array}{cccccc}x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2\end{array}\right]_{\partial f},$$

and $\sigma_2=[1\ 2\ 2\ 1\ 2\ 1]$ compensates for the discarded elements. The number of them to retain in the ith column of $\mathbf{J}_{\nabla_{p-1}}$ is

$$N_{p,i}=\sum_{l=i}^{m}N_{p-1,l},\quad N_{1,i}=1,\ \ i=1,2,\dots m,$$

Generalizing, we have:

Algorithm 2: The ith rows $\bar{\mathbf{f}}_{i,p}$ of $\bar{\mathbf{F}}_{1,p}$ in the non-redundant Taylor expansion are recursively computed by

$$\begin{aligned}&\nabla_p=\overline{\text{vec}}(\mathbf{J}_{\nabla_{p-1}}),\ \ p=1,2,\dots,P,\ \ \nabla_0=f_i(\mathbf{x})\\ &\bar{\mathbf{f}}_{i,p}=\frac{1}{p!}\sigma_p\odot\nabla_p(\mathbf{0}),\end{aligned}$$

and likewise for $\bar{\mathbf{G}}_{1,p}$. □

Implementation is similar to that discussed in Section III-B but now with a MEX $\overline{\text{vec}}(.)$ function operating on a string type input. Its string type output is evaluated as $\nabla_p(\mathbf{0})$ (as discussed in Section III-B), and then converted back to symbolic type for the recursion in p. The results are in Table IV and, against those in Table I, show that cutting redundancy pays off: for instance, for setup (B) ($m=11$) and $P=5$, the execution time is reduced by a factor of 39, closely following that of

TABLE III
EXECUTION TIMES (SEC) OF LITERAL TAYLOR EXPANSION/NUMERICAL EVALUATION, USING SPARSITY.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
(A)	0.11 / 0.0027	0.28 / 0.0073	0.66 / 0.022	1.5 / 0.058
(B)	0.32 / 0.0044	1.0 / 0.011	4.5 / 0.031	44 / 0.077

[4]In the interest of reproducibility, scripts for all results are in [61].

TABLE IV
EXECUTION TIMES (SEC) OF NON-REDUNDANT TAYLOR EXPANSION.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
	literal expansion / numerical evaluation			
(A)	0.15 / 0.0036	0.29 / 0.0063	0.53 / 0.011	0.85/ 0.019
(B)	0.35 / 0.010	0.86 / 0.033	2.6 / 0.11	6.8 / 0.33
	numerical expansion			
(A)	0.10	0.21	0.45	1.0
(B)	0.24	0.51	1.3	3.1

TABLE V
EXECUTION TIMES (SEC) OF LITERAL NON-REDUNDANT TAYLOR EXPANSION/NUMERICAL EVALUATION, USING SPARSITY.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
(A)	0.14 / 0.0023	0.30 / 0.0038	0.53 / 0.0066	0.84 / 0.012
(B)	0.33 / 0.0040	0.75 / 0.0061	2.5 / 0.010	6.6/ 0.017

the terms of the expansion, which is $\sum_{p=1}^{P} m^p / \sum_{p=1}^{P} \bar{m}_p = 177155/4367 = 41$ for $m = 11$ and $P = 5$.

As in Section III-C, sparse representations of $\bar{\mathbf{F}}_{1,p}$ and $\bar{\mathbf{G}}_{1,p}$ make their numerical evaluation faster. Results are in Table V, showing, for instance, a 19 times gain for setup (B) and $P = 5$ against that of Table IV.

E. Direct Carleman bilinearization

The Taylor expansion (in redundant form) of Section III-B provides blocks $\mathbf{F}_{1,p}$ and $\mathbf{G}_{1,p}$ in (12). The remaining $\mathbf{F}_{q,p}$ and $\mathbf{G}_{q,p}$ are then obtained from (9). No actual multiplications are required, since for any $\mathbf{A} \in \mathbb{R}^{q\times r}$, $\mathbf{I}_{m^l} \otimes \mathbf{A}$ yields a block-diagonal matrix where $\mathbf{A}$ appears m^l times, while $\mathbf{A} \otimes \mathbf{I}_{m^l}$ is made of qr blocks of size $m^l \times m^l$, each one given by the product of an element of $\mathbf{A}$ by $\mathbf{I}_l$. As to additions, $\mathbf{F}_{q,p}$ requires $q-1$ summations of $m^{p+q}-$element matrices, totaling (to obtain $\mathbf{F}$ and $\mathbf{G}$)

$$\mathcal{A}(m,P) = \sum_{q=2}^{P}(q-1)\left[\sum_{j=q}^{P} m^{p+j} + \sum_{j=q-1}^{P-1} m^{p+j}\right] \quad (14)$$

additions. This is exemplified in Table VI, along with the execution times of our Matlab implementation of the procedure. The blockwise computation of $\mathbf{F}$ and $\mathbf{G}$ allows quite vectorized and efficient code[5]. This can be assessed by noting, for instance, that the 61 ms required for setup (A) and $P = 5$, which involves 6.7×10^6 additions, is of the same order than the 22 ms required in our environment to add two $6.7 \times 10^6-$element vectors.

Comparing Tables VI and I, we see that direct CB dominates the total execution time as m and P grow, and especially so in the mPM case, where only the numerical evaluation step of the literal Taylor expansion would be carried out for each parameter set. Another problem is the memory requirement of

[5]We note that this direct CB was not implemented in [12].

TABLE VI
NUMBER OF ADDITIONS $\mathcal{A}$ (TOP) AND EXECUTION TIME IN SECONDS (BOTTOM) OF DIRECT CARLEMAN BILINEARIZATION.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
(A)	3.2×10^2	1.2×10^4	3.0×10^5	6.7×10^6
	2.4×10^{-4}	6.6×10^{-4}	4.5×10^{-3}	6.1×10^{-2}
(B)	1.6×10^4	4.1×10^6	7.5×10^8	1.2×10^{11}
	5.2×10^{-4}	5.8×10^{-2}	7.9	$1.3 \times 10^{3\dagger}$

†: estimate

$\mathbf{F}$ and $\mathbf{G}$, which have $M_P^2 = (\sum_{p=1}^{P} m^p)^2$ elements[6]. Both problems are addressed in the following.

F. Non-redundant Carleman bilinearization

In order to reduce computational and memory requirements, we now apply to the CB task the same non-redundancy and sparsity ideas we applied to the Taylor expansion[7].

A more efficient CB should obtain $\bar{\mathbf{F}}$ and $\bar{\mathbf{G}}$ from the non-redundant expansion (13) such that, analogously to (11),

$$\bar{\mathbf{x}}'_{\otimes} \overset{P}{=} \bar{\mathbf{F}}\bar{\mathbf{x}}_{\otimes} + \bar{\mathbf{G}}\bar{\mathbf{x}}_{\otimes}u + \mathbf{b}u, \quad (15)$$

where

$$\bar{\mathbf{x}}_{\otimes} = [\mathbf{x}^T \;\; \bar{\mathbf{x}}_2^T \;\; \bar{\mathbf{x}}_3^T \;\; \ldots \;\; \bar{\mathbf{x}}_P^T]^T \in \mathbb{R}^{\bar{M}_P}. \quad (16)$$

An initial obstacle to this is that, differently from the direct bilinearization, $\bar{\mathbf{x}}_p \neq \mathbf{x}\otimes\bar{\mathbf{x}}_{p-1}$ and, therefore, $\bar{\mathbf{F}}$ and $\bar{\mathbf{G}}$ cannot be calculated in blockwise fashion as in (9) and (12).

Nonetheless, a recursive *row-wise* calculation of these matrices, starting from (13), can be carried out as follows. The jth row of $\bar{\mathbf{x}}_{\otimes}$ in (15) has the form

$$(x_{j_1} \ldots x_{j_i})' \overset{P}{=} \bar{\mathbf{f}}_j\bar{\mathbf{x}}_{\otimes} + \bar{\mathbf{g}}_j\bar{\mathbf{x}}_{\otimes}u + b_j u, \quad (17)$$

where $i \leq P$, $1 \leq j_1 \leq \ldots \leq j_i \leq m$ and $\bar{\mathbf{f}}_j$ and $\bar{\mathbf{g}}_j$ are, respectively, the jth rows of $\bar{\mathbf{F}}$ and $\bar{\mathbf{G}}$. Using the product rule of derivation, the left hand-side of (17) satisfies

$$(x_{j_1} \ldots x_{j_i})' = x_{j_{r+1}} \ldots x_{j_i} \times (x_{j_1} \ldots x_{j_r})' + x_{j_1} \ldots x_{j_r} \times (x_{j_{r+1}} \ldots x_{j_i})', \quad (18)$$

$i > 1$ and $r < i$, and where $(x_{j_1} \ldots x_{j_r})'$ and $(x_{j_{r+1}} \ldots x_{j_i})'$ correspond to previous rows of $\bar{\mathbf{x}}'_{\otimes}$, say the kth and the lth row, respectively. For the kth row, for instance, we would have already determined $\bar{\mathbf{f}}_k$ and $\bar{\mathbf{g}}_k$ such that

$$(x_{j_1} \ldots x_{j_r})' \overset{P}{=} \bar{\mathbf{f}}_k\bar{\mathbf{x}}_{\otimes} + \bar{\mathbf{g}}_k\bar{\mathbf{x}}_{\otimes}u + b_k u. \quad (19)$$

In particular, if $k \leq m$ (or $l \leq m$), then the kth (lth) row of $\bar{\mathbf{x}}'_{\otimes}$ is given by the Taylor expansion (13), which provides the starting point of the recursion.

Now, using (19), the term $(x_{j_1} \ldots x_{j_r})' x_{j_{r+1}} \ldots x_{j_i}$ in (18) can be written as $(\bar{\mathbf{f}}_k\bar{\mathbf{x}}_{\otimes} + \bar{\mathbf{g}}_k\bar{\mathbf{x}}_{\otimes}u + b_k u)\, x_{j_{r+1}} \ldots x_{j_i}$. This corresponds to applying to each element of row vectors $\bar{\mathbf{f}}_k$

[6]In particular, for $m = 11$ and $P = 5$ the memory requirement largely exceeded the 16 GB of our system and thus the corresponding execution time in Table VI was estimated by the time measured for setup (A) and $P = 5$ multiplied by $\mathcal{A}(11,5)/\mathcal{A}(4,5)$.

[7]The redundancy of the direct CB is already pointed out in [4].

TABLE VII
NON-REDUNDANT CARLEMAN BILINEARIZATION OF ORDER P.

1: $\mathbf{F}(1:m,:) \leftarrow [\mathbf{0}_{m\times 1}\ \mathbf{F}_{1,1}\ \ldots\ \mathbf{F}_{1,P}]$
2: $\mathbf{G}(1:m,:) \leftarrow [\mathbf{G}_{1,0}\ \mathbf{G}_{1,1}\ \ldots\ \mathbf{G}_{1,P-1}\ \mathbf{0}_{m\times \bar{m}_P}]$
3: **for** $p = 2$ **to** P **do**
4: $\quad r \leftarrow \mathtt{floor}(p/2)$
5: $\quad$ **for** $j = \bar{M}_p - \bar{m}_p + 1$ **to** $\bar{M}_p$ **do**
6: $\quad\quad (j_1, j_2, \ldots, j_p) \leftarrow I^{-1}(j)$
7: $\quad\quad k \leftarrow I(j_1, \ldots, j_r)$
8: $\quad\quad l \leftarrow I(j_{r+1}, \ldots, j_p)$
9: $\quad\quad \mathbf{f}_j \leftarrow l \underline{\times} \mathbf{f}_k + k \underline{\times} \mathbf{f}_l$
10: $\quad\quad \mathbf{g}_j \leftarrow l \underline{\times} \mathbf{g}_k + k \underline{\times} \mathbf{g}_l$
11: $\quad$ **end for**
12: **end for**
13: $\mathbf{G} \leftarrow [\mathbf{G}(:, 2:\bar{M}_{P-1}+1)\ \mathbf{0}_{\bar{M}_P\times \bar{m}_P}]$
14: $\mathbf{F} \leftarrow \mathbf{F}(:, 2:\bar{M}_P)$

TABLE VIII
NUMBER OF ADDITIONS $\bar{\mathcal{A}}$ (TOP) AND EXECUTION TIME IN SECONDS (BOTTOM) OF NON-REDUNDANT CARLEMAN BILINEARIZATION.

Setup	$P=2$	$P=3$	$P=4$	$P=5$
(A)	1.8×10^{2}	1.4×10^{3}	6.7×10^{3}	2.3×10^{4}
	1.4×10^{-3}	3.9×10^{-3}	1.2×10^{-2}	1.7×10^{-2}
(B)	5.8×10^{3}	1.5×10^{5}	2.3×10^{6}	2.5×10^{7}
	8.3×10^{-3}	5.9×10^{-2}	3.6×10^{-1}	2.4

and $\bar{\mathbf{g}}_k$ a certain shift that depends on $(j_{r+1}, \ldots, j_i)$ and on the ordering of $\bar{\mathbf{x}}_\otimes$ implied by (16). As multi-index $(j_{r+1}, \ldots, j_i)$ is associated here with the lth row of $\mathbf{x}_\otimes$, we will indicate this shift operation by $l\underline{\times}$. Proceeding likewise for $(x_{j_{r+1}} \ldots x_{j_i})' x_{j_1} \ldots x_{j_r}$, it follows that $\bar{\mathbf{f}}_j$ and $\bar{\mathbf{g}}_j$ in the right hand-side of (17) can be calculated in a simple manner from previously determined $\bar{\mathbf{f}}_k$, $\bar{\mathbf{g}}_k$, $\bar{\mathbf{f}}_l$ and $\bar{\mathbf{g}}_l$.

This strategy was originally proposed in [12], based on aspects of CB also mentioned in [9]. It is described in pseudo-code in Table VII, where the inputs are the non-redundant Taylor expansion blocks of (13). For notational simplicity, the bars over $\mathbf{F}$, $\mathbf{G}$, $\mathbf{f}$ and $\mathbf{g}$ are dropped and $\mathbf{b}$ is aggregated to $\mathbf{G}$ by means of $\mathbf{G}_{1,0}$. The shift operation $\underline{\times}$ is efficiently performed with an easily pre-computed lookup table for mapping the multi-index of each product of state variables to its corresponding position in $\bar{\mathbf{x}}_\otimes$ [60]. Such map and its inverse, denoted by $I(\cdot)$ and $I^{-1}(\cdot)$, respectively, are also used in lines 6, 7 and 8 of Table VII.

The computation of each pth-order row of $\bar{\mathbf{F}}$ in Table VII requires $\bar{M}_P + 1 - \bar{M}_{p-1} - 1 = \bar{M}_P - \bar{M}_{p-2} - \bar{m}_{p-1}$ additions, with $\bar{M}_P = \sum_{p=1}^{P} \bar{m}_p$. This follows from the first $\bar{M}_{p-1}+1$ elements of the each term involved in the sum of line 9 being always null, since the Taylor expansion (whether redundant or non-redundant) of $(x_{j_1} \ldots x_{j_p})'$ does not contain terms of order lower than p.[8] Similarly, the computation of each pth-order row of $\bar{\mathbf{G}}$ requires $\bar{M}_{P-1} - \bar{M}_{p-2}$ additions. Hence, as there are $\bar{m}_p$ rows of order p, the total number of required additions is

$$\bar{\mathcal{A}}(m,P) = \sum_{p=2}^{P} \bar{m}_p(\bar{M}_P + \bar{M}_{P-1} - 2\bar{M}_{p-2} - \bar{m}_{p-1}), \quad (20)$$

[8]This can be more easily seen in (12) and is guaranteed here by the application of the $\underline{\times}$ operator, which shifts the elements of previous rows $\bar{\mathbf{f}}_k$ and $\bar{\mathbf{f}}_l$ to the right.

TABLE IX
USING SPARSITY, NUMBER OF ADDITIONS (TOP) AND EXECUTION TIME IN SECONDS (BOTTOM) FOR SETUP (B).

Direct CB		Non-redundant CB	
$P=4$	$P=5$	$P=4$	$P=5$
4.3×10^{4}	6.4×10^{5}	2.6×10^{3}	9.4×10^{3}
5.6×10^{-2}	1.3	9.4×10^{-3}	9.0×10^{-2}

which is exemplified in Table VIII.

The algorithm in Table VII doesn't lend itself to vectorization as much as in the direct case and, thus, a significant reduction in execution time is attained only for sufficiently large values of m and P. In particular, for setup (B) gains of 22 and 542 times over the results in Table VI are attained for $P = 4$ and $P = 5$, respectively. Also, returning to the results in Tables I and IV, we see that the non-redundant CB doesn't dominate the total execution time as much as the direct CB does. Finally, regarding memory requirements, $\bar{\mathbf{F}}$ and $\bar{\mathbf{G}}$ have $\bar{M}_P^2$ elements, which, for large m and P, is much smaller than the M_P^2 elements of $\mathbf{F}$ and $\mathbf{G}$.

G. Using sparsity in Carleman Bilinearization

When $\bar{\mathbf{F}}_{1,p}$ and $\bar{\mathbf{G}}_{1,p}$ are sparse, most of the additions in lines 9 and 10 of Table VII are unnecessary, since at least one of the added terms is zero. Let $\bar{\mathbf{f}}_j^{\mathcal{C}}$ contain the nonzero elements of $\bar{\mathbf{f}}_j$ and let $\bar{\mathbf{f}}_j^{\mathcal{I}}$ contain their corresponding indices. Then, to carry out line 9 (line 10 being totally analogous), initially the assignment $\bar{\mathbf{f}}_j \leftarrow l\underline{\times}\bar{\mathbf{f}}_k$ is performed by

$$\bar{\mathbf{f}}_j^{\mathcal{I}} \leftarrow (l\underline{\times}\bar{\mathbf{f}}_k)^{\mathcal{I}} \ \text{ and } \ \bar{\mathbf{f}}_j^{\mathcal{C}} \leftarrow \bar{\mathbf{f}}_k^{\mathcal{C}},$$

where $(l\underline{\times}\bar{\mathbf{f}}_k)^{\mathcal{I}}$ indicates that each element of $\bar{\mathbf{f}}_k^{\mathcal{I}}$ is mapped to its corresponding shifted position, according to l. Next, $\bar{\mathbf{f}}_j \leftarrow \bar{\mathbf{f}}_j + k\underline{\times}\bar{\mathbf{f}}_l$ is performed, where adding any α to the cth column $\bar{f}_{j,c}$ of $\bar{\mathbf{f}}_j$ amounts to performing either the sum

$$\bar{f}_{j,n_c}^{\mathcal{C}} \leftarrow \bar{f}_{j,n_c}^{\mathcal{C}} + \alpha, \quad \text{if } \exists n_c \ \text{ such that } \bar{f}_{j,n_c}^{\mathcal{I}} = c, \qquad (21)$$

or, otherwise, the concatenation

$$\bar{\mathbf{f}}_j^{\mathcal{C}} \leftarrow [\bar{\mathbf{f}}_j^{\mathcal{C}}\ \alpha] \ \text{ and } \ \bar{\mathbf{f}}_j^{\mathcal{I}} \leftarrow [\bar{\mathbf{f}}_j^{\mathcal{I}}\ c]. \qquad (22)$$

For the implementation of this procedure, a binary tree [63, p. 139] with nodes given by the elements of $\bar{\mathbf{f}}_j^{\mathcal{I}}$ is built for each $\bar{\mathbf{f}}_j$. Due to the native support for binary tree search in C, the search for an index c in $\bar{\mathbf{f}}_j^{\mathcal{I}}$ can be carried out very efficiently, accelerating operations (21)–(22). Table IX shows the number of additions (*i.e.* executions of (21)) and the execution times of this implementation. To save space, we present only the results for setup (B) with $P = 4$ and 5. Gains of 38 and 27 times in speed over the results in Table VIII were attained for $P = 4$ and 5, respectively. In an extension of our work, further gains could possibly result in the mPM case from not evaluating $\bar{\mathbf{F}}_{1,p}$ in the Taylor step, keeping $\bar{\mathbf{f}}_j^{\mathcal{C}}$ in string form throughout (21)–(22) and evaluating it only at the end.

We now sketch a procedure for using sparsity to construct the direct CB of (12). From (9), $\mathbf{F}_{q,p}$ depends only on $\mathbf{F}_{1,p-q+1}$, so we can iterate over each row of the latter, "propagating" each nonzero element to the appropriate positions in

TABLE X
EXECUTION TIMES (SEC) FOR n SETS OF PM PARAMETER VALUES (TOP) AND FOR $n = 100$ (BOTTOM), SETUP (B) AND $P = 5$.

NE†	LE-NR‡	LE-NR-S§
$33n + 1300n$	$6.8 + 0.33n + 2.4n$	$6.6 + 0.017n + 0.09n$
1.3×10^5	280	17.3

†: Numerical expansion and direct CB, ‡: literal non-redundant expansion/ evaluation and non-redundant CB, §: same, using sparsity

$\mathbf{F}_{q,p}$, which follow from the structure of the terms in (9) . This is quickly done using the vectors $\mathbf{f}_i^{\mathcal{I}}$ of the rows of $\mathbf{F}_{1,p-q+1}$. The rows of $\mathbf{F}_{q,p}$ are also stored in sparse representations, so that adding an element to $\mathbf{F}_{q,p}$ is carried out similarly to (21)–(22). As in the non-redundant case, the implementation of this procedure is based on binary trees, the results being shown in Table IX. Comparing the results with those of Table VI, it can be seen that using sparsity had a very large impact, namely gains of 141 and 1000 times in speed, for $P = 4$ and $P = 5$, respectively. Even so, however, direct CB is still 6 and 14 times slower, respectively, than non-redundant CB.

H. Comparison of overall execution times

We adopt as the baseline overall execution time that of the numerical Taylor expansion (Section III-B) followed by the direct CB (Section III-E), in spite of their already efficient implementation. Next, we consider the literal non-redundant Taylor expansion (Section III-D) followed by the non-redundant CB (Section III-F) and, finally, the use of sparsity in both these procedures (Sections III-D and III-G). Overall execution times for setup (B), $P = 5$ and n sets of PM parameter values are given in Table X as the sum of the contributions of each step. As discussed in Section III-B, only the numerical evaluation step of CB must be executed n times. The results of those sums for $n = 100$ is also shown. As can be seen, exploring redundancy and sparsity leads to reducing execution time by a factor of 7514 in this case.

We note that in our previous work [12] the non-redundant CB was preceded by the numerical Taylor expansion, which leads (see Sections III-D and III-F) to an overall execution time of $3.1n+2.4n$ seconds in the same conditions as above. Thus, from Table X, our introduction of the literal Taylor expansion and the sparsity oriented procedures leads, for $n = 100$, to reducing execution time by a factor of $550/17.3 = 31.8$ in relation to our previous work.

IV. KERNEL COMPUTATION

After bilinearization (CB) of the PM, (6) provides the reference Volterra kernels. We consider in this section the most usual computation of $\{h_p\}$ over a regular grid, yielding a Volterra filter. However, our routines can be adapted to other ends, *e.g.* when keeping only a small number of diagonals around the main diagonal of each kernel [17]. We note that, unlike in [12], we compute the kernels directly in the time domain and exploit a low-rank decomposition of $\mathbf{G}$ for speeding up this computation. As a result, our proposed procedure is orders of magnitude faster than that developed in [12]. For notation simplicity, we drop the bars over $\mathbf{F}, \mathbf{G}$, $\mathbf{b}$ and $M_P = \sum_{p=1}^{P} \bar{m}_p$, although the non-redundant CB of the PM will, most likely, have been employed.

TABLE XI
ORDER-RECURSIVE KERNEL COMPUTATION.

Inputs: $\mathbf{F}$, $\mathbf{G}$, $\mathbf{b}$, $\mathbf{c}$, N
Outputs: h_p, $p = 1, \ldots, P$
$e_F(0) \leftarrow \mathbf{I}, \quad e_F(1) \leftarrow e^{\mathbf{F}T_s}, \quad e_T \leftarrow e^{\mathbf{F}T_s}$
for $k = 2$ **to** $N-1$ **do**
  $e_F(k) \leftarrow e_F(k-1)e_T$
end for
for $k_1 = 0$ **to** $N-1$ **do**
  $\mathbf{v}_1 \leftarrow \mathbf{c}^T e_F(k_1)$
  $h(k_1) \leftarrow \mathbf{v}_1\mathbf{b}$
  $\mathbf{v}_1 \leftarrow \mathbf{v}_1\mathbf{G}$
  for $k_2 = 0$ **to** $N-1$ **do**
    $\mathbf{v}_2 \leftarrow \mathbf{v}_1 e_F(k_2 - k_1)$
    $h(k_1, k_2) \leftarrow \mathbf{v}_2\mathbf{b}$
    $\mathbf{v}_2 \leftarrow \mathbf{v}_2\mathbf{G}$
    for $k_3 = 0$ **to** $N-1$ **do**
      $h(k_1, k_2, k_3) \leftarrow \mathbf{v}_2 e_F(k_3 - k_2)\mathbf{b}$
    end for
  end for
end for

A. Recursive calculation in the kernel order

Adopting a regular grid $\{\boldsymbol{k}T_s\}$ on the triangular domain $k_1 \leq \cdots \leq k_p \leq N-1$, with the same sampling period T_s for all p, allows computations used for obtaining h_{p-1} to be reused for h_p. Indeed, sampling (6) on this grid, we can write $h_p(\boldsymbol{k}T_s) = \mathbf{v}_p\mathbf{b}$, where $\mathbf{v}_1 = \mathbf{c}^T e_F(k_1)$, $e_F(k) \triangleq e^{kT_s\mathbf{F}}$ and $\mathbf{v}_q = \mathbf{v}_{q-1}\mathbf{G}e_F(k_q - k_{q-1})$ for $1 < q \leq p$.

Table XI contains the pseudocode of this method for $P = 3$, the extension to $P > 3$ being trivial. In the first stage, $e_F(0), \ldots, e_F(N-1)$ are computed, with $e_F(1)$ requiring $\mathcal{O}(M_P^3)$ multiplications and each $e_F(k)$, $k > 1$, additional M_P^3 multiplications, giving thus a total cost

$$\mathcal{C}_e = \mathcal{O}(\bar{M}_P^3) + (N-2)\bar{M}_P^3. \tag{23}$$

In the second stage, the l_p-element kernels h_p are computed, $l_p = \binom{N+p-1}{p}$. The operations $\mathbf{v}_{p-1}\mathbf{G}$ (for $p > 1$), $\mathbf{c}^T e_{\mathbf{F}}$ or $\mathbf{v}_{p-1}e_{\mathbf{F}}$ and $\mathbf{v}_p\mathbf{b}$ require $l_{p-1}M_P^2$, $l_pM_P^2$ and l_pM_P multiplications, respectively, giving a total cost

$$\mathcal{C}_h = L_P M_P + (L_P + L_{P-1})M_P^2, \tag{24}$$

where $L_p = \sum_{q=1}^{p} l_q$. Table XII shows some values of P and L_P, the notation xey standing as usual for $x \times 10^y$.

The tradeoff related to matrices $e_F(k)$ should be noted: not pre-computing them would save $\mathcal{O}(NM_P^2)$ memory positions while making the computational cost of the second stage about M_P times larger. We note also that in Table XI multiple indices such as (k_1, k_2) are used only for notational simplicity. In fact, since the kernels are in triangular form, storing sequentially each one in a vector $\mathbf{h}_p \in \mathbb{R}^{l_p}$ requires less memory.

B. Exploiting rank-retaining decomposition of $\mathbf{G}$

As the last $\bar{m}_P$ columns of $\mathbf{G}$ are null, $\text{rank}(\mathbf{G}) \leq M_{P-1}$ and thus $\mathbf{G}$ admits a rank-retaining decomposition (RRD) of the form $\mathbf{G} = \mathbf{A}\mathbf{B}^T$, with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{M_P \times M_{P-1}}$. To reduce

TABLE XII
DIMENSION OF (NON-REDUNDANT) EXTENDED STATE VECTOR AND NUMBER OF KERNEL COEFFICIENTS.

	M_P			
$m \setminus P$	2	3	4	5
4	14	34	69	125
11	77	363	1364	4367

	L_P			
$N \setminus P$	2	3	4	5
25	3.5e2	3.3e3	2.4e4	1.4e5
100	5.2e3	1.8e5	4.6e6	9.7e7

computing cost, from (6) and with $d_m = k_m - k_{m-1}$ we write for $p > 1$

$$h_p(\boldsymbol{k}T_s) = e_A(k_1)e_{AB}(d_2)\dots e_{AB}(d_{p-1})e_B(d_p), \qquad (25)$$

where $e_A(.) = \mathbf{c}^T e_F(.)\mathbf{A}$, $e_{AB}(.) = \mathbf{B}^T e_F(.)\mathbf{A}$ and $e_B(.) = \mathbf{B}^T e_F(.)\mathbf{b}$. These are pre-computed up to $k = N-1$ by

$$\begin{aligned} \mathbf{c}_e(k) &\leftarrow \mathbf{c}_e(k-1)e_F(1), \quad e_A(k) \leftarrow \mathbf{c}_e(k)\mathbf{A}, \\ \mathbf{B}_e(k) &\leftarrow \mathbf{B}_e(k-1)e_F(1), \quad e_{AB}(k) \leftarrow \mathbf{B}_e(k)\mathbf{A}, \\ \mathbf{b}_e(k) &\leftarrow e_F(1)\mathbf{b}_e(k-1), \quad e_B(k) \leftarrow \mathbf{B}^T\mathbf{b}_e(k), \end{aligned}$$

with $\mathbf{c}_e(0) = \mathbf{c}^T$, $\mathbf{B}_e(0) = \mathbf{B}^T$ and $\mathbf{b}_e(0) = \mathbf{b}$. This requires

$$\mathcal{C}_e^r = \mathcal{O}(M_P^3) + NM_P(1 + 2M_{P-1} + 2M_P + M_{P-1}^2 + M_{P-1}M_P)$$

multiplications, where the first term encompasses the cost of $e_F(1)$ and the RRD $\mathbf{G} = \mathbf{A}\mathbf{B}^T$. This represents a reduction of roughly $\mathcal{O}(M_P/M_{P-1})$ with respect to $\mathcal{C}_e$ in (23).

Now, (25) is implemented analogously to Table XI but with less costly matrix products, since $e_{AB}(k) \in \mathbb{R}^{M_{P-1}\times M_{P-1}}$, whereas $\mathbf{G}, e_F(k) \in \mathbb{R}^{M_P \times M_P}$. It requires thus

$$C_h^r = (L_P - l_1)M_{P-1} + (L_{P-1} - l_1)M_{P-1}^2$$

multiplications, meaning a cost reduction of roughly $L_P M_P^2 / L_{P-1} M_{P-1}^2$ relatively to $\mathcal{C}_h$ in (24).

C. Use of developed MEX functions

Faster execution of the nested loops in recursive kernel computation can be attained using a MEX function. It is important that it is written using the basic linear algebra subprograms (BLAS) library for matrix-matrix and matrix-vector products. In particular, BLAS can take advantage of the availability of multiple processors or cores to automatically parallelize these operations when the involved matrices have sufficiently large dimensions.

D. Numerical comparison

We now compare the execution times of the above presented methods when applied to setups (A) and (B) described in Section III-A. The RRD used in the procedure of Sec. IV-B is the economical singular value decomposition (SVD) $\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, by setting $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}^{1/2}$ and $\mathbf{B} = \mathbf{V}\mathbf{\Sigma}^{1/2}$, which has cost $\mathcal{O}(M_P^3)$, where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{M_P \times M_{P-1}}$ and $\mathbf{\Sigma} \in \mathbb{R}^{M_{P-1}\times M_{P-1}}$.

The measured times for $P \in \{2,3,4,5\}$ and $N \in \{25, 100\}$ are shown in Table XIII. For setup (B), the matrix products were automatically parallelized across multiple cores when the operands were sufficiently large. One can see that the cost reduction delivered by RRD grows with P, as expected. In particular, it attains a factor of $4978/50 \approx 100$ for $P = 4$ and $N = 100$ in setup (B). Furthermore, the use of the developed MEX functions can significantly bring down execution time. For instance, a reduction of $212/15 \approx 14$ is achieved for $P = 5$ and $N = 100$ in setup (A). We note that the combined used of the strategies discussed in Sections IV-B and IV-C yields a remarkable overall computing time reduction. For instance, of $4978/27 \approx 184$ for $P = 4$ and $N = 100$ in setup (B).

TABLE XIII
ROUTINES OF SECTIONS IV-A, IV-B AND IV-C: EXECUTION TIMES (SEC)

Setup (A)

	Recursive (IV-A)		Recursive+RRD (IV-B)		Rec.+RRD+mex (IV-C)	
P	$N = 100$	$N = 25$	$N = 100$	$N = 25$	$N = 100$	$N = 25$
2	1.9e-2	1.9e-3	1.3e-2	1.8e-3	2.7e-3	1.4e-3
3	0.70	1.8e-2	0.36	1.1e-2	1.2e-2	3.9e-3
4	29	0.17	9.6	7.4e-2	0.28	1.2e-2
5	2922	4.7	212	0.43	15	8.7e-2

Setup (B)

	Recursive (IV-A)		Recursive+RRD (IV-B)		Rec.+RRD+mex (IV-C)	
P	$N = 100$	$N = 25$	$N = 100$	$N = 25$	$N = 100$	$N = 25$
2	4.8e-2	6.63e-3	2.3e-2	6.0e-3	1.2e-2	5.5e-3
3	$18^\dagger$	$0.47^\dagger$	0.62	0.14	0.26	0.14
4	$4978^\dagger$	$36^\dagger$	$50^\dagger$	$7.3^\dagger$	$27^\dagger$	$7.2^\dagger$
5	‡	$1874^\dagger$	$5577^\dagger$	$285^\dagger$	$5090^\dagger$	$284^\dagger$

† : automatic usage of multiple cores. ‡ : computationally impractical.

E. Comparison with other approaches

Deriving Volterra kernels by means of CB has advantages over other approaches such as the growing exponential approach (GEA) and the variational equation approach (VEA) [4]. Namely, CB is easier to implement and more versatile than both GEA and VEA, since it can be used to directly compute kernels or GFRFs. Despite the relative popularity of Volterra models, the numerical computation of kernels from state-space equations has received little attention in the literature. In [51], [52], a procedure based on multi-indexes relying on GEA is proposed. It should be noted that the considered PM is more general, involving a differential equation that can be nonlinear in the input. However, the method is more complex than ours and requires more symbolic processing. It should also be noted that a somewhat more general form can be considered by replacing $u(t)$ by $h(u(t))$ in (4), where $h(.)$ is a nonlinearity. This corresponds then to the cascade of a system $\mathcal{S}_1$ defined by $v(t) = h(u(t))$ with a linear-analytic system $\mathcal{S}_2$ like (4) with $v(t)$ as input. In several of the applications of our method discussed in the Introduction and in Section VI, it would simply be a matter of cascading $\mathcal{S}_1$ with the suitable approximation of $\mathcal{S}_2$ derived using our method.

In comparison with the frequency domain method of our previous work [12], the time domain methods presented here are much more efficient, due to the strategies presented in Sections IV-A to IV-C. With them, for instance, for $P = 4$, $N = 100$ and setup (B) the kernels take only 27 seconds to compute (*cf.* Table XIII). If the procedure of [12] is employed instead, then the filters of the setup can be incorporated during

the GFRFs computation, and thus the starting point is a PM of order $m = 3$, instead of $m = 11$. Even so, the kernels still take much longer to compute: 10612 seconds (about 393 times more). Furthermore, although directly sampling the kernels in the time domain introduces aliasing [12], they are exact when employed to model a typical structure found in digital signal processing applications, which is described next.

V. Discrete-time kernels

The efficient computation of bilinear approximations and Volterra kernels presented in the preceding sections is now employed to obtain and apply reference discrete-time Volterra kernels in the context of signal processing.

A. Overall setup

In many problems, such as acoustic echo cancellation [18], equalization [20], closed-loop linearization [22], [23] and active noise control [14], input and output signals of a nonlinear system originate and end in discrete-time, as in Figure 1, which was discussed in Section III-A.[9] This configuration is directly found, for instance, in acoustic echo cancellation: the nonlinear adaptive filter aims to approximate a discrete-time version of the kernels of V, where V_o comprises the amplifier, the loudspeaker and the room acoustic response. The same configuration also appears implicitly in other contexts such as nonlinear model predictive control [64], where, in particular, the filter H_r may incorporate the dynamics of some actuator.

B. Reference discrete-time Volterra kernels

By augmenting the state-space physical model (PM) of the nonlinear system V_o with filters H_r and H_a, a bilinear approximation of the overall system V in Figure 1 can be computed with the CB procedures of Section III (this applies in particular to our setup example (B)). The triangular kernels $h_p(\boldsymbol{k}T_s)$ of V sampled on a regular grid can then be computed as described in Section IV. Now, when $D(z) \equiv 1$, from (1), the pth order component of $y(n)$ is given by

$$y_p(n) = \int_{\mathbb{R}_+^p} h_p(\boldsymbol{\tau}) \prod_{i=1}^{p} u(nT_s - \tau_i) d\boldsymbol{\tau}. \tag{26}$$

Assuming an ideal D/A converter and replacing its output $u(t) = \sum_{k=-\infty}^{\infty} u(k)\delta(t - kT_s)$ in (26) leads to

$$y_p(n) = \sum_{k_1=0}^{\infty} \sum_{k_2=k_1}^{\infty} \dots \sum_{k_p=k_{p-1}}^{\infty} v_p(\boldsymbol{k}) \prod_{i=1}^{p} u(n - k_i), \tag{27}$$

where the discrete-time kernels are

$$v_p(\boldsymbol{k}) \triangleq \frac{1}{m_1! \dots m_q!} h_p(\boldsymbol{k}T_s), \tag{28}$$

with q denoting the number of distinct indexes among $k_1, \dots, k_p$ and $m_1, \dots, m_q$ denoting their corresponding numbers of occurrences [65]. In practice, this procedure can be applied in the case of a zero-order hold (ZOH) D/A converter followed by a correction filter with response $\text{sinc}^{-1}(fT_s)$ up to the stopband edge of the reconstruction filter H_r.

[9] In particular, we recall that $D(z)$ is used to obtain a ZOH D/A behaviour.

C. Uncorrected ZOH D/A converter

More generally, a ZOH D/A converter can be modeled by cascading the differentiator $D(z) = 1 - z^{-1}$ and the ideal D/A converter and augmenting V with an integrator at its input [66]. Using the formal series $V_p(\boldsymbol{z}) = \sum_{k_1=0}^{\infty} \dots \sum_{k_p=k_{p-1}}^{\infty} v_p(\boldsymbol{k}) z_1^{-k_1} \dots z_p^{-k_p}$ it can be verified that $V_p(\boldsymbol{z}) = D(z_1)D(z_2)\dots D(z_p)V_p^I(\boldsymbol{z})$, where $V_p^I(\boldsymbol{z})$ corresponds to the integrator-augmented system. From this a computational procedure to determine the kernels $v_p(\boldsymbol{k})$ from the computed kernels $v_p^I(\boldsymbol{k})$ can be obtained. Setup example (A) falls into this case.

VI. Applications in signal processing

With the procedures we have described, Volterra and bilinear models can be efficiently computed from a reference PM. Their application is discussed in the following, extending the discussion in the Introduction. Sections VI-A to VI-D are connected whereas the remaining subsections are more independent. It should be noted that the use of a state-space structure based on the PM for applications such as nonlinear acoustic echo cancelling and predistortion, discussed below, would have the problems discussed in Section II-C.

A. Overview of systems for attenuation of nonlinearities

Nonlinear acoustic echo cancellation (NLAEC) [18] and linearization [22]–[24] are examples of the goals of such systems. Usually, their design requires 1) choosing strucutural parameters such as nonlinear order and memory length, and either 2) establishing values for the set of parameters defined by the structural parameters, or 3) choosing an adaptive algorithm to do so in the course of operation. The conventional approach is to use measured or simulated input/output signals of the nonlinear system to guide those design tasks in an iterative manner. An initial choice of structural parameters is made and then the non-structural parameters are identified using the input/output signals. Depending on the resulting modeling error, another choice of structural parameters is made. However, it may be not clear whether the modeling error is due to an inadequate choice of structural parameteres or to a deficiency of the identification algorithm, which, besides, may also be under development. Using a reference model as we propose makes this easier, since it provides a realistic *a priori* guide to the choice of structural parameters and also an initial point for refining parameter values. The choice of structural parameters in the context of NLAEC is exemplified in some detail in Section VI-B, and is discussed in Section VI-C for linearization by predistortion. In particular, when using simulations instead of measurements, reference Volterra or bilinear models can, additionally, allow much faster simulations than the PM itself, as seen in Section VI-D.

B. Selection of structural parameters for NLAEC

With the reference discrete-time kernels, the parameters of a related non-linear processing structure can be chosen. Typically this involves computing its output $y(n)$ and (with a numerical integration routine) the output $y_{\text{PM}}(n)$ of the PM,

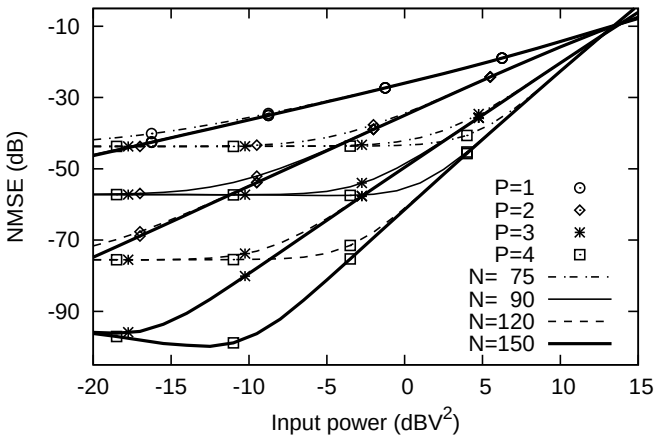

Fig. 2. Loudspeaker modeling: normalized MSE for Volterra Filter of order P and memory length N.

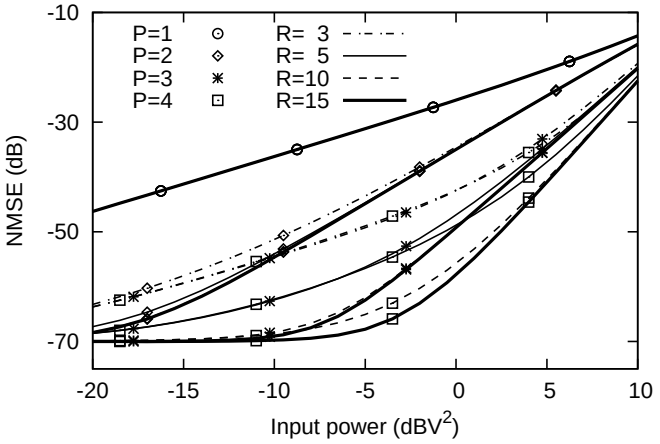

Fig. 3. Loudspeaker modeling: normalized MSE for Volterra-CPD of order P, rank R and memory length $N = 100$.

for different realizations of a random process that models the input $u(n)$ in the application of interest.

As an example, in the context of non-linear acoustic echo cancellation (NLAEC), we consider setup (A) from Section III-A, which corresponds to $H_r(z) = H_a(z) \equiv 1$ in Fig. 1. For such task, we initially evaluate the Volterra Filter (VF) given by the discrete-time kernels $v_p(\boldsymbol{k})$ for some order P and memory length N. Figure 2 shows the normalized MSE (NMSE) $||y(n) - y_{\mathrm{PM}}(n)||^2/||y_{\mathrm{PM}}(n)||^2$ for 100 realizations of 500 samples of a Gaussian input signal with band $\pi/4$ and sampling frequency 5 kHz. To obtain a range of input powers, the same 100 realizations were scaled accordingly.

It can be seen that if a NMSE smaller than, for instance, -60 dB were desired for inputs up to 0 dBV^2, one would then choose $P = 4$ and some $90 < N < 120$. For $N = 120$, such a VF would have $L_P = 9.4 \times 10^6$ coefficients, which is too large in practice. We note also that since the VF is a polynomial system and the loudspeaker is not, for high input powers the NMSE actually gets worse as P grows. Still in the NLAEC setting, we now evaluate the Volterra-CPD[10] structure [15], which for a given NMSE tends to have much less coefficients than the VF, provided the (symmetric) kernels admit low-rank approximations (but is no longer linear in the parameters). We employed Tensorlab 3.0 [67] to decompose each symmetric higher-order kernel. For $p = 2$, the truncated SVD was used. The resulting NMSE, for $N = 100$ and in the same conditions as above, is shown in Fig. 3. A NMSE smaller than -60 dB for inputs smaller that 0 dBV^2 would now require $P = 4$ and $R = 15$. The resulting number of coefficients, $[1+(P-1)R]N = 4600$, is 3 orders of magnitude smaller than that of the VF.

C. Selection of structural parameters for predistortion

One common approach for attenuating nonlinear distortions of a system is by predistorting its input so that the overall output is (approximately) linear. In particular, a well-developed theory exists for linearizing a Volterra system up to a certain order P [2], [6], [24].[11] In the design of a predistortion system, choosing structural parameters could be made easier by using the reference kernels obtained from a PM of the system to be linearized.

D. Simulation of nonlinear systems

Structural assessments as in Section VI-B can be also used to characterize speed *vs.* accuracy tradeoffs in the simulation of nonlinear systems. The range of such tradeoff is exemplified by the fact that, for the results above, Matlab's integration routine `ode133` (with default precision) took around 6 hours to compute outputs $y_{\mathrm{PM}}(n)$ for the set of 100 realizations and 25 scales of the input. On the other hand, the VF with $P = 4$, $N = 120$ and a diagonal representation of the kernels [6, p. 414] and the Volterra-CPD with $N = 100$, $P = 4$ and $R = 15$ took only 21 minutes and 0.88 seconds, respectively. Both (but not `ode133`), moreover, are little affected by the number of scales, since the output factors $s^p y_p(n)$ for input $su(n)$ are given directly from the factors $y_p(n)$ for input $u(n)$.

E. Reduced-order bilinear modeling

One utility of the CB is providing a bilinear approximation of the reference PM. Though the order of the resulting bilinear model grows rapidly with the nonlinearity order P, one can employ order reduction techniques such as that of [68] to obtain a more compact model, whose use can be significantly less costly in comparison with the CB's output. We can expect this indirect procedure to be algebraically simpler than trying to obtain a good reduced-order bilinear approximation directly from the reference PM.

F. Calculation of nonlinear distortion

For design purposes, it may be of interest to study how certain components affect the nonlinear distortions introduced

[10]From *Canonical Polyadic Decomposition*, also referred to as PARAFAC decomposition, from *Parallel Factors*.

[11]This approach requires that the inverse of the linear subsystem be stable and causal [2].

by a given device. Assuming a known reference PM involves parameters which quantify properties of these components, one can compute reference kernels for particular parameter values and then measure the resulting harmonic and intermodulation distortions from their associated GFRFs[12]. From the discussion in Section II-C and the example in Section VI-D, this approach can be faster than one based on directly simulating the reference PM.

G. Kernel-based fault diagnosis

Detecting abnormal operation of a mechanical system is an important application. One possible approach is to classify its operation based on features extracted from identified Volterra kernels [69]. If a reference PM is known, one can derive reference Volterra kernels corresponding to parameter configurations which characterize normal and abnormal operation regimes. These kernels can then be exploited to classify subsequently identified kernels of the operating system.

H. Evaluation of tensor modeling algorithms

Another interesting use of the reference kernels is in the evaluation of algorithms which estimate tensor models [70]. For instance, measuring the performance of CP decomposition algorithms using data tensors derived from a PM (instead of artificially generated ones) allows to realistically assess their potential in a practical application.

VII. Conclusion

Approximating a nonlinear system by a bilinear model is a useful technique for several purposes. Among them, the computation of associated Volterra kernels provides important information on the system's dynamics, allows assessing the parametric complexity of many candidate nonlinear models and can also be used for simulating its operation. The package of computational tools that we have offered in this paper allows an efficient computation of a bilinear approximation of a linear-analytic system, via Carleman's bilinearization method, and of the Volterra kernels associated with a bilinear representation. They include techniques which exploit sparsity and which perform a partially symbolic computation for an efficient computation for various sets of parameter values. To illustrate the usefulness of these contributions, reproducible examples included in this package were presented. Namely, we have calculated reference bilinear models and kernels of a loudspeaker model considering a typical DSP configuration possibly involving reconstruction and anti-aliasing filters. These kernels were then used for evaluating the suitability of a more compact tensor-based nonlinear model and also for efficiently simulating the system's operation under multiple inputs of same waveform but varying power levels.

[12]The spectral analysis for non-linear systems by means of GFRFs is studied in [53].

References

[1] N. Wiener, *Nonlinear problems in random theory*. Massachusetts Institute of Technology, 1958.

[2] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. John Wiley & Sons, 1980.

[3] S. A. Billings, "Identification of nonlinear systems–a survey," in *IEE Proceedings D-Control Theory and Applications*, vol. 127, no. 6. IET, 1980, pp. 272–285.

[4] W. J. Rugh, *Nonlinear System Theory: The Volterra/Wiener Approach*. Baltimore, MD: Johns Hopkins University Press, 1981.

[5] A. Isidori, *Nonlinear Control Systems*, 3rd ed. New York: Springer, 1995.

[6] V. J. Mathews and G. L. Sicuranza, *Polynomial signal processing*. Wiley, 2000.

[7] V. Marmarelis, *Nonlinear Dynamic Modeling of Physiological Systems*. Hoboken, NJ, USA: John Wiley & Sons, 2004.

[8] T. Ogunfunmi, *Adaptive Nonlinear System Identification: The Volterra and Wiener model approaches*. New York: Springer, 2007.

[9] R. W. Brockett, "Volterra series and geometric control theory," *Automatica*, vol. 12, no. 2, pp. 167–176, Mar. 1976.

[10] T. Koh and E. J. Powers, "Second-order volterra filtering and its application to nonlinear system identification," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 33, no. 6, pp. 1445–1455, 1985.

[11] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with Volterra series," *IEEE Transactions on Circuits and Systems*, vol. CAS-32, no. 11, pp. 1150–1161, Nov. 1985.

[12] J. H. de M. Goulart and P. M. S. Burt, "Efficient kernel computation for Volterra filter structure evaluation," *IEEE Signal Processing Letters*, vol. 19, no. 3, pp. 135–138, Mar. 2012.

[13] P. M. S. Burt and J. H. de M. Goulart, "Evaluating the potential of Volterra-PARAFAC IIR models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 5745–5749.

[14] A. Carini and G. L. Sicuranza, "A new class of FLANN filters with application to nonlinear active noise control," in *European Signal Processing Conference (EUSIPCO)*, Aug. 2012, pp. 1950–1954.

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

[16] A. Y. Kibangou and G. Favier, "Tensor analysis-based model structure determination and parameter estimation for block-oriented nonlinear systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 3, pp. 514–525, June 2010.

[17] M. Zeller and W. Kellermann, "Fast and Robust Adaptation of DFT-Domain Volterra Filters in Diagonal Coordinates Using Iterated Coefficient Updates," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1589–1604, mar. 2010.

[18] L. A. Azpicueta-Ruiz, M. Zeller, A. R. Figueiras-Vidal, J. Arenas-García, and W. Kellermann, "Adaptive Combination of Volterra Kernels and its Application to Nonlinear Acoustic Echo Cancellation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 1, pp. 97–110, Jan. 2011.

[19] C. Crespo-Cadenas, J. Reina-Tosina, M. J. Madero-Ayora, and J. Muñoz Cruzado, "A new approach to pruning volterra models for power amplifiers," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2113–2120, Apr. 2010.

[20] B. F. Beidas, "Intermodulation Distortion in Multicarrier Satellite Systems: Analysis and Turbo Volterra Equalization," *IEEE Transactions on Communications*, vol. 59, no. 6, pp. 1580–1590, June 2011.

[21] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. Signal Process.*, vol. 49, no. 8, pp. 1667–1676, 2001.

[22] K. Shi and A. Redfern, "Blind Volterra system linearization with applications to post compensation of ADC nonlinearities," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 3581–3584.

[23] D. Franken, K. Meerkotter, and J. Waßmuth, "Observer-based feedback linearization of dynamic loudspeakers with AC amplifiers," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, pp. 233–242, Mar. 2005.

[24] A. Carini, G. Sicuranza, and V. J. Mathews, "Equalization and linearization of nonlinear systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998, pp. 1617–1620.

[25] A. Germani, C. Manes, and P. Palumbo, "Filtering of stochastic nonlinear differential systems via a Carleman approximation approach," *IEEE Trans. on Autom. Control*, vol. 52, no. 11, pp. 2166–2172, Nov. 2007.
[26] J. Arroyo, E. Barocio, R. Betancourt, and A. R. Messina, "A bilinear analysis technique for detection and quantification of nonlinear modal interaction in power systems," in *IEEE Power Engineering Society General Meeting*. IEEE, 2006.
[27] T. Mukherjee, G. K. Fedder, D. Ramaswamy, and J. White, "Emerging simulation approaches for micromachined devices," *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1572–1589, Dec. 2000.
[28] C. Fager, J. C. Pedro, N. B. de Carvalho, and H. Zirath, "Prediction of IMD in LDMOS transistor amplifiers using a new large-singal model," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 12, pp. 2834–2842, Dec. 2002.
[29] S. A. Maas, *Nonlinear Microwave and RF circuits*. Artech House, 2003.
[30] A. Gothenberg and H. Tenhunen, "Performance analysis of sampling switches in voltage and frequency domains using volterra series," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2004.
[31] S. B. Kuntze, A. J. Zilkie, L. Pavel, and J. Aitchison, "Nonlinear State–Space Model of Semiconductor Optical Amplifiers With Gain Compression for System Design and Analysis," *Journal of Lightwave Technology*, vol. 26, no. 14, pp. 2274–2281, jul. 2008.
[32] H. Michalska, "Generic nonlinear stabilization of systems with matching algebraic structure," *Annual Reviews in Control*, vol. 35, pp. 215–234, 2011.
[33] M. Ekman, "Suboptimal control for the bilinear quadratic regulator problem: Application to the activated sludge process," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 1, pp. 162–168, Jan. 2005.
[34] A. Asma and A. Kamel, "Nonlinear system identification based on Volterra and Laguerre models," in *Proceedings of the 14th International Conference on Methods and Models in Automation and Robotics*, 2009, pp. 489–494.
[35] P. Chevalier, A. Oukaci, and J.-P. Delmas, "Third order widely nonlinear Volterra MVDR beamforming," in *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, may 2011, pp. 2648–2651.
[36] R. Boyer, R. Badeau, and G. Favier, "Fast orthogonal decomposition of Volterra cubic kernels using oblique unfolding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, may 2011, pp. 4080 –4083.
[37] E. L. O. Batista and R. Seara, "A fully LMS/NLMS adaptive scheme applied to sparse-interpolated volterra filters with removed boundary effect," *Signal Processing*, vol. 92, no. 10, pp. 2381 – 2393, 2012.
[38] R. R. Mohler, *Bilinear Control Process*. New York and London: Academic Press, 1973.
[39] C. Bruni, G. Dipillo, and G. Koch, "Bilinear systems: An appealing class of "nearly linear" systems in theory and applications," *IEEE Transactions on Automatic Control*, vol. 19, no. 4, pp. 334–348, 1974.
[40] R. R. Mohler, "An overview of bilinear system theory and applications," *IEEE Transanctions on Systems, Man, and Cybernetics*, vol. 10, pp. 683–688, Oct. 1980.
[41] D. L. Elliott, *Bilinear Control Systems*. New York: Springer, 2009.
[42] R. W. Brockett, "The early days on geometric nonlinear control," *Automatica*, vol. 50, pp. 2203–2224, 2014.
[43] V. J. Mathews and J. Lee, "Adaptive algorithms for bilinear filtering," *SPIE Proceedings*, vol. 2296, pp. 317–327, Oct. 1994.
[44] F. Fnaiech and L. Ljung, "Recursive identification of bilinear systems," *International Journal of Control*, vol. 45, no. 2, pp. 453–470, 1987.
[45] W. Favoreel, B. De Moor, and P. Van Overschee, "Subspace identification of bilinear systems subject to white inputs," *IEEE Transactions on Automatic Control*, vol. 44, no. 6, pp. 1157–1165, Jun. 1999.
[46] S. Gibson, A. Wills, and B. Ninness, "Maximum-likelihood parameter estimation of bilinear systems," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1581–1595, Oct. 2005.
[47] E. D. Sontag, Y. Wang, and A. Megretski, "Input classes for identifiability of bilinear systems," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 195–207, Feb. 2009.
[48] A. J. Krener, "Linearization and bilinearization of control systems," in *Allerton Conference on Circuit and System Theory*, 1974.
[49] J. Deutscher, "Asymptotically exact input-output linearization using Carleman linearization," in *European Control Conference (ECC)*, 2003.
[50] S. Irving and C. Joaquin, "On stabilization of non linear systems by using Carleman linearization and periodic systems theory," in *International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, 2011.
[51] A. Bauer and W. Schwarz, "Circuit analysis and optimization with automatically derived Volterra kernels," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1. IEEE, 2000, pp. 491–494.
[52] A. Bauer, "Efficient algorithms for the computation and application of Volterra kernels in the behavior analysis of nonlinear circuits and systems," in *IEEE European Conference on Circuit Theory and Design*, vol. 2, 2005.
[53] S. Billings and K. Tsang, "Spectral analysis for non-linear systems, part II: Interpretation of non-linear frequency response functions," *Mechanical Systems and Signal Processing*, vol. 3, no. 4, pp. 341–359, 1989.
[54] A. V. Balakrishnan, "On the controllability of nonlinear systems," *Proc. Nat. Acad. Sci. USA*, vol. 55, pp. 465–568, 1966.
[55] X. Zhao and V. Marmarelis, "On the relation between continuous and discrete nonlinear parametric models," *Automatica*, vol. 33, no. 1, pp. 81–84, 1997.
[56] J. Vlach and K. Singhal, *Computer methods of circuit analysis and design, 2nd ed.* New York: Van Nostrand, 1994.
[57] K. Meerktter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1989, pp. 720–723.
[58] e. a. Schn, T. B., "System identification of nonlinear state-space models," *Automatica*, vol. 47, pp. 39–49, 2011.
[59] A. J. Krener, "Bilinear and nonlinear realizations of input-output maps," *Siam Journal on Control*, vol. 13, no. 4, pp. 827–834, Jul. 1975.
[60] J. H. de M. Goulart, "Derivação eficiente e utilização de filtros de Volterra de referência na avaliação de formalismos não-lineares," Master's thesis (in portuguese), Escola Politécnica da Universidade de São Paulo, 2012.
[61] J. H. de M. Goulart and P. M. S. Burt. (2016, Dec.) Carleman bilinearization and Volterra kernels Toolbox. [Online] Available: http://www.lcs.poli.usp.br/∼phillip/sw/cbvk.
[62] D. Franken, K. Meerkotter, and J. Waßmuth, "Passive parametric modeling of dynamic loudspeakers," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 8, pp. 885–891, November 2001.
[63] B. W. Kernighan and D. M. Ritchie, *The C programming language, 2nd ed.* Prentice-Hall, 1988.
[64] A. Soni, "Control-relevant system identification using nonlinear Volterra and Volterra-Laguerre models," Ph.D. dissertation, University of Pittsburgh, 2006.
[65] P. M. S. Burt and J. H. de M. Goulart, "The Sampling of Triangular Kernels," *Online Supplementary Material*.
[66] H. A. Barker and S. Ambati, "Nonlinear sampled-data system analysis by multidimensional z transforms," *Proc. IEE*, vol. 119, no. 9, pp. 1407–1413, Sep. 1972.
[67] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016, Mar.) Tensorlab 3.0. [Online]. Available: http://www.tensorlab.net
[68] T. Breiten and T. Damm, "Krylov subspace methods for model order reduction of bilinear control systems," *Systems & Control Letters*, vol. 59, no. 8, pp. 443–450, 2010.
[69] H. Tang, Y. H. Liao, J. Y. Cao, and H. Xie, "Fault diagnosis approach based on Volterra models," *Mechanical Systems and Signal Processing*, vol. 24, no. 4, pp. 1099–1113, May 2010.
[70] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, p. 16, 2016.

Phillip M. S. Burt (M'85) received the bachelor and doctor degrees in Electrical Engineering from Escola Politécnica da Universidade de São Paulo, Brazil, where he is an Associated Professor. From 1984 to 1992 he worked in the telecommunications industry as a development engineer. In 2003 he spent a sabbatical year at the INT, Evry, France. His research interests include nonlinear signal processing, adaptive filtering and digital signal processing for telecommunications and audio.

José Henrique de M. Goulart received the B.Sc. degree in Computer Science from Universidade Federal de Sergipe, Brazil, in 2006, the M.Sc. degree in Electronic Systems from Escola Politécnica da Universidade de São Paulo, Brazil, in 2012, and the Ph.D. degree in signal processing from Université Nice Sophia Antipolis, France, in 2016. He is currently a post-doctoral researcher at GIPSA-Lab (Grenoble Laboratory of Image, Speech, Signal, and Automation), in France. His research interests include tensor models and decompositions, tensor methods in signal processing and nonlinear system modeling and identification.