Nonlinear Adaptive Algorithms on Rank-One Tensor Models
Felipe C. Pinheiro, Cassio G. Lopes, Senior Member, IEEE.

Abstract—This work proposes a low complexity nonlinearity model and develops adaptive algorithms over it. The model is based on the decomposable—or rank-one, in tensor language—Volterra kernels. It may also be described as a product of FIR filters, which explains its low-complexity. The rank-one model is also interesting because it comes from a well-posed problem in approximation theory. The paper uses such model in an estimation theory context to develop an exact gradient-type algorithm, from which adaptive algorithms such as the least mean squares (LMS) filter and its data-reuse version—the TRUE-LMS—are derived. Stability and convergence issues are addressed. The algorithms are then tested in simulations, which show its good performance when compared to other nonlinear processing algorithms in the literature.

Index Terms—Adaptive filtering, nonlinear signal processing, tensors, estimation theory.

I. INTRODUCTION

NONLINEAR signal processing has its uses whenever the performance of linear techniques start to become inadequate. A few situations that may ask for their use are echo cancellation [1], equalization of communication systems [2], acoustics [3], or broadband noise canceling [4].

While an important part of current signal processing research, an usual feature of nonlinear techniques is their high computational complexity, which sometimes render them unusable for some applications, such as those that use slower computers or require low power consumption. This complexity may be high even for modern computers, as is the case of the Volterra series [5], [6] and its exponentially increasing complexity.

The Volterra series is an important model because it is able to work with any continuous nonlinearity. It is also a very simple structure that can be described in a linear-in-the-parameters model. Given these advantages, various structures try to work with this model. But many times some restriction is necessary to allow for fast computation. Examples of these approaches are the filters with truncated diagonals [7], [8], which works with the central coefficients of the model, which are usually the most significant ones. Another kind of model is based in interpolation techniques [9], [10], in a way that the algorithms work with only a few coefficients while interpolating the others.

Another aspect of nonlinear signal processing is the difficulty in their analyses, specially if the models of nonlinearity are nonlinear in the parameters. This is the case with cascade structures [11]. Some problem of this kind is the lack of global asymptotic stability, with one example shown in this paper. Other models that have yet to be fully analyzed. An example is ones based in tensor decomposition [12].

Tensors are objects that can be accessed via many number of indexes. Vectors are one-index tensors, while matrices are two-index ones. Tensors are also a bridge between the multilinear [13] world—of multilinear functions, or functions of many variables that are linear in each one of them—and the linear world. As it will be shown in this paper, this property creates a conceptual link between the Volterra series and tensors, with these objects being the natural way of representing a Volterra kernel. Not surprisingly, tensors have indeed been used in nonlinear signal processing-related problem for quite some time [14], [15].

The tensor representation, and specially tensor rank decomposition [16] allows for a dramatic decrease in representational complexity of the model and, should this be exploited in the Volterra series, many low-complexity algorithms may be obtained. Of particular importance is the rank-one approximation [17]. This concept has seem some success in approximating linear responses [18], [19] with certain structures, and the possible gains to nonlinear signal processing are even more pronounced—exponential reductions in complexity can be achieved [20]. The other theoretical advantage of rank-one approximations is their well-posedness, which may not be the case for general low-rank decompositions [21].

This paper focuses on the development of the rank-one approximation of the Volterra series and on the results adaptive algorithms one can derive from the model. For this objective, the paper shall be organized as follows:

Therefore, the paper structure is as follows:
1) Introduction and notation.
2) A presentation on tensors and rank-one approximations.
3) The Volterra series and its relation to tensors. The decomposable (rank-one) model as a product of FIR linear filters, with low computational complexity.
4) Estimation theory of theory of the decomposable model.
5) The steepest descent algorithm as an iterative solution to the estimation problem.
6) LMS and TRUE-LMS algorithms as instantaneous approximations to the steepest descent. Stability issues and choice of parameters.
7) Simulations: testing performance, stability and applicability.
8) The presence of chaotic behavior.
9) Concluding remarks.

A. Notation
The notation of the paper follows the one on [22], while introducing new notation when necessary. The conventions are summarized as
• Scalars and vectors are represented by lowercase letters. (E.g. $x$ and $a$.)
• Time varying vectors are indexed as $x_i$, while time varying scalars are presented as $a(i)$. The time variable is always $i$ or $j$.
• Matrices and constants are represented as uppercase letters. (E.g. $R$ and $A$.)
• Tensors are represented as calligraphic letters. (E.g. $\mathcal{W}$ and $\mathcal{X}$.)
• Time-varying matrices and tensors follow the vector convention.
• Tensors by nature have a matrix representation. When interpreted as matrices, the involved operations are to be interpreted as the matrix equivalents. (E.g. the tensor product $\otimes$ is to be interpreted as the Kronecker product $\otimes$.)

This paper uses the classical convention of lower and upper indexes. Column vectors are indexed with an upper index (e.g. $(v)_i$) and matrices with both an upper one and a lower one (e.g. $(A)_i^j$). The convention for general tensors will be explained in through the paper.

II. TENSORS

Just like vectors are objects parameterized by a single index and matrices by two, tensors are objects parameterized by an arbitrary number $K$ of indexes—such number being called the order of the tensor. For example, an order $K$ tensor $T$ may be indexed as $(T)^{i_1 \cdots i_K}$. Such notions have been used by some developments in signal processing, usually dealing with multidimensional data. \[1\]

Just like vectors can alternatively characterized by the concept of a vector space, tensors can be abstractly defined as objects of an algebraic tensor product \[2\]. Given $K$ vector spaces $V_1, \ldots, V_K$, their tensor product is a new and bigger vector space $V_1 \otimes \cdots \otimes V_K$ together with a $K$-linear function $\otimes: V_1 \times \cdots \times V_K \to V_1 \otimes \cdots \otimes V_K$—called the tensor product of vectors—with the following universal property: any $K$-linear function $f: V_1, \ldots, V_K \to W$ to any vector space $W$ may be decomposed, uniquely, as $f = f \circ \otimes$, where $\tilde{f}: V_1 \otimes \cdots \otimes V_K \to W$ is a linear function that represents $f$ in tensor language. In other words, the tensor product makes it possible to represent $K$-linear functions, often unfamiliar, as linear functions, which are well understood. This can be summarized in the following commutative diagram.

\[ V_1 \times \cdots \times V_K \xrightarrow{\otimes} V_1 \otimes \cdots \otimes V_K
\]

The tensor product of the vectors $v_1 \in V_1, \ldots, v_k \in V_K$ is often represented as $v_1 \otimes \cdots \otimes v_k$, which is an order $K$ tensor. In computational terms, the tensor product can be implemented as a Kronecker product \[3\]. Any tensor that can be decomposed in a tensor product of vectors is called decomposable. It is a consequence of the definition of the algebraic tensor product that any tensor $T = V_1 \otimes \cdots \otimes V_K$ can be written as a sum of decomposable tensors, as in \[4\].

\[ T = \sum_{r=1}^{R} v_{1,r} \otimes \cdots \otimes v_{K,r}. \] \[5\]

There are many ways of writing a tensor this way, but the well-ordering principle implies that, for any given tensor, there exists a minimum number $R$ for which this is possible. This is called the rank of the tensor and will be represented as rank $T$.

The rank of the zero tensor is defined as zero, being the only tensor with such rank. Decomposable tensors have rank either one or zero. As an abuse of language, the paper shall refer to decomposable tensors also as rank-one tensors. This will not create confusion and is quite usual in tensor literature. When $K = 2$, the notion of tensor rank reduces to that of matrix rank.

Other property of interest is the dimension of the algebraic tensor product:

\[ \text{dim } V_1 \otimes \cdots \otimes V_K = (\text{dim } V_1) \cdots (\text{dim } V_K), \] \[6\]

that is, the dimension of the resulting vector space is the product of the dimensions of the individual vector spaces. This is consistent with the multi-index representation of tensors.

In classical tensor literature, lower and upper indexes in the coordinate representation are treated differently. This is to facilitate the introduction of operations similar to the product of matrices. Upper indexes are related to column vectors, in the way of $(v)^i$, while row vectors are represented by a lower index: $(w)_j$. A general tensor may be represented as $(T)^{i_1 \cdots i_K}$.

In this paper the following convention will be made: given tensors $T_1$ and $T_2$, indexed as $(T_1)^{i_1 \cdots i_L}$ and $(T_2)^{j_1 \cdots j_P}$, the product $T_1 T_2$ is defined by the coordinates

\[ (T_1 T_2)_{i_1 \cdots i_L}^{j_1 \cdots j_P} = \sum_{k_1 \cdots k_P} (T_1)_{i_1 \cdots i_L}^{k_1 \cdots k_P} (T_2)_{k_1 \cdots k_P}^{j_1 \cdots j_P}. \] \[7\]

Notice that this is defined only for when the number of lower indexes of $T_1$ equals the number of upper indexes of $T_2$. In particular, this convention implies, for the column vector $(v)^i$ and the row vector $(w)_j$, that the products $w v$ and $v w$ are, respectively, the inner and outer products of $v$ and $w$, which is consistent with matrix notation.

Another important operation is the tensor product of tensors, which takes the two tensors $(T_1)^{i_1 \cdots i_K}$ and $(T_2)^{k_1 \cdots k_Q}$ and result in a higher order tensor $T_1 \otimes T_2$ given by

\[ (T_1 \otimes T_2)_{i_1 \cdots i_K}^{j_1 \cdots j_P} = (T_1)_{i_1 \cdots i_K}^{j_1 \cdots j_P} (T_2)_{k_1 \cdots k_Q}^{k_1 \cdots k_Q}, \] \[8\]

that it, given by juxtaposition of their coordinates via multiplication.

A. Rank-one approximation

It is possible to naturally extend norms and inner products of vectors to corresponding norms of tensors. The details of how to compute them are presented on Appendix A. These notions allow for the proposition of approximation problems. For example, it is possible to pose the rank $R$ approximation...
problem: given a tensor $\mathcal{T}$, find a tensor $\mathcal{X}$ that solves the problem
\[
\min_{\mathcal{X}} \|\mathcal{T} - \mathcal{X}\|, \quad \text{s. t.} \quad \text{rank} \mathcal{X} \leq R. \tag{6}
\]
This problem has the potential to decrease the number of parameters necessary to represent the tensor, but it is, in general, not well posed and may not have a solution \cite{21}. One case when this always has a solution is with order 2 tensors—that is, matrices. But, for general order $K$ tensors, the case where it is always guaranteed to have a solution is when $R = 1$, the so called rank-one approximation. This fact follows from the next proposition.

**Theorem 1.** The set $D = \{\mathcal{X} : \text{rank} \mathcal{X} \leq 1\}$ of decomposable tensors is closed.

**Proof.** Proof given by \cite{21} in Proposition 4.2. \hfill \square

It is, therefore, advantageous to base signal processing algorithms over the problem of rank-one approximations, as these algorithms will most likely be well-behaved.

### III. The Volterra Series

Traditionally, the Volterra series is treated as an universal approximator for nonlinear systems. It can be described as a polynomial representation of the system, closely related to a Taylor series representations. Explicitly, given a nonlinear system with input signal $u(i)$ and output $y(i)$, its Volterra series representation is the sum
\[
y(i) = y^0 + y^1(i) + y^2(i) + y^3(i) + \cdots, \tag{7}
\]
where $y^0$ is a constant and each $y^k(i)$ is called the order $K$ homogeneous component of the system and is given by
\[
y^k(i) = \sum_{i_1, \ldots, i_k} \mathcal{H}_k(i_1, \ldots, i_k) u(i - i_1) \cdots u(i - i_k), \tag{8}
\]
in which $\mathcal{H}_k(i_1, \ldots, i_k)$ is a set of parameters called the order $K$ Volterra kernel.

The truncated Volterra series is obtained by limiting this series both in order and in time. This results in
\[
y(i) = y^0 + y^1(i) + \cdots + y^K(i) \tag{9}
\]
and
\[
y^k(i) = \sum_{i_1, \ldots, i_k=0}^M \mathcal{H}_k(i_1, \ldots, i_k) u(i - i_1) \cdots u(i - i_k), \tag{10}
\]
where $K$ is the order of the representation and $M$ its memory parameter. If follows from the Stone–Weierstrass theorem that the truncated Volterra series is an universal approximator—it can arbitrarily approximate any continuous nonlinearity, given that $K$ and $M$ are big enough.

Although this is an interesting property, the representational complexity, the number of the parameters necessary to describe an order $k$ Volterra kernel is $O(M^k)$, or, by exploring symmetry, $O\left(\binom{M-k+1}{k}\right)$. This high complexity gets transferred to Volterra-based algorithms, something that has historically barred their use in applications.

To deal with this problem, a common approach in the literature is to assume some restriction in the series and/or kernels. For example, for second order kernels, one possible approach is to truncate some of the diagonals of $\mathcal{H}_2(i,j)$ \cite{17, 8}, which allows the the computation to be written as
\[
y^2(i) = \sum_{d=0}^{D-1} \sum_{i_1} \mathcal{H}_2(i_1, i_1 + d) u(i - i_1) u(i - i_1 - d), \tag{11}
\]
where the $D$ is the number of diagonal utilized. When $D = 1$, this uses only the main diagonal, with $M$ parameters, resulting in a model called the Power Filter \cite{24}.

Another approach is to start from tensor representations of the Volterra kernel. The multi-index structure of the Volterra kernel is quite obvious and it may as well have been represented as $(\mathcal{H}_k)^{i_1\cdots i_k}$, making evident that it is a tensor. In a more conceptual way, one may realize that the $k$-th order component may as well be computed as a $k$-linear transformation.

First, define the input vector as the row vector $(u_i)_j = u(i - j + 1)$, or:
\[
u_i = [ u(i) \ u(i-1) \ \ldots \ u(i-M+1)] \tag{12}
\]
Then, notice how (10) may be rewritten, using the conventions of \cite{5}, as
\[
y^k(i) = u^{\otimes k} \mathcal{H}_k, \tag{13}
\]
where $u^{\otimes k} \triangleq u_i \otimes \cdots \otimes u_i$ is the tensor power of $u_i$—the tensor product of $u_i$ with itself $k$ times. Eq. (13) could be seen as a function $f: \mathbb{C}^M \times \cdots \times \mathbb{C}^M \rightarrow \mathbb{C}$ given by
\[
f(v_1, \ldots, v_k) = (v_1 \otimes \cdots \otimes v_k) \mathcal{H}_k, \tag{14}
\]
which is, by the multilinearity of the tensor product, a $k$-linear function. So, from the universal property \cite{17}, it is natural to expect the Volterra series to have a tensor representation.

This representation has been used, for symmetric tensors, in \cite{12}, to reduce the representational complexity of the series.

In this paper, the restricting hypothesis will be simply the one of decomposability of kernel.

#### A. Decomposable Model

Consider the Volterra kernel tensor $\mathcal{H}_K$ of order $K$ and assume that it is a decomposable tensor, that is, that there exists vectors $v_1, \ldots, v_K \in \mathbb{C}^M$ such that
\[
\mathcal{H}_K = v_1 \otimes \cdots \otimes v_K. \tag{15}
\]
A kernel like this will be said to satisfy the decomposable model—also called the Simple Multilinear Model (SML) \cite{20}.

This model allows for great gains in computational complexity, as it can be seen by carrying out the following computations:
\[
y^K(i) = u_i^{\otimes k} \mathcal{H}_K = (u_i \otimes \cdots \otimes u_i) (w_1 \otimes \cdots \otimes w_K) \\
= \sum_{i_1, \ldots, i_K=0}^M u(i - i_1) \cdots u(i - i_K) w_1(i_1) \cdots w_K(i_K) \\
= \sum_{i_1=1}^M w_1(i_1) u(i - i_1) \cdots \sum_{i_K=1}^M w_1(i_K) u(i - i_1) \\
= (w_1 \otimes \cdots \otimes w_K). \tag{16}
\]
This means that the output may be computed as a product of the outputs of \( K \) FIR linear systems. This is represented in Fig. 1. In terms of computational complexity, this result in something based on \( O(KM) \), an exponential reduction when compared to the original complexity of the Volterra series, which was of \( O(M^K) \). This, together with the well-posedness of the estimation problem should make for a good basis onto which develop adaptive algorithms.

IV. ESTIMATION THEORY OF THE DECOMPOSABLE MODEL

Suppose one wants to estimate the best rank-one approximation of the homogeneous component \( y_K(i) \) of the Volterra series by estimating the approximation of its kernel \( H_K \). The problem can be posed in the following way: one is given a random signal \( d \) (the desired signal) and a random \( 1 \times M \) vector \( u \) (the regressor). One wants to find the \( K \) order decomposable Volterra kernel \( W \) that best estimates \( d \), in the mean-square sense. This can be put in terms of the constrained optimization problem below:

\[
\min_W \mathbb{E} \left[ |d - u^\otimes K W|^2 \right], \quad \text{s. t.} \quad \text{rank}(W) \leq 1 \tag{17}
\]

This problem has a cost function given by \( J(W) = \mathbb{E} \left[ |d - u^\otimes K W|^2 \right] \). This cost function is called the Mean Square Error (MSE), while the signal \( e \) is \( d - u^\otimes K W \). The decomposability constraint can be included in this function by using \( W = w_1 \otimes \cdots \otimes w_K \), for vectors \( w_1, \ldots, w_K \ (M \times 1) \). It also will be convenient to also define the vector \( w \ (KM \times 1) \) built by vertically stacking the vectors \( w_1, \ldots, w_K \).

It is possible to explicitly develop the MSE as

\[
J(w) = \mathbb{E}|d|^2 - \mathbb{E} \left[ (d - u^\otimes K W)^* (d - u^\otimes K W) \right] = \mathbb{E}|d|^2 - \mathbb{E}|u^\otimes K W|^2 - \mathbb{E} \left[ (d - u^\otimes K W)^* (d - u^\otimes K W) \right] + \mathbb{E} \left[ (u^\otimes K W)^* (u^\otimes K W) \right] = R_d - W^* R_u^K d - R_u^K W + W^* R_u^K W, \tag{18}
\]

where the correlation parameters were defined as

\[
R_u^K = \mathbb{E} \left[ (u^\otimes K)^* u^\otimes K \right], \quad R_u^K d = \mathbb{E} \left[ (u^\otimes K)^* d \right], \quad R_d = \mathbb{E} |d|^2. \tag{19-21}
\]

The minimum of this function is achieved at a point where the gradient, over \( w \), is zero. This gradient has a block structure, each depending only on one \( w_s \):

\[
\nabla J(w) = \left[ \frac{\partial J}{\partial w_1} \frac{\partial J}{\partial w_2} \cdots \frac{\partial J}{\partial w_K} \right]. \tag{22}
\]

Each block may be computed as in the next proposition. The derivatives used are Wirtinger derivatives.

**Proposition 1.** The gradient of the MSE function with respect to \( w_s \) can be computed as

\[
\frac{\partial J}{\partial w_s} = -R_u^K + W^* R_u^K \mathbb{E} W(s), \tag{23}
\]

with \( \mathbb{E} W(s) = (w_1 \otimes \cdots \otimes \hat{w}_s \otimes \cdots \otimes w_K) \), where \( \hat{w}_s \) implies that \( w_s \) has been substituted for the identity matrix \( I_M \) of order \( M \) in the product.

**Proof.** The following tensor indexations shall be used:

\[
(R_u^K)^{j_1 \ldots j_K} = \mathbb{E} \left[ (u^\otimes K)^{j_1 \ldots j_K} (u^\otimes K)^{j_1 \ldots j_K} \right] \quad (25)
\]

\[
(R_u^K)^{j_1 \ldots j_K} = \mathbb{E} \left[ [u^\otimes K]^{j_1 \ldots j_K} \right] \tag{26}
\]

Through this, we can write

\[
R_u^K W = \sum_{j_1 \ldots j_K} (R_u^K)^{j_1 \ldots j_K} \prod_{\ell=1}^K (w_\ell)^{j_\ell}, \tag{27}
\]

where \((w_\ell)^{j_\ell}\) is the \( j_\ell \)-th coordinate of \( w_\ell \), and

\[
W^* R_u^K W = \sum_{i_1 \ldots i_K p=1}^{K} \prod_{j=1}^K (w_p)^{j_p} (R_u^K)^{i_1 \ldots i_K j_1 \ldots j_K} \prod_{\ell \neq s} (w_\ell)^{j_\ell}. \tag{28}
\]

The other terms from \( (18) \) involve only the conjugates of the entries of \( w \), so their Wirtinger derivatives become zero \( (22) \).

The gradient over the vector \( w_s \) is given by the derivatives over each of its components, which, by introducing the symbol \( \delta_{ij}^s \)—called the Kronecker delta—that evaluates to 0 if \( i \neq j \) and to 1 if \( i = j \), results in:

\[
\frac{\partial (R_u^K d W)}{\partial (w_s)^{j_s}} = \sum_{j_1 \ldots j_K} (R_u^K d)^{j_1 \ldots j_K} \prod_{\ell \neq s} (w_\ell)^{j_\ell} \delta_{ij_s}. \tag{29}
\]

As \( j_s \) take values from 1 through \( M \), \( (29) \) indeed renders the vector \((25)\). This can be shown as follows. The Kronecker delta \( \delta_{ij}^s \) indexes the identity matrix, that is, \((I_M)^{j_s} = \delta_{ij}^s \). In \( (29) \), the delta works as if occupying the positions of the coordinates \((w_\ell)^{j_\ell}\)—that is, the coordinates of an identity \( I_M \) are occupying the positions of the coordinates of the vector \( w_s \). This is valid for any index \( j_s \), thus, when reconstructing the tensor form implied by the coordinates computed in \( (29) \), the substitution with \( I_M \) in \( (24) \) must be made:

\[
\frac{\partial (R_u^K d W)}{\partial (w_s)^{j_s}} = R_u^K d (w_1 \otimes \cdots \otimes \hat{w}_s \otimes \cdots \otimes w_K) = R_u^K d W(s). \tag{30}
\]

For \( W^* R_u^K W \), we have, remembering we do not derivate the conjugates,

\[
\frac{\partial (W^* R_u^K W)}{\partial (w_s)^{j_s}} = \sum_{i_1 \ldots i_K p=1}^{K} \prod_{\ell \neq s} (w_p)^{j_p} (R_u^K)^{i_1 \ldots i_K j_1 \ldots j_K} \prod_{\ell \neq s} (w_\ell)^{j_\ell} \delta_{ij_s}. \tag{31}
\]

\(^1\)The star operation (*) on a tensor \((T)^{i_1 \ldots i_k}_{j_1 \ldots j_k}\) consists of changing lower indexes for upper indexes and taking the complex conjugate of the entries:

\((T^*)^{i_1 \ldots i_k}_{j_1 \ldots j_k} = (T)^{j_1 \ldots j_k}_{i_1 \ldots i_k} \).
Under the same argument,
\[
\frac{\partial (WR_{u,K} W^*)}{\partial w_s} = W^* R_{u,K} W(s).
\]
Therefore, one combines those two terms to get
\[
\frac{\partial J}{\partial w_s} = \frac{\partial}{\partial w_s} (R_d - W^* R_{u,K} W) = -\frac{\partial (R_{u,K} W)}{\partial w_s} + \frac{\partial (W R_{u,K} W^*)}{\partial w_s} = [-R_{u,K}^* + W^* R_{u,K}] W(s).
\]

In a critical point \( \nabla J(w) = 0 \) of this surface, \( w \) satisfies a series of polynomial equations in its entries reminiscent of the normal equations of linear estimation theory:
\[
[-R_{u,K}^* + W^* R_{u,K}] W(s) = 0, \quad 1 \leq s \leq K.
\]

One can verify the possibility—because this surface is multimodal—of minimality of a certain \( w \) by testing it in this equation, but a solution via direct methods may be difficult. It should be more viable to use iterative algorithms.

V. THE STEEPEST DESCENT ALGORITHM

The steepest descent algorithm can be used to find the minimum of (38). This algorithm is based on the recursion
\[
w_i = w_{i-1} - \mu [\nabla J(w_{i-1})]^*.
\]

In terms of the individual vectors, this may be rewritten as
\[
w_{s,i} = w_{s,i-1} + \mu W(s) [R_{du,K} - R_{u,K} W].
\]

Due to the nonlinear nature of the algorithm, there are some issues with the initializations of the \( w_s \) parameters. First, there should not be initialized all at zero, as (36) shows that this would make the parameters stay at zero through all of the iteration process. Another bad initialization is to set all of them at the same initial values. The recursion shows that this would lead to them being adapted exactly in the same way, never being able to diverge from one another. Initializing the parameters as pairs of opposite vectors leads to a similar problem with alternating iterations. Aside from random initializations, the previous discussion makes the heuristic argument to initialize the vectors in different scales. The following set of initializations has shown good results in our experiments.
\[
w_{s,-1} = \begin{bmatrix} 2^{-s+1} & 0 & \ldots & 0 \end{bmatrix}, \quad 1 \leq s \leq K - 1
\]
\[
w_{K,-1} = \begin{bmatrix} 0 & 0 & \ldots & 0 \end{bmatrix}.
\]

This recursion was used to find a solution to the estimation problem with
\[
d = u \otimes K \mathcal{H} + v,
\]
where \( u \) is a 1 \times 10 independent Gaussian vector, \( v \) is a signal independent from \( u \) with variance \( \sigma^2 = 10^{-3} \) and the plant \( \mathcal{H} \) is decomposable. The resulting MSE curve is on Figs. 2a (\( K = 2 \)) and 2b (\( K = 3 \)). It is possible to notice how the algorithm is able to reach the theoretical minimum MSE of \( \sigma^2 \). Moreover, it was possible to verify that the solutions found satisfy the normal equations (34).

Although some good behavior of this algorithm can be attested by these simulations, in practice it is very hard for one to obtain the \( R \) parameters. So a recursion like (36) is rarely used—the interest in it is a rather theoretical one. A solution to this problem is to compute real-time approximations to (39).

This is called an adaptive solution. Not only is solves the problem of unavailability of the correlation parameters, it also gives the algorithm some tracking capabilities. When doing this to the linear estimation problem, the algorithm obtained is the classical LMS (22). When done to the decomposable model, it should, therefore, lead to similar algorithms.

VI. ADAPTIVE ALGORITHMS

As it is impractical to use the steepest descent algorithm, one finds an adaptive solution that implements a recursion that approximates (36). This is done by computing a real-time approximation to the parameters (19).

A. RECTANGULAR WINDOW APPROXIMATION

A possible approximation is to use a rectangular window to estimate the correlation parameters—that is, the last \( L \) samples of the signals are used in a average to compute the approximations
\[
\tilde{R}_{u,K} = \frac{1}{L} \sum_{j=-L+1}^{1} u_j \otimes K \tilde{u}_j \otimes K,
\]
\[
\tilde{R}_{u,K}^* = \frac{1}{L} \sum_{j=-L+1}^{i} u_j^* \otimes K d(j)^* = \tilde{R}_{du,K}.
\]

This allows for the computation of an estimate of the block-gradient (23):
\[
\frac{\partial J}{\partial w_s} = [-\tilde{R}_{u,K}^* + W^* \tilde{R}_{u,K}] W(s) = \frac{1}{L} \sum_{j=-L+1}^{i} [-d(j)^* + W^* u_j^* \otimes K] u_j^* \otimes K W(s).
\]

Equation (42) may be conveniently rewritten as follows. First define \( y_s(j) \triangleq (u_j w_1) \cdot \ldots \cdot (u_j w_K) \), where the hat implies a factor being omitted, which implies \( u_j^* \otimes K W(s) = y_s(j) u_j \), then define three variables:
\[
d_i \triangleq \begin{pmatrix} d(i) \\ \vdots \\ d(i-L+1) \end{pmatrix} \quad (L \times 1), \quad U_i^K \triangleq \begin{pmatrix} u_i^K \\ \vdots \\ u_{i-L+1}^K \end{pmatrix} \quad (L \times K M),
\]
\[
y_j \triangleq (y_1(j) \ldots y_K(j)) \quad (1 \times K).
\]
Eq. (42) represents a block-gradient. For each \( j \), the terms of this sum will be multiplied only by \( y_s(j) \)—the \( s \)-th element of \( y_j \)—which shows that the total gradient has a Kronecker structure. This follows from the fact that \( u_j^{0 \otimes K} \mathbf{W}^{(s)} = y_s(j)u_j \). Therefore, write

\[
\nabla J = \frac{1}{L} \sum_{j=1}^{i} \left[ -d(j)^* + W^* u_j^{0 \otimes K} \right] (y_j \otimes u_j).
\]  

(44)

By using the variable

\[
T_i \triangleq \begin{pmatrix} y_i \otimes u_i \\ \vdots \\ y_{i-L+1} \otimes u_{i-L+1} \end{pmatrix} \quad (L \times K M),
\]  

(45)

a compact expression for the instantaneous gradient is obtained:

\[
\nabla J = -\frac{1}{L} [d_i - U_i^{0 \otimes K} \mathbf{W}]^* T_i = -\frac{1}{L} e_i^* T_i,
\]  

(46)

where \( e_i \triangleq d_i - U_i^{0 \otimes K} \mathbf{W} \) (\( L \times 1 \)). Finally, defining \( y(j) \triangleq u^{0 \otimes K} \mathbf{W} \) and

\[
y_i \triangleq (y(i) \quad \cdots \quad y(i - L + 1))^T \quad (L \times 1),
\]  

(47)

results in an expression for \( e_i \) as

\[
e_i = d_i - y_i.
\]  

(48)

B. LMS Algorithm

When \( L = 1 \), Eq. (46) becomes

\[
\nabla J = -e(i)^* (y_i \otimes u_i),
\]  

(49)

where \( e(i) \triangleq d(i) - u_i^{0 \otimes K} \mathbf{W} \). When separated for each \( w_s \), the LMS equation becomes

\[
w_{s,i} = w_{s,i-1} + \mu e(i) y_s(i)^* u_i^*.
\]  

(50)

This is a Least Mean Squares (LMS) recursion.

C. True-LMS Algorithm

When using a general value of \( L \), the recursion of the TRUE-LMS (also sometimes called the Dat Reuse LMS) is derived:

\[
w_i = w_{i-1} + \frac{\mu}{L} T_i^* e_i.
\]  

(51)

This can, by separating each \( w_s \), be further simplified to

\[
w_{s,i} = w_{s,i-1} + \frac{\mu}{L} \sum_{j=i-L+1}^{i} [d(j) - u_j^{0 \otimes K} \mathbf{W}_{i-1}] y_{s}(j)^* u_j^*
\]  

\[
= w_{s,i-1} + \frac{\mu}{L} (y_{s,i} \odot U_i)^* e_i,
\]  

(52)

where

\[
y_{s,i} \triangleq \begin{pmatrix} y_{s}(i) \\ \vdots \\ y_{s}(i - L + 1) \end{pmatrix} \quad (L \times K), \quad U_i \triangleq \begin{pmatrix} u_i \\ \vdots \\ u_{i-L+1} \end{pmatrix} \quad (L \times M)
\]  

(53)

and \( \odot \) denotes row-wise scalar multiplication.

D. Stabilization of the Algorithms

It has been known that nonlinearities in the update equations may lead to a nonzero probability of divergence for the algorithm, no matter how small is the step-size [23], whenever the probability distribution of the signals has infinite support. This is related to the fact that the algorithms are not globally asymptotically stable. An example of this phenomenon is illustrated in Fig. 3a. The big “jumps” in the MSE curve shows the algorithm entering momentarily an area of instability. If the parameters got far enough from the stability region, the algorithm would diverge to infinity, as is shown on Fig. 3b.

For effective use of these algorithms, some normalization may be necessary. In the case of the LMS, a normalization may be performed by conditionally changing the update equation as in (54):

\[
w_{s,i} = \begin{cases} w_{s,i-1} + \mu e(i) y_{s}(i)^* u_i^* & \text{if } |y_{s}(i)| \leq \text{MAX} \\
 w_{s,i-1} + \mu e(i) u_i^* & \text{if } |y_{s}(i)| > \text{MAX} \end{cases}
\]  

(54)

This can be justified in a heuristic way. Whenever the algorithm starts diverging, \( |y_{s}(i)| \) starts to increase. If it gets above a certain threshold, the update equation turns into the second line of (54). If the step-size is small enough, this equation is approximately the classical LMS, an algorithm that is known to be stable in various situations. This stops the divergent behavior of the algorithm. At this point, the \( w \) is not converging to the optimal value. This works as a gross convergence mode, that is guaranteed to be stable enough to allow \( |y_s(i)| \) to decrease, entering again the mode of fine convergence. Figure 4 shows a simulation of this modified algorithm, averaged through 1 million realizations, with no signs of instability.
Additionally, a TRUE-LMS version of this modification can be done as

\[
w_{s,i} = \begin{cases} w_{s,i-1} + \frac{\mu}{L} e_i & \text{if } |y(i)| \leq \text{MAX} \\ w_{s,i-1} + \frac{\mu}{L} u_i & \text{if } |y(i)| > \text{MAX} \end{cases}
\]

which works for the same reasons as before.

E. Step Bounds

For an initial description of the algorithms to complete, there must exist a method to choose the parameters of the algorithms. In this section a heuristic bound is obtained, based on comparisons with the classical linear LMS.

When computing the gradient for the classical algorithm, one obtains \(\nabla J = -e(i) u_i\). This leads to an instantaneous bound for the algorithm of \(0 < \mu < 2/\|u_i\|^2\). The gradient obtained for the new algorithm is in (59). Given that the bound depends only on this gradient, one would expect that an equivalent inequality for the new algorithm would be

\[
0 < \mu < \frac{2}{\|y_i \otimes u_i\|^2} = \frac{2}{\|y_i\|^2 \|u_i\|^2}.
\]

This result is similar to the one obtained in [19]. Although important, instantaneous bounds may not be sufficient to choose the parameters of the algorithm. The classical LMS has a bound given by \(0 < \mu < 2/(3 \text{tr } R_u)\) that guarantees convergence in the MSE [26]. Given that \(R_u = \mathbb{E}\{u_i^* u_i\}\), an argument by comparison shows that the new nonlinear algorithm would depend on the trace of

\[
\mathbb{E}\{(y_i \otimes u_i)^* (y_i \otimes u_i)\} = \mathbb{E}\{(y_i^* y_i) \otimes (u_i^* u_i)\}.
\]

In fact, simulations show that, for the exact gradient algorithm, this is exactly the parameter that governs convergence. If \(\lambda_{max}\) is the biggest eigenvalue of \(\lim_{i \to \infty} \mathbb{E}\{(y_i^* y_i) \otimes (u_i^* u_i)\}\), then the steepest descent algorithm on (36) with initial conditions (37) converges to its global minimum whenever \(0 < \mu < 2/\lambda_{max}\). This is analogous to the linear steepest descent. [22]

For the adaptive algorithms, since \(y_i\) is not a stationary signal, this is still not an universal bound. To make it so, one should consider the signal after convergence, since, at that point, the weight vectors \(u_1, \ldots, u_K\) should be stationary, which makes \(y_i\) an stationary signal, given that \(u_i\) is also one. Under these conditions, the bound should be

\[
0 < \mu < \lim_{i \to \infty} \frac{2}{\alpha \text{tr } \mathbb{E}\{(y_i^* y_i) \otimes (u_i^* u_i)\}},
\]

where \(\alpha\) is a constant. Due to the complexity involved in determining analytically the value of \(\alpha\), which must take into account the modified algorithms (54) and (55), an experimental approach was used, which pointed to a possible value of \(\alpha = 3^K\), which depends on the order of nonlinearity \(K\). This dependence can be explained by noting that higher \(K\) yields a noisier convergence, which may put the algorithms in a unstable region, if the step-size is too big. One should note that in the linear case \((K=1)\) this bound reduces to the classical LMS bound.

It is also possible to give an estimate to the MAX parameter:

\[
\text{MAX} = \lim_{i \to \infty} (K+1) \sqrt{\mathbb{E}\{|u_i^\otimes K W_i|^2\}}.
\]

This value was also obtained mostly through experimental verification, but heuristic reasoning is also possible. The \(\sqrt{\mathbb{E}\{|u_i^\otimes K W_i|^2\}}\) factor appears because the parameter MAX gives a condition on \(|y_k(i)|\), so it should depend on the power of the output \(y(i)\). The factor \(K+1\) appears due to the experimental observation that the bigger the \(K\), the noiser was the convergence, therefore a bigger value of MAX is necessary to allow this natural convergence behavior to be expressed, otherwise the algorithm would not converge to the lowest possible minimum.

These results should be valid for both the LMS and the TRUE-LMS, while, as it will be shown in simulations, they may be conservative choices for the latter, as it has better stability properties. In fact, it can be verified experimentally that for reasonably sized \(L\) the ideal value of MAX for the TRUE-LMS is roughly half of (59).

F. Computational Complexity

Table I shows an efficient implementation of the TRUE-LMS algorithm, together with the necessary number of operations in each step of the process. It uses the following auxiliary variables:
TABLE I
IMPLEMENTATION OF THE TRUE-LMS.

| $\forall s, \ell, y_{oK}(i)$ | $L K (M - 1)$ | $L K M$ |
|----------------------------|-------------|--------|
| $\forall s, \ell, y_{oK}(i)$ | $0$ | $L K (K - 2)$ |
| $\forall s, \ell, y_{oK}(i)$ | $0$ | $L$ |
| $e_i$ | $L$ | $0$ |
| $T_i$ | $0$ | $L K M$ |
| $w_o + \frac{1}{\lambda} T_i e_i$ | $K M$ | $K M$ |

Total $2 L K M - L K + L$ $3 L K M + L K^2 + K M - 2 L K + L$

TABLE II
IMPLEMENTATION OF THE LMS.

| $\forall s, y_{as}(i)$ | $K (M - 1)$ | $K M$ |
|----------------------------|-------------|--------|
| $\forall s, y_{as}(i)$ | $0$ | $K (K - 2)$ |
| $\forall s, y_{as}(i)$ | $0$ | $1$ |
| $\forall s, [\mu e(i)] u_i$ | $0$ | $M + 1$ |
| $\forall s, w_o + [\mu e(i)] y_{oK}(i)$ | $K M$ | $0$ |
| $\forall s, w_o + [\mu e(i)] y_{oK}(i)$ | $u_i^T$ | $0$ |

Total $2 K M - K + 1$ $K M + K^2 + M - K + 2$

\[
y_{oK}(i) = (u_{i-\ell+1} w_o) \tag{60}
\]
\[
y_o(i) \triangleq y_{oK}(i) \cdots y_{oK}(i) \cdots y_{oK}(i) \tag{61}
\]
\[
y_o(i) \triangleq y_{oK}(i) y_{oK}(i) \tag{62}
\]

In Table I and implementation of the LMS is shown. Some optimizations in relation to the TRUE-LMS are possible, which makes it more efficient than one would expect by using $L = 1$ in the previous table.

VII. SIMULATIONS

Some experiments with the algorithm were run. In all of these, the input vector $u_i$ was chosen to have a delay-line structure as in

\[
u_i = [u(i) \ u(i - 1) \ \ldots \ u(i - M + 1)],
\]

with $u(i)$ an i.i.d. signal sampled from the normal distribution $N(0, 1)$.

The desired signal follows a “system identification” model, with

\[
d(i) = u_i^T W_o + v(i), \tag{64}
\]

where $W_o$ is the plant to be identified and $v(i)$ is an i.i.d. signal sampled from the distribution $N(0, \sigma_v^2)$ independent from $u(i)$.

All the adaptive algorithms simulated here are the stabilized versions \[54] and \[55]. They will be referred hereafter as SML.

A. Decomposable plants

The simulations in this section use a decomposable plant, which represents the best case scenario for the SML algorithms. They were put to run against the Volterra-LMS and Wiener-LMS algorithms \[27]. The Volterra algorithm is based on \[10] and the Wiener one is a modified version in which the resulting regressor is statistically orthonormal. They both are linear-in-the-parameters algorithms, in contrast with the SML ones, which are not.

Simulations were done for $K = 2$ and $\sigma_v^2 = 10^{-3}$ and only the SML-LMS algorithm was tested. The Excess Mean Square Error (EMSE) curves are on Fig. \[5a\]. The algorithms all have similar performances, but a difference starts to show after a few iterations, when the SML starts to converge faster and shows itself to be the first to reach convergence.

Fig. \[5b\] shows a similar scenario, with $\sigma_v^2 = 10^{-6}$; the only thing that was changed was the plant $w_o$. It is still decomposable and normalized plant, just a different one. This figure shows how the SML performance may be dependent on the plant being identified.

Fig. \[5c\] shows a simulation of the identification of an order $K = 3$ plant. Similar results were obtained. The special feature here is the really distinct trajectory of the SML algorithm, which is caused by it being nonlinear in the parameters.

On the first $K = 2$ plant, the LMS was simulated against the TRUE-LMS and the results are on Fig. \[6\]. The TRUE-LMS is shown as marginally faster while converging to the same EMSE as the LMS. Although the differences in the figure are minimal, the TRUE-LMS appeared to be a more stable algorithm. This can be explained by noting that the statistical parameters in its formulation are an average of samples, which

\[2\] Formally defined as $E[(W_o - W_i)^2]$. 

Fig. 5. Curves of the estimated EMSE showing the performance of the algorithms while identifying decomposable plants.
helps remove outliers which otherwise would try to take the algorithm to a region of instability.

B. Stability

As previously addressed in subsection VI-D, non-linearities in the algorithm recursion may lead to non-global stability. The following experiments use the same decomposable plant as the one used on Fig. 5a and are intended to showcase the stability of the algorithms with various step-sizes.

Fig. 6. EMSE of the LMS and the TRUE-LMS.

Fig. 7. Tests on Gaussian plants for hypothesis of decomposability.

Fig. 8. Comparing the steepest descent algorithm with the LMS.

For this plant, the following parameter was computed:

\[
\mu_0 = \lim_{i \to \infty} \frac{2}{3K \text{tr} \mathbb{E} \left[ (y_i^* y_i) \otimes (u_i^* u_i) \right]} \approx 0.1052. \tag{65}
\]

This experiment was run through a total 10,000 realizations and the number of diverging realizations was counted. Table III shows the results, for the LMS and the TRUE-LMS \((L = 4)\) and \((L = 8)\), as a function of the step-size.

It is possible to notice the stability of the algorithms for the steps lower than \(\mu_0\). Moreover, they become more stable the greater the value of \(L\), which was expected. Additionally, for the TRUE-LMS, the step-bound \(\mu_0\) may be too conservative, but this depends in a complex way on the value of \(L\).

C. Adaptive algorithms as an approximation of the steepest descent

Continuing with the plant on Fig. 5a, this section intends to show how the adaptive algorithms can be considered an approximation to the the steepest descent algorithm on (36), as long as the step-size is small enough. Here the curves of the TRUE-LMS are not shown, as they would be almost superimposed on the ones of the LMS, as it can be seen on Fig. 6. The curves of the adaptive algorithms form an ensemble average of 10,000 realizations and are present Fig. 8.

When the step-size is small, as in Fig. 8a, the curves are almost superimposed. When it is increased, they start to take different trajectories. Another noticeable effect is the increase of the minimum MSE in the adaptive algorithm.

| \(L\) | \(0.5\mu_0\) | \(0.9\mu_0\) | \(\mu_0\) | \(1.5\mu_0\) | \(2\mu_0\) |
|------|---------|---------|--------|--------|--------|
| LMS  | 0       | 0       | 1      | 50     | 420    |
| TRUE-LMS \((L = 4)\) | 0       | 0       | 0      | 12     | 74     |
| TRUE-LMS \((L = 8)\) | 0       | 0       | 0      | 0      | 8      |

Table III

Number of divergent realizations in the experiment.
D. A family of non-decomposable plants

In this section the hypothesis of decomposability is studied. A Volterra kernel given by a bivariate normal probability function was chosen as the family of test plants. The reason for this is that the the correlation coefficient $\rho$ turns out to be a good measure of the decomposability of the plant, with $\rho = 0$ being perfect decomposability and $\rho = 1$ the worst-case scenario. In explicit terms, the plant is given by

$$w_{\nu}(i, j) = \alpha \exp \left[ -\frac{(i - 1)^2 + (i - 1)^2 + 2\rho(i - 1)(j - 1)}{18(1 - \rho^2)} \right]$$

(66)

where $\alpha$ is a normalization parameter. A typical plant is shown on Fig. 7a.

The tests were made with $\sigma_i^2 = 10^{-3}$ and by varying the $\rho$ parameter between 0 and 1 by increments of 0.1. The results of the tests are on Fig. 7b. As predicted, the best results were with low values of $\rho$ and the worst one were with high values. This can be explained by thinking about the SVD. When $\rho \approx 0$, the singular value distribution of $w_{\nu}$ is concentrated in a single one. It is to the matrix associated to this singular value that the algorithm converges. Since this matrix contains most of the energy of the system, the minimum MSE is low. Conversely, with $\rho \approx 1$ the distribution of singular values of this matrix is more uniform. Even if the algorithm converges to the term with most energy, there will be a lot of energy remaining in the system, which corresponds to a higher minimum MSE.

E. SML versus algorithms in the literature

The algorithm was tested against various others from the literature, with $K = 2$. The ones tested were:

- The Power Filter (PF) [24] on [11] with $D = 1$, which uses only the diagonal elements of the Volterra series. 22 coefficients.
- The Simplified Volterra Filter (SV) [7], [8], the truncated diagonals model in [11], with $D = 3$. 60 coefficients.
- The Sparse Interpolated Volterra (IV) [9], [10], which estimates only a few entries of the Volterra kernel and interpolates the others. 66 coefficients.
- The full Volterra model (V) on [10]. 231 coefficients.
- The SML-LMS algorithm on [50]. 42 coefficients.

The plant to be identified is a smooth one, as in [10]. It is represented in Fig. 9a. Here the memory parameter was $M = 21$. The results are on Fig. 9b. It shows the fast convergence and low minimum MSE of the SML filter, doing so with only 42 coefficients.

Fig. 9c shows a simulation against the Parallel Cascade Filter [11] (CF – 65 coefficients), a model that can be represented as a product of exactly two Volterra filters, and the SML (20 coefficients). They try to identify an $K = 3$ random decomposable plant. This experiment is here to showcase an structurally similar algorithm to the SML. They are both formed by products and are nonlinear in the parameters, but in this case the SML fares better. What is important to notice from this figure is that both algorithms have similar convergence paths, something that didn’t happen with any of the others.

VIII. Chaotic Behavior

One last experiment concerns the emergence of chaotic behavior of the algorithm for certain values of the step-size. Assume a scalar model, with $M = 1$, and $K = 2$. In this simple case, the algorithm will be adapting two parameters. Moreover, assume the model

$$d(i) = 100u(i)^2,$$

(67)

where $u(i) = 1$ is a constant signal.

The bifurcation diagram of the LMS algorithm, parametrized by $\mu$, is given by Fig. 10. It is possible to recognize a well-behaved convergence region in $0 < \mu < 0.01$, as predicted by step bounds. From there on, convergence starts to become oscillatory and starts to experience period doubling, until a point that is becomes fully chaotic. Starting from 0.016, the parameter starts jumping from positive to negative regions. A typical trajectory in this region is presented in Fig. 11.

When for $\mu > 0.02$, the algorithm starts to diverge, but until 0.023 this divergence seems chaotic. From there on it starts to appear monotonic, as expected.

Such behavior may be used in the way of improving convergence properties, such being part of a method to search for global minimums in multi-modal surfaces. This shall be a topic of investigation in further publications.

IX. Conclusion

This paper presents a low-complexity nonlinear adaptive filter based of the rank one approximation of the Volterra
Fig. 10. The bifurcation diagram of the system.

Fig. 11. Trajectory of $w_1 \approx 0.016$.

series. This is motivated both by the exponential reduction in complexity and in the well-posedness of the problem of rank-one approximations.

The rank-one model can be represented as a product of linear FIR filters, which results in algorithms with complexity $O(KM)$, as opposed to the initial $O(M^K)$ of the full Volterra series.

The description of this model is followed by the posing of an estimation problem. Its solution is given in the form of a steepest descent algorithm, which, as in classical adaptive filter theory, can make for the ideal model of an adaptive filter—they are approximations of the exact gradient algorithm. Indeed, simulations show that in the limit of small step sizes, the adaptive algorithms are expected to approximate really well the exact algorithm.

These adaptive filters derived as a rectangular window instantaneous approximation of the steepest descent. Depending on the number of samples on the window, two algorithms were derived: the LMS, with one sample, and the TRUE-LMS, which uses a general $L$ number of samples. Their trajectories can be verified to be roughly the same, but their properties differ in terms of stability.

Due to nonlinearities on the equations that implement the algorithms, it is expected that they do not achieve global asymptotic stability. An attempt at ameliorating this problem was done via a heuristic limitation of the algorithm. Other heuristic considerations were done in the choice of parameters for the filter. These considerations were later verified, albeit non-extensively, in simulations.

Other simulations involved testing of the algorithm in the identification of both rank-one and higher rank plants. It fares well in comparison with other algorithms when identifying many of these plants, specially those that can be considered approximately decomposable. The cases were the algorithms have poor performance were identified, for order $K = 2$ plants, as those whose Volterra kernels have evenly distributed singular values.

This paper finalizes with considerations on extensions. Some properties, like chaotic regions of the algorithm, may be studied further. Whether these properties either show to be detrimental to its performance or if reveal themselves to be of some application remains to be investigated.

Other areas of extension is in the algorithms themselves. The solution to the estimation problem can be further studied by the used of Newton’s method [20]. There is some work to be done in the ill-posedness of the general rank approximation and any development there could be applied in the development of general rank algorithms. Another extension is in the use of combinations of filters. For example, data-reuse-type algorithms, such as the TRUE-LMS, can have its performance...
improved by the use of incremental combinations. [28] The same should apply in the algorithms developed here.

APPENDIX A
NORMS AND INNER PRODUCTS OF TENSORS

Let \((V_1, \langle \cdot, \cdot \rangle_1), (V_2, \langle \cdot, \cdot \rangle_2), \ldots, (V_K, \langle \cdot, \cdot \rangle_K)\) be complex inner product spaces. The natural way of inducing an inner product on the tensor product \(V_1 \otimes V_2 \otimes \cdots \otimes V_K\) is by defining the inner product on the decomposable tensors as

\[
\langle v_1 \otimes v_2 \otimes \cdots \otimes v_K, w_1 \otimes w_2 \otimes \cdots \otimes w_K \rangle \triangleq \langle v_1, w_1 \rangle_1 \langle v_2, w_2 \rangle_2 \cdots \langle v_K, w_K \rangle_K
\]

and extending through sesquilinearity for general tensors, which are linear combinations of decomposable ones.

This inner product induces an \(\ell_2\) norm in the usual way as \(\|T\| = \sqrt{\langle T, T \rangle}\). For decomposable tensors, this is expressed as

\[
\|v_1 \otimes v_2 \otimes \cdots \otimes v_K\| = \|v_1\|_1 \|v_2\|_2 \cdots \|v_K\|_K ,
\]

where each \(\|\cdot\|_k\) is the respective induced \(\ell_2\) norm. For general normed spaces the induced tensor norms are not as simply (or uniquely) defined, but reasonable ones should satisfy the cross norm identity on the decomposable tensors, which is also represented by the previous equation. On Banach spaces, an example of tensor norm is the projective norm, given by

\[
\|T\|_\pi = \inf \left\{ \sum_i \|v_{1,i}\|_1 \|v_{2,i}\|_2 \cdots \|v_{K,i}\|_K : \sum_i v_{1,i} \otimes v_{2,i} \otimes \cdots \otimes v_{K,i} = T \right\},
\]

that is, the smallest sum of norms of all the possible forms of writing \(T\) as a sum of decomposable terms. Another norm, called the injective norm, is defined in terms of linear functionals, but, regardless of the details, what is important is that these are all cross norms i.e. they separate on the decomposable tensors.

REFERENCES

[1] A. N. Birkett and R. A. Goubran, “Limitations of handsfree acoustic echo cancellers due to nonlinear loudspeaker distortion and enclosure vibration effects,” in Applications of Signal Processing to Audio and Acoustics, 1995. IEEE ASSP Workshop on, Oct 1995, pp. 103–106.
[2] R. W. Lucky, “Modulation and detection for data transmission on the telephone channel,” in New Directions in Signal Processing in Communication and Control, 1975.
[3] F. X. Y. Gao and W. M. Snelgrove, “Adaptive linearization of a loudspeaker,” in Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on, Apr 1991, pp. 3589–3592 vol.5.
[4] H. K. Kwan and Q. P. Li, “New nonlinear adaptive fir digital filter for broadband noise cancellation,” IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 41, no. 5, pp. 355–360, May 1994.
[5] V. Mathews, “Adaptive polynomial filters,” Signal Processing Magazine, IEEE, vol. 8, no. 3, pp. 10–26, July 1991.
[6] V. J. Mathews and G. L. Sicuranza, Polynomial Signal Processing, Wiley-Interscience, 2000.
[7] A. Fermo, A. Carini, and G. L. Sicuranza, “Low-complexity nonlinear adaptive filters for acoustic echo cancellation in gsm handset receivers,” European Transactions on Telecommunications, vol. 14, no. 2, pp. 161–169, 2003. [Online]. Available: http://dx.doi.org/10.1002/ett.908
[8] ———, “Simplified Volterra filters for acoustic echo cancellation in gsm receivers,” in Signal Processing Conference, 2000 10th European, Sept 2000, pp. 1–4.
[9] E. Batista, O. J. Tobias, and R. Seara, “A fully lms adaptive interpolated Volterra structure,” in Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on, March 2008, pp. 3613–3616.
[10] E. Batista, O. Tobias, and R. Seara, “A sparse-interpolated scheme for implementing adaptive Volterra filters,” Signal Processing, IEEE Transactions on, vol. 58, no. 4, pp. 2022–2035, April 2010.
[11] T. M. Panicker, V. J. Mathews, and G. L. Sicuranza, “Adaptive parallel-cascade truncated Volterra filters,” IEEE Transactions on Signal Processing, vol. 46, no. 10, pp. 2664–2673, Oct 1998.
[12] 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, 2012.
[13] W. Greub, Multilinear Algebra, ser. Universitext. Springer New York, 2012.
[14] T. Andre, R. Nowak, and B. Van Veen, “Low rank estimation of higher order statistics,” in Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on, vol. 5, May 1996, pp. 3026–3028 vol. 5.
[15] R. Nowak and B. Van Veen, “Tensor product basis approximations for volterra filters,” Signal Processing, IEEE Transactions on, vol. 44, no. 1, pp. 36–50, Jan 1996.
[16] P. Comon, “Tensors: a brief introduction,” IEEE Signal Processing Magazine, vol. 31, no. 3, pp. 44–53, 2014.
[17] A. P. da Silva, P. Comon, and A. L. F. de Almeida, “A finite algorithm to compute rank-1 tensor approximations,” IEEE Signal Processing Letters, vol. 23, no. 7, pp. 959–963, July 2016.
[18] M. Rupp and S. Schwarz, “Gradient-based approaches to learn tensor products,” in Signal Processing Conference (EUSIPCO), 2015 23rd European, Aug 2015, pp. 2486–2490.
[19] ———, “A tensor lms algorithm,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2015, pp. 3347–3351.
[20] F. C. Pinheiro and C. Lopes, “Newton-like nonlinear adaptive filters via simple multilinear functionals,” in 2016 24th European Signal Processing Conference (EUSIPCO) (EUSIPCO 2016), Budapest, Hungary, Aug. 2016, pp. 1588–1592.
[21] V. de Silva and L.-H. Lim, “Tensor rank and the ill-posedness of the best low-rank approximation problem,” SIAM Journal on Matrix Analysis and Applications, vol. 30, no. 3, pp. 1084–1127, 2008. [Online]. Available: http://dx.doi.org/10.1137/0606518X.
[22] A. H. Sayed, Adaptive Filters. Wiley-IEEE Press, 2008.
[23] J. Brewer, “Kronecker products and matrix calculus in system theory,” Circuits and Systems, IEEE Transactions on, vol. 25, no. 9, pp. 772–781, Sep 1978.
[24] F. Knech, A. Minnacht, and W. Kellermann, “Nonlinear acoustic echo cancellation using adaptive orthogonalized power filters,” in Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP ’05). IEEE International Conference on, vol. 3, March 2005, pp. iii/105–iii/108 Vol. 3.
[25] V. H. Nascimento and J. C. M. Bermudez, “Probability of divergence for the least-mean fourth algorithm,” IEEE Transactions on Signal Processing, vol. 54, no. 4, pp. 1376–1385, April 2006.
[26] A. Feuer and E. Weinstein, “Convergence analysis of lms filters with uncorrelated gaussian data,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 33, no. 1, pp. 222–230, Feb 1985.
[27] T. Ogunfunmi, Adaptive Nonlinear System Indentification: The Volterra and Wiener Model Approaches. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[28] L. F. O. Chamon and C. G. Lopes, “There’s plenty of room at the bottom: Incremental combinations of sign-error lms filters,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2014, pp. 7248–7252.