IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 21, 2021, accepted September 30, 2021, date of publication October 5, 2021, date of current version October 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3118095

Adaptive System Identification via Low-Rank Tensor Decomposition

CHRISTINA AUER[1], (Member, IEEE), OLIVER PLODER[1], (Graduate Student Member, IEEE), THOMAS PAIREDER[1], (Graduate Student Member, IEEE), PÉTER KOVÁCS[2,3], OLIVER LANG[2], (Member, IEEE), AND MARIO HUEMER[2], (Senior Member, IEEE)

[1]Christian Doppler Laboratory for Digitally Assisted RF Transceivers for Future Mobile Communications, Institute of Signal Processing, Johannes Kepler University, 4040 Linz, Austria
[2]Institute of Signal Processing, Johannes Kepler University, 4040 Linz, Austria
[3]Department of Numerical Analysis, Eötvös Loránd University, 1117 Budapest, Hungary

Corresponding authors: Christina Auer (christina.auer@jku.at) and Oliver Ploder (oliver.ploder@jku.at)

This work was supported in part by Austrian Federal Ministry for Digital and Economic Affairs, in part by the National Foundation for Research, Technology and Development, and in part by Christian Doppler Research Association. The work of Péter Kovács was supported by János Bolyai Research Scholarship of Hungarian Academy of Sciences.

ABSTRACT Tensor-based estimation has been of particular interest of the scientific community for several years now. While showing promising results on system estimation and other tasks, one big downside is the tremendous amount of computational power and memory required – especially during training – to achieve satisfactory performance. We present a novel framework for different classes of nonlinear systems, that allows to significantly reduce the complexity by introducing a least-mean-squares block before, after, or between tensors to reduce the necessary dimensions and rank required to model a given system. Our simulations show promising results that outperform traditional tensor models, and achieve equal performance to comparable algorithms for all problems considered while requiring significantly less operations per time step than either of the state-of-the-art architectures.

INDEX TERMS Tensor decomposition, LMS, machine learning, low rank approximation.

I. INTRODUCTION

In recent years, machine learning (ML) gained in popularity among the scientific and industrial as well as the signal processing communities. Although ML is usually associated with deep learning (DL) and neural networks (NN) [1] and is among the more broadly used techniques in signal processing and other fields [2]–[6], this term covers many other techniques such as random forests [7], [8], support vector machines [9]–[11], kernel adaptive filters [12]–[14] and tensor-based learning (see e.g. [15]). The latter is often disregarded for the comparably big amount of memory and computational power needed to approximate a nonlinear system, although this approach can achieve on-par performance with all other mentioned techniques [16], [17].

Despite of their high computational complexity, tensors are becoming a more and more popular tool for the signal processing society. Applications in this domain include system identification [18]–[20], channel identification [21], [22], separation of speech signals [23], emitter localization for radar, communication applications and passive sensing [24], [25]. Of course, tensors are also used in more classical ML tasks such as face recognition, mining musical scores and detecting cliques in social networks [15], [26], [27].

However, a big downside for many in this domain is, that tensors not only require a vast amount of memory in order to yield an appropriate estimate, but the training of such an estimator is usually also a rather complex task with current approaches. One way to mitigate these problems are tensor trains (TTs) [28], which aim to reduce the necessary size of a tensor by splitting it into multiple – smaller – tensors which are cascaded one after another, in an attempt to make the overall system less complex and less memory intensive. Further, [29], [30], developed an (extended) Kalman filter which makes use of tensors in order to enhance performance and keep the rank of the used tensor low. Recently, [31] developed a combination of TTs and B-splines to, again, keep the complexity of the tensors low while also leveraging the advantages of B-splines for their system identification task.

The associate editor coordinating the review of this manuscript and approving it for publication was Shovan Barma.

A well known approach is to split systems up in their linear and nonlinear parts (if model knowledge is available), so called Wiener and Hammerstein model [32] or combinations thereof. Scarpiniti et.al. [33] used this approach to train spline adaptive filters (SAFs) to identify the system at hand with rather high success. We introduce a similar concept that uses a cascade of tensors and linear filters, which are adapted using a least mean squares (LMS) type algorithm to approximate an unknown system. In this paper, we derive the necessary update equations for all subsystems and show that systems that can be described by a combination of linear and nonlinear parts can be modeled effectively by our methods. Besides surpassing a pure tensor-only approach in all considered simulation examples, our approach reduces the necessary rank and dimensionality of the involved tensors by up to a percentile of the tensor-only approach. This results in incredibly low-complex models with exceptional performance as will be seen in the remainder of this paper. A further advantage of the proposed method is, that the memory of linear parts of the system can be put into the linear estimator(s) and therefore do not need to be included in the tensor part of the estimator. This can greatly reduce the necessary size of the tensor and therefore yields a low memory and low complexity estimator.

This paper is structured as follows Section II reviews state-of-the-art tensor algorithms and methods as well as outlines the overall problem. Section III introduces the novel approaches, while in Section IV we make a stability analysis via normalizing the proposed methods. Section V derives the complexity of all considered algorithms. Section VI shows simulation results for all considered algorithms, and Section VII concludes this paper.

II. PROBLEM DESCRIPTION

A. PRELIMINARIES AND NOTATION

There are different meanings of the term tensor. In this paper, we adhere to the widely adopted meaning as in [16], [17] that a tensor may be represented as an M-dimensional array, indexed by $i_1, i_2, i_3, \ldots, i_M$. We denote a tensor by $\mathcal{X}$. We use the notation $\circledcirc, \circledast, \odot$ to refer to the outer (tensor) product, Hadamard product and Khatri-Rao product, respectively.

A pure (also called rank-1) tensor $\dot{\mathcal{X}}$ of order M (also called M-way tensor), is the outer product of a collection of M vectors $\hat{\mathbf{a}}_m \in \mathbb{R}^{I_m \times 1}$, $\forall m \in \{1, \ldots, M\}$

$$\dot{\mathcal{X}} = \hat{\mathbf{a}}_1 \circledcirc \cdots \circledcirc \hat{\mathbf{a}}_M \tag{1}$$

which can also be written as

$$\dot{\mathcal{X}}(i_1, i_2, \ldots, i_M) = \hat{\mathbf{a}}_1(i_1) \cdots \hat{\mathbf{a}}_M(i_M) = \prod_{m=1}^{M} \mathbf{A}_m(i_m, 1) \tag{2}$$

where $\hat{\mathbf{a}}_m(i_m) = \mathbf{A}_m(i_m, 1)$.

Any M-way tensor $\mathcal{X}$ with a higher rank than one can be decomposed into a sum of rank-1 tensors

$$\mathcal{X} = \sum_{r=1}^{R} \dot{\mathcal{X}}^{(r)} = \sum_{r=1}^{R} \hat{\mathbf{a}}_1^{(r)} \circledcirc \cdots \circledcirc \hat{\mathbf{a}}_M^{(r)} \Longleftrightarrow \tag{3}$$

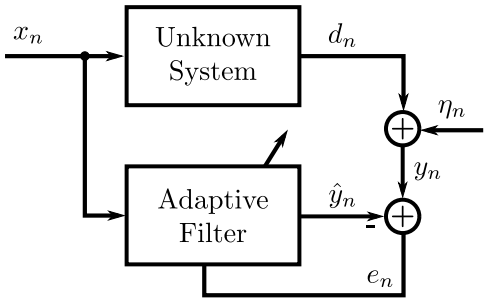

FIGURE 1. An overview of the general problem of adaptive system identification.

$$\mathcal{X}(i_1, i_2, \ldots, i_M) = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_m(i_m, r) \tag{4}$$

where $\mathbf{A}_m = [\hat{\mathbf{a}}_m^{(1)}, \ldots, \hat{\mathbf{a}}_m^{(R)}]$ and $\mathbf{A}_m \in \mathbb{R}^{I_m \times R}$. This notation allows to rewrite the decomposition with M matrices $\mathbf{A}_m$ for $1 \leq m \leq M$. If the number of rank-1 components is minimal, then the decomposition is called canonical polyadic decomposition (CPD) and the tensor rank is defined as this number R. A CPD, also called a tensor rank decomposition, can be considered as a generalization of the matrix singular value decomposition.

We introduce some common notation as in [16], [17]. Let m' be an arbitrary but fixed number $m' \in \{1, \ldots, M\}$. Initially, we introduce flattening the tensor into a matrix. This reshaping is also called matricization or unfolding. A tensor can be flattened along all the coordinates. We denote the resulting matrix by $\mathcal{Y}_{(m')}$, where the index (m') indicates on how the tensor is flattened. For example, in the three-dimensional case a cube can be cut horizontally, in front slabs, or sideways. In general stacking the resulting matrices leads to reshaping the tensor into a matrix $\mathcal{Y}_{(m')} \in \mathbb{R}^{I_1 \cdots I_{m'-1}, I_{m'+1} \cdots I_M \times I'_m}$. Moreover, we denote the Khatri-Rao product over all matrices $\mathbf{A}_m$ with $m \neq m'$ as

$$\odot_{m \neq m'} \mathbf{A}_m := \mathbf{A}_M \odot \cdots \odot \mathbf{A}_{m'+1} \odot \mathbf{A}_{m'-1} \odot \cdots \odot \mathbf{A}_1 \tag{5}$$

and, equivalently, the Hadamard product as $\circledast_{m \neq m'} \mathbf{A}_m$. We employ the short notation

$$\mathbf{Q}_{m'} := \odot_{m \neq m'} \mathbf{A}_m. \tag{6}$$

B. PROBLEM STATEMENT

The general setup is depicted in Figure 1, where the aim is to approximate an unknown system with an adaptive filter by utilizing the same input signal x_n and only observing the output y_n. Naturally, the ideal output of the unknown system d_n is subject to noise η_n to yield the overall output $y_n = d_n + \eta_n$. Besides updating the approximation of the system with each observed sample (i.e. one optimization step per time-step), the adaptive filter further assumes that the unknown system itself may not remain static over time, hence adaptation can never be turned off. Generally, the goal is to drive the error $e_n = y_n - \hat{y}_n$ as close to zero as possible.

However, due to the additive noise this, of course, can never truly equal zero.

In the past, this task was mainly achieved by trying to model the system as good as possible and adapting the parameters of the estimator accordingly, e.g. for a strictly linear system, the LMS algorithm may yield good adaptive approximations of the unknown system. However, if we assume minimal model knowledge and that the system also contains nonlinearities, traditional approaches quickly fall apart. One solution to this issue are data-based algorithms, i.e. algorithms that approximate a given system solely by observing the input and output data. One of such algorithms is tensor completion, which we expand and simplify in the remainder of this paper.

C. EXISTING ALGORITHMS

State-of-the-art algorithms for learning a tensor decomposition to approximate a given system include stochastic gradient descent (SGD) and alternating least squares (ALS), among others. In this subsection, we describe these algorithms.

1) ALTERNATING LEAST SQUARES ALGORITHM

Adopting a least squares criterion as in [16], [17], we get the following problem formulation

$$\mathbf{A}_{m',n+1} = \arg\min_{\mathbf{A}_{m'}} \left\| \mathcal{Y}_{(m')} - (\odot_{m\neq m'} \mathbf{A}_{m,n}) \mathbf{A}_{m'}^{\mathrm{T}} \right\|_{\mathrm{F}}^2 \tag{7}$$

where $\mathcal{Y}$ is a tensor containing the corresponding data points, where $\mathcal{Y}_{(m')}$ is the m' matricization of the tensor $\mathcal{Y}$, where n denotes the iteration index and where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. The minimization of the norm above leads to

$$\mathbf{a}_{m',n+1}^{(i_{m'})} = \left(\mathbf{Q}_{m',n}^{\mathrm{T}} \mathbf{Q}_{m',n}\right)^{-1} \mathbf{Q}_{m',n}^{\mathrm{T}} \mathbf{y}_{m'}^{(i_{m'})} \quad \forall m' \in \{1,\ldots,M\} \tag{8}$$

where $\left(\mathbf{a}_{m',n+1}^{(i_{m'})}\right)^{\mathrm{T}}$ is the $i_{m'}$-th row of the matrix $\mathbf{A}_{m',n+1}$ at iteration $n+1$ and $\mathbf{y}_{m'}^{(i_{m'})} = \mathcal{Y}_{(m')}(:, i_{m'})$. This algorithm iteratively adapts the matrix $\mathbf{A}_{m'}, \forall m' \in \{1,\ldots,M\}$. The update (8) is repeated until convergence.

2) STOCHASTIC GRADIENT DESCENT ALGORITHM

For the stochastic gradient descent algorithm [17], we investigate minimizing the following cost function

$$J_{\mathrm{SGD}} = \left(\mathcal{Y}(i_1, i_2, \ldots, i_M) - \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_m(i_m, r) \right)^2 =: e^2. \tag{9}$$

In order to solve the minimization problem, we calculate the derivative

$$\frac{\partial J_{\mathrm{SGD}}}{\partial \mathbf{A}_{m'}(i_{m'}, :)} = -2e \left(\circledast_{m\neq m'} \mathbf{A}_m(i_m, :) \right). \tag{10}$$

With this derivative, we get the following update formulation

$$\mathbf{A}_{m',n+1}(i_{m',n+1}, :) = \mathbf{A}_{m',n}(i_{m',n}, :) - \mu \left(\frac{\partial J_{\mathrm{SGD}}}{\partial \mathbf{A}_{m'}(i_{m'}, :)} \right)^{\mathrm{T}} \tag{11}$$

where μ is the step size. For this algorithm we randomly picked a data point $\mathcal{Y}(i_1, i_2, \ldots, i_M)$ from the available ones and only take the gradient step for those model parameters that have an effect on $\mathcal{Y}(i_1, i_2, \ldots, i_M)$.

III. PROPOSED ALGORITHM

While some of the algorithms for tensor decomposition are less complex than others, all of them have in common, that they only train a single tensor with the goal to approximate a given system model as good as possible with this setup. Consequently, if this system depends on a lot of past samples or has a high dimensional input, the resulting tensor requires high dimensions as well. This, coupled with a high number of discretization points and a high rank approximation of the tensor by matrices, can yield to a very high memory footprint of the resulting estimator. Furthermore, such an architecture would require a significant amount of complexity for updating the estimator in case of a time-variant system. Another alternative are TTs, which basically entail plugging several tensors together in the hope of reducing complexity compared to the tensor-only case. As we shall see, for certain cases this approach can be computationally demanding. However, if some knowledge about the system is available, one might be able to significantly reduce the computational demands compared to TTs and tensor only approaches.

In the following section, we incorporate such knowledge. On the one hand we can reduce both rank and dimensionality of the resulting tensors. On the other hand we can incorporate some memory in the nonlinear part, whereas this does not work reasonable for the splines, therefore, we can reach more degrees of freedom. In this paper, we investigate four major models, which are of great significance in the signal processing community: the Hammerstein model, the Wiener model and two Sandwich approaches. In Figure 2 the different cases are shown, which represent the ''Unknown System'' box in Figure 1. The linearities are always treated with a simple LMS while the nonlinearities are treated with the tensors. We start by describing the discretization of the tensor input.

A. DISCRETIZATION

Before describing the proposed method in detail, an important part to consider is that either the input signal $\mathbf{x}_n$ or the output of the LMS $\mathbf{z}_n$ serves as input to the tensor (depending on the architecture used). Naturally, both signals are discrete as we are considering digital signal processing. However, the precision of this discretization might be too high for the tensor we would like to approximate. Therefore, either signal has to pass a separate discretization step, that limits the values of

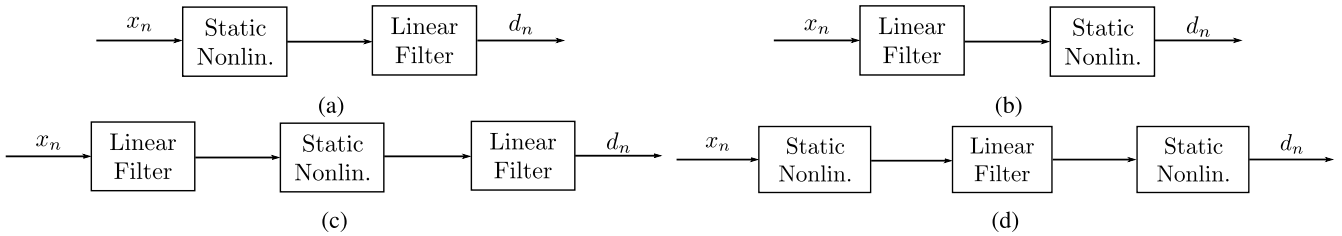

FIGURE 2. Considered system models for (a) the Hammerstein case, (b) the Wiener case, (c) a nonlinearity between two linear filters, and (d) a linear filter between two nonlinearities.

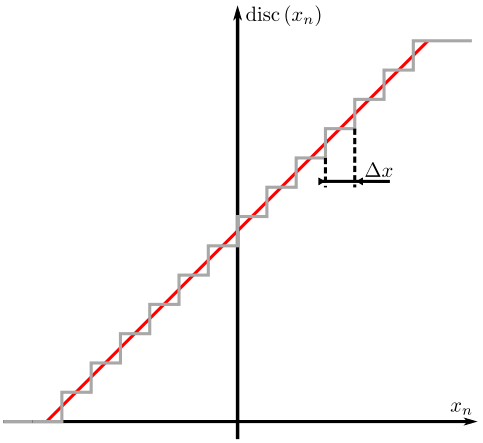

FIGURE 3. Discretization of the input signal of the tensor. The input signal x_n is mapped onto N_{Bins} equal intervals to yield the output signal disc (x_n). The red line shows discretization with an ideal, infinitely fine grid, and Δx denotes the discretization interval size.

the signal to N_{Bins} values. This is done by dividing the range between the expected maximum and minimum values of the input values in even intervals (see Fig. 3). If the maximum and/or minimum values are unknown, then a certain cut-off point can be defined. Mathematically this discretization step is given by the function disc(·)

$$\text{disc}(x_n) = \left\lfloor \frac{x_n}{\Delta x} \right\rfloor + \frac{N_{\text{Bins}}}{2} \tag{12}$$

if N_{Bins} is even and where Δx is the discretization interval.

This additional step allows to further reduce the size of the resulting approximation matrices $\mathbf{A}_n$ by reducing the number of discretization points as much as the required performance allows.

B. TENSOR-LMS (TLMS)

In this section, we combine the tensor with the LMS algorithm, as shown in Fig. 4a. The joint cost function consists of the LMS cost function with the tensor output plugged in as the LMS input

$$\begin{aligned} J_{\text{TLMS}} &= \left(y_n - \sum_{p=1}^{P} w_{n,p} z_{n-p+1} \right)^2 \\ &= \left(y_n - \sum_{p=1}^{P} w_{n,p} \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n-p+1}(i_{m,n-p+1}, r) \right)^2 \\ &=: e_n^2 \end{aligned} \tag{13}$$

where

$$\begin{aligned} \mathbf{z}_n &= \begin{pmatrix} z_n \\ \vdots \\ z_{n-P+1} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r) \\ \vdots \\ \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n-P+1}(i_{m,n-P+1}, r) \end{pmatrix} \end{aligned} \tag{14}$$

and where y_n is the desired LMS output. To derive an update for the coefficients $\mathbf{A}_{m'}$ we approximate the gradient of the cost function as

$$\mathbf{G}_{\text{TLMS},m',n} := \frac{\partial J_{\text{TLMS}}}{\partial \mathbf{z}_n} \frac{\partial \tilde{\mathbf{z}}_n}{\partial \mathbf{A}_{m'}^{\text{T}}} \tag{15}$$

with

$$z_n = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r) \tag{16}$$

$$\approx \sum_{r=1}^{R} \mathbf{A}_{m'}(i_{m',n}, r) \prod_{\substack{m=1 \\ m \neq m'}}^{M} \mathbf{A}_{m,n}(i_{m,n}, r) =: \tilde{z}_n. \tag{17}$$

Note it is important to rewrite an approximation $\tilde{z}_n$ of z_n, where in $\mathbf{A}_{m',n}$ the time is omitted, so one can take the derivative with respect to $\mathbf{A}_{m'}$, this approach can also be seen in [34].

The first and second term of (15) individually coincide with the terms of standard least mean squares (LMS) updates and SGD in (11) respectively. Taking the derivative of the total cost function J_{TLMS} with respect to one entire matrix $\mathbf{A}_{m'}$ is like described above taking the outer derivative multiplied with the inner derivative

$$\mathbf{G}_{\text{TLMS},m',n} = -2\, e_n \mathbf{S}_{m',n} \tag{18}$$

with

$$\mathbf{S}_{m',n} := \sum_{p=1}^{P} w_{n,p} \begin{pmatrix} \mathbf{0}_{i_{m',n-p+1}-1 \times R} \\ \circledast_{m \neq m'} \mathbf{A}_{m,n-p+1}(i_{m,n-p+1}, :) \\ \mathbf{0}_{I_{m'}-i_{m',n-p+1} \times R} \end{pmatrix}, \tag{19}$$

where $\mathbf{0}_{i_{m',n-p+1}-1 \times R} \in \mathbb{R}^{i_{m',n-p+1}-1 \times R}$ denotes a matrix with all elements being zero. Therefore, the update for the tensor is given by

$$\mathbf{A}_{m',n+1} = \mathbf{A}_{m',n} + 2\mu_2\, e_n \mathbf{S}_{m',n}. \tag{20}$$

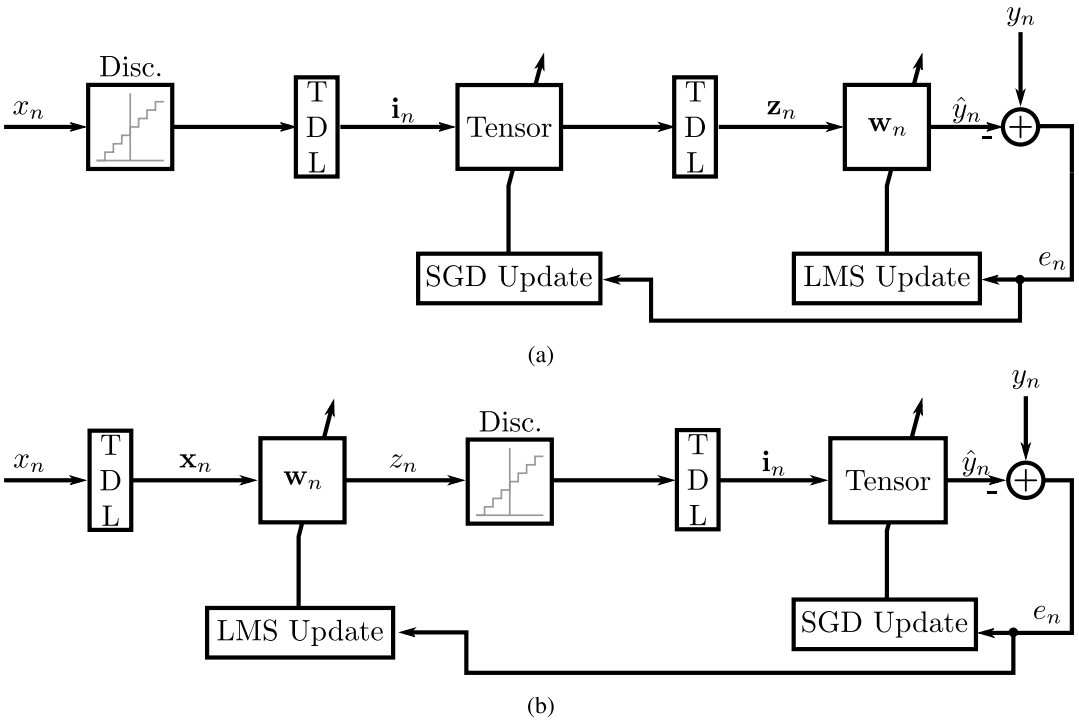

FIGURE 4. The proposed architecture for the (a) tensor-LMS case, and (b) LMS-tensor case, the sandwich approaches are a suitable combination of both architectures. The TDL blocks denote a tapped delay line.

The tensor-LMS method described in this subsection is summarized in Algorithm 1. Note that the m-th element of a vector $\mathbf{i}_n$ is indicated by $i_{m,n}$. The function tdl$(\cdot)$ denotes the tapped delay line (TDL) of the vector $\mathbf{x}_{n-1}$ and the sample x_n

$$\text{tdl}(x_n, \mathbf{x}_{n-1}) = [x_n, \mathbf{x}_{n-1}^{\mathrm{T}}(1:P-1)]^{\mathrm{T}},\ \mathbf{x}_n \in \mathbb{R}^P. \quad (21)$$

C. LMS-TENSOR (LMST)

To model a linear term followed by a nonlinearity we also propose an LMS-tensor architecture, see Fig. 4b, which is derived in the following. While the update of the matrices defining the tensor are the same as in (11), the update equation for the LMS filter before the tensor becomes a little more complex than in the previous tensor-LMS configuration. To derive the update equation, we define the cost function

$$J_{\text{LMST}} = \left(y_n - \sum_{r=1}^{R}\prod_{m=1}^{M}\mathbf{A}_{m,n}(i_{m,n}, r)\right)^2 =: e_n^2. \quad (22)$$

By applying the chain rule we get (assuming the LMS output to be denoted by z)

$$\frac{\partial J_{\text{LMST}}}{\partial \mathbf{w}_n} = \sum_{m'=1}^{M} \frac{\partial J_{\text{Ten}}}{\partial \mathbf{A}_{m',n}(i_{m',n},:)} \frac{\partial \mathbf{A}_{m',n}(i_{m',n},:)}{\partial i_{m',n}} \times \frac{\partial i_{m',n}}{\partial z_{n-m'+1}} \frac{\partial z_{n-m'+1}}{\partial \mathbf{w}_n}, \quad (23)$$

where the sum over all $m' = 1, \ldots, M$ occurs because of the product rule and $i_{m',n}$ is the m'-th entry in the TDL before the tensor at time step n in Fig. 4b. The first and last terms of this equation are straightforward to calculate and are given by

$$\frac{\partial J_{\text{LMST}}}{\partial \mathbf{A}_{m',n}(i_{m',n},:)} = -2e_n \underbrace{\left(\circledast_{m\neq m'}\mathbf{A}_{m,n}(i_{m,n},:)\right)}_{=:\mathbf{u}_{m',n}} \quad (24)$$

and

$$\frac{\partial z_{n-m'+1}}{\partial \mathbf{w}_n} = \mathbf{x}_{n-m'+1}^{\mathrm{T}}, \quad (25)$$

where $\mathbf{x}_n$ is the input of the LMS. The second and third terms in (23) require approximations. The derivative of the index i with respect to z can be approximated by the slope of the (ideal) discretization curve and is the same for all $i_{m',n}$ (cf. Fig. 3) in the form of

$$\frac{\partial i_{m',n}}{\partial z_{n-m'+1}} \approx \frac{N_{\text{Bins}}}{z_{\text{in,max}} - z_{\text{in,min}}} = \frac{1}{\Delta z}, \quad (26)$$

where N_{Bins} is the number of discretization intervals, and $z_{\text{in,max}}$ as well as $z_{\text{in,min}}$ correspond to the maximum and minimum value of the input signal, respectively. If the maximum and minimum values are not known, a cut-off point can be defined for either border. Finally, the second term in (23) can be approximated using discrete calculus to yield

$$\frac{\partial \mathbf{A}_{m',n}(i_{m',n},:)}{\partial i_{m',n}} \approx \delta\mathbf{A}_{m',n}^{\mathrm{T}} := \begin{cases} \mathbf{A}_{m',n}(i_{m',n}+1,:) - \mathbf{A}_{m',n}(i_{m',n},:) & i_{m',n} = 1 \\ \mathbf{A}_{m',n}(i_{m',n},:) - \mathbf{A}_{m',n}(i_{m',n}-1,:) & i_{m',n} = I_m \\ \frac{1}{2}\left(\mathbf{A}_{m',n}(i_{m',n}+1,:) - \mathbf{A}_{m',n}(i_{m',n}-1,:)\right) & \text{else.} \end{cases} \quad (27)$$

Algorithm 1 The Tensor-LMS Algorithm

function TLMS($x_{1,\dots,N}, y_{1,\dots,N}$)
Initialization $\mathbf{w}_1 := \mathbf{0}, \mathbf{i}_0 := \mathbf{0}, \mathbf{z}_0 := \mathbf{0}, \mathbf{A}_{m,1} := \text{randn}(I_m, R)$
  for $n = 1 : N$ **do**
    $\mathbf{i}_n = \text{tdl}(\text{disc}(x_n), \mathbf{i}_{n-1})$
    $z_n = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r)$
    $\mathbf{z}_n = \text{tdl}(z_n, \mathbf{z}_{n-1})$
    $\hat{y}_n = \mathbf{w}_n^{\mathrm{T}} \mathbf{z}_n$
    $e_n = y_n - \hat{y}_n$
    $\mathbf{w}_{n+1} = \mathbf{w}_n + \mu_1 \, e_n \mathbf{z}_n$
    $\mathbf{S}_{m',n} = \sum_{p=1}^{P} w_{n,p} \begin{pmatrix} \mathbf{0}_{i_{m',n-p+1}-1 \times R} \\ \circledast_{m \neq m'} \mathbf{A}_{m,n-p+1}(i_{m,n-p+1}, :) \\ \mathbf{0}_{I_{m'}-i_{m',n-p+1} \times R} \end{pmatrix} \quad \forall m' \in \{1, \dots, M\}$
    $\mathbf{A}_{m',n+1} = \mathbf{A}_{m',n} + 2\mu_2 \, e_n \mathbf{S}_{m',n}$
  end for
  return $\hat{y}_{1,\dots,N}$
end function

Even though both terms are approximations, simulations will show that these approximations are sufficient for the LMS update and yield good performance. Lastly, it is worth mentioning, that the approximation in (27) requires the matrix $\mathbf{A}$ to be smooth as discrete calculus would not work otherwise.

Therefore, the full update equation for the LMS part of this configuration is given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + 2\mu_1 e_n \frac{1}{\Delta z} \sum_{m'=1}^{M} \mathbf{x}_{n-m'+1} \delta \mathbf{A}_{m',n}^{\mathrm{T}} \mathbf{u}_{m',n}^{\mathrm{T}}. \quad (28)$$

The LMS-tensor method described in this subsection is summarized in Algorithm 2.

D. SANDWICH APPROACHES

Lastly, the two previous approaches can be combined to yield either an estimator with a linearity between to nonlinearities or the other way around. Of course, any combination thereof is possible and the update equations just need to be put together according to the simple versions derived above.

1) LMS-TENSOR-LMS (LMSTLMS)

In the case of two linearities enclosing a nonlinearity an LMS-tensor-LMS (LMSTLMS) system can be used for the estimation. Therefore, we define the cost function

$$J_{\text{LMSTLMS}} = \left(y_n - \sum_{p_2=1}^{P^{(2)}} w_{n,p_2}^{(2)} z_{n-p_2+1}^{(2)} \right)^2 =: e_n^2. \quad (29)$$

Furthermore, the output of the tensor becomes the input for the LMS

$$z_n^{(2)} = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r), \quad (30)$$

and the discretization of the results of the LMS are the new indices for the matrices

$$i_{m,n} = \text{disc}\left(\sum_{p_1=1}^{P^{(1)}} w_{n,p_1}^{(1)} x_{n-m+1-p_1+1} \right) \quad (31)$$

$$\approx \text{disc}\left(\sum_{p_1=1}^{P^{(1)}} w_{p_1}^{(1)} x_{n-m+1-p_1+1} \right) =: \tilde{i}_{m,n} \quad (32)$$

x denotes the input of the system and disc$(\cdot)$ denotes the function discretizing the output of the LMS to become the indices for the tensor. In this case the tensor and last LMS are updated according to Section III-B and the gradient of the first LMS filter is given by

$$\frac{\partial J_{\text{LMSTLMS}}}{\partial \mathbf{w}^{(1)}} = \frac{\partial J_{\text{LMSTLMS}}}{\partial \hat{y}_n} \sum_{p_2=1}^{P^{(2)}} \frac{\partial \hat{y}_n}{\partial z_{n-p_2+1}^{(2)}} \times \sum_{m'=1}^{M} \frac{\partial z_{n-p_2+1}^{(2)}}{\partial i_{m',n-p_2+1}} \frac{\partial \tilde{i}_{m',n-p_2+1}}{\partial \mathbf{w}^{(1)}}. \quad (33)$$

Some of the terms can only be given by an approximation. To derive an update for the coefficients $\mathbf{w}_n^{(1)}$ we approximate the gradient of the cost function as

$$\Delta \mathbf{w}_n^{(1)} = -2e_n \sum_{p_2=1}^{P^{(2)}} w_{n,p_2}^{(2)} \times \sum_{m'=1}^{M} \circledast_{m \neq m'} \mathbf{A}_{m,n-p_2+1}(i_{m,n-p_2+1}, :) \delta \mathbf{A}_{m',n-p_2+1} \times \frac{1}{\Delta z^{(1)}} \mathbf{x}_{n-p_2+1-m'+1}, \quad (34)$$

where $\delta \mathbf{A}_{m',n-p_2+1}$ corresponds to the discrete derivative in (27) for the last tensor and $\frac{1}{\Delta z^{(1)}}$ corresponds to the discrete

Algorithm 2 The LMS-Tensor Algorithm

function LMST($x_{1,\dots,N}, y_{1,\dots,N}$)
Initialization $\mathbf{w}_1 := \mathbf{0}, \mathbf{x}_0 := \mathbf{0}, \mathbf{i}_0 := \mathbf{0}, \mathbf{A}_{m,1} := \text{randn}(I_m, R)$
  for $n = 1 : N$ **do**
    $\mathbf{x}_n = \text{tdl}(x_n, \mathbf{x}_{n-1})$
    $z_n = \mathbf{w}_n^{\mathrm{T}} \mathbf{x}_n$
    $\mathbf{i}_n = \text{tdl}(\text{disc}(z_n), \mathbf{i}_{n-1})$
    $\hat{y}_n = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r)$
    $e_n = y_n - \hat{y}_n$
    $\mathbf{u}_{m',n} = \circledast_{m \neq m'} \mathbf{A}_{m,n}(i_{m,n}, :)$
    $\mathbf{w}_{n+1} = \mathbf{w}_n + 2\mu_1 e_n \frac{1}{\Delta z} \sum_{m'=1}^{M} \mathbf{x}_{n-m'+1} \delta \mathbf{A}_{m',n}^{\mathrm{T}} \mathbf{u}_{m',n}^{\mathrm{T}}$, $\delta \mathbf{A}_{m',n}$ see Eq. (27)
    $\mathbf{A}_{m',n+1} = \mathbf{A}_{m',n} + 2\mu_2 \, e_n \mathbf{u}_{m',n} \quad \forall m' \in \{1, \dots, M\}$
  end for
  return $\hat{y}_{1,\dots,N}$
end function

derivative in (26). Therefore, the update equation for the first LMS is given by

$$\mathbf{w}_{n+1}^{(1)} = \mathbf{w}_n^{(1)} - \mu_3 \Delta \mathbf{w}_n^{(1)}. \tag{35}$$

The LMS-tensor-LMS method is summarized in Algorithm 3.

2) TENSOR-LMS-TENSOR (TLMST)

If the system encloses a linear element in between two nonlinearities, a tensor-LMS-tensor approach can be used for the identification task. In this approach we define the cost function by

$$J_{\text{TLMST}} = \left(y_n - \sum_{r_2=1}^{R^{(2)}} \prod_{m_2=1}^{M^{(2)}} \mathbf{A}_{m_2,n}^{(2)}(i_{m_2,n}^{(2)}, r_2) \right)^2 =: e_n^2. \tag{36}$$

Furthermore, the discretization of the results of the LMS are the new indices for the matrices

$$i_{m_2,n}^{(2)} = \text{disc}\left(\sum_{p=1}^{P} w_{n-m_2+1,p} z_{n-m_2+1-p+1}^{(2)} \right) \tag{37}$$

and ongoing the tensor output is the input for the LMS algorithm

$$\begin{aligned} z_n^{(1)} &= \sum_{r_1=1}^{R^{(1)}} \prod_{m_1=1}^{M^{(1)}} \mathbf{A}_{m_1,n}^{(1)}(i_{m_1,n}^{(1)}, r_1) \\ &\approx \sum_{r_1=1}^{R^{(1)}} \mathbf{A}_{m_1'}^{(1)}(i_{m_1',n}, r_1) \prod_{\substack{m_1=1 \\ m_1 \neq m_1'}}^{M^{(1)}} \mathbf{A}_{m_1,n}^{(1)}(i_{m_1,n}^{(1)}, r_1) =: \tilde{z}_n^{(1)}. \end{aligned} \tag{38}$$

In this case, the LMS and the last tensor get updated according to Section III-C while the gradient of the cost function J_{TLMST} with respect to the coefficients of the first tensor is approximated by

$$\begin{aligned} \mathbf{G}_{\text{TLMST},m_1',n}^{(1)} := \frac{\partial J_{\text{TLMST}}}{\partial \hat{y}_n} \sum_{m_2'=1}^{M^{(2)}} \frac{\partial \hat{y}_n}{\partial i_{m_2',n}^{(2)}} \\ \times \sum_{p=1}^{P} \frac{\partial i_{m_2',n}^{(2)}}{\partial z_{n-m_2'+1-p+1}^{(1)}} \frac{\partial \tilde{z}_{n-m_2'+1-p+1}^{(1)}}{\partial (\mathbf{A}_{m_1'}^{(1)})^{\mathrm{T}}}. \end{aligned} \tag{39}$$

Since some of the terms can only be given by an approximation, to derive an update for the coefficients $\mathbf{A}_{m',n}^{(1)}$ we approximate the gradient of the cost function as

$$\begin{aligned} &\mathbf{G}_{\text{TLMST},m_1',n}^{(1)} \\ &= -2e_n \sum_{m_2'=1}^{M^{(2)}} (\delta \mathbf{A}_{m_2',n}^{(2)})^{\mathrm{T}} \left[\circledast_{m_2 \neq m_2'} \mathbf{A}_{m_2,n}^{(2)}(i_{m_2,n}^{(2)}, :) \right]^{\mathrm{T}} \frac{1}{\Delta z^{(2)}} \\ &\quad \times \sum_{p=1}^{P} w_{p,n-m_2'+1} \\ &\quad \times \begin{pmatrix} \mathbf{0}_{i_{m_1',n-m_2'+1-p+1}^{(1)} - 1 \times R^{(1)}} \\ \circledast_{m_1 \neq m_1'} \mathbf{A}_{m_1,n-m_2'+1-p+1}^{(1)}(i_{m_1,n-m_2'+1-p+1}^{(1)}, :) \\ \mathbf{0}_{I_{m_1'}^{(1)} - i_{m_1',n-m_2'+1-p+1}^{(1)} \times R^{(1)}} \end{pmatrix}, \end{aligned} \tag{40}$$

where $\delta \mathbf{A}_{m_2',n}^{(2)}$ corresponds to the discrete derivative in (27) for the last tensor and $\frac{1}{\Delta z^{(2)}}$ corresponds to discrete derivative in (26). Therefore, the update equation for the first tensor becomes

$$\mathbf{A}_{m_1',n+1}^{(1)} = \mathbf{A}_{m_1',n}^{(1)} - \mu_3 \mathbf{G}_{\text{TLMST},m_1',n}^{(1)}. \tag{41}$$

The tensor-LMS-tensor method is summarized in Algorithm 4.

IV. NORMALIZATION

In order to improve stability of the proposed methods, we derive normalized update equations for the TLMS and

Algorithm 3 The LMS-Tensor-LMS Algorithm

function LMSTLMS($x_{1,\dots,N}, y_{1,\dots,N}$)
Initialization $\mathbf{w}_1^{(1,2)} := \mathbf{0}, \mathbf{i}_0 := \mathbf{0}, \mathbf{x}_0 := \mathbf{0}, \mathbf{z}_0^{(2)} := \mathbf{0}, \mathbf{A}_{m,1} := \text{randn}(I_m, R)$
  for $n = 1 : N$ **do**
    $\mathbf{x}_n = \text{tdl}(x_n, \mathbf{x}_{n-1})$
    $z_n^{(1)} = \left(\mathbf{w}_n^{(1)}\right)^{\mathrm{T}} \mathbf{x}_n$
    $\mathbf{i}_n = \text{tdl}(\text{disc}(z_n^{(1)}), \mathbf{i}_{n-1})$
    $z_n^{(2)} = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{A}_{m,n}(i_{m,n}, r)$
    $\hat{y}_n = \sum_{p_2=1}^{P^{(2)}} w_{n,p_2}^{(2)} z_{n-p_2+1}^{(2)}$
    $e_n = y_n - \hat{y}_n$
    $\mathbf{z}_n^{(2)} = \text{tdl}(z_n^{(2)}, \mathbf{z}_{n-1}^{(2)})$
    $\mathbf{w}_{n+1}^{(2)} = \mathbf{w}_n^{(2)} + \mu_1 e_n \mathbf{z}_n^{(2)}$
    $\mathbf{S}_{m',n} = \sum_{p_2=1}^{P^{(2)}} w_{n,p_2}^{(2)} \begin{pmatrix} \mathbf{0}_{i_{m',n-p_2+1}-1 \times R} \\ \circledast_{m \neq m'} \mathbf{A}_{m,n-p_2+1}(i_{m,n-p_2+1}, :) \\ \mathbf{0}_{I_{m'}-i_{m',n-p_2+1} \times R} \end{pmatrix} \quad \forall m' \in \{1, \dots, M\}$
    $\mathbf{A}_{m',n+1} = \mathbf{A}_{m',n} + 2\mu_2 \, e_n \mathbf{S}_{m',n}$
    $\Delta \mathbf{w}_n^{(1)} = -2e_n \sum_{p_2=1}^{P^{(2)}} w_{n,p_2}^{(2)} \sum_{m'=1}^{M} \circledast_{m \neq m'} \mathbf{A}_{m,n-p_2+1}(i_{m,n-p_2+1}, :) \delta \mathbf{A}_{m',n-p_2+1} \frac{1}{\Delta z^{(1)}} \mathbf{x}_{n-p_2+1-m'+1}$
    $\mathbf{w}_{n+1}^{(1)} = \mathbf{w}_n^{(1)} - \mu_3 \Delta \mathbf{w}_n^{(1)}$
  end for
  return $\hat{y}_{1,\dots,N}$
end function

Algorithm 4 The Tensor-LMS-Tensor Algorithm

function TLMST($x_{1,\dots,N}, y_{1,\dots,N}$)
Initialization $\mathbf{w}_1 := \mathbf{0}, \mathbf{i}_0^{(1,2)} := \mathbf{0}, \mathbf{A}_{m,1}^{(1,2)} := \text{randn}(I_m, R)$
  for $n = 1 : N$ **do**
    $\mathbf{i}_n^{(1)} = \text{tdl}(\text{disc}(x_n), \mathbf{i}_{n-1}^{(1)})$
    $z_n^{(1)} = \sum_{r_1=1}^{R^{(1)}} \prod_{m_1=1}^{M^{(1)}} \mathbf{A}_{m_1,n}^{(1)}(i_{m_1,n}^{(1)}, r_1)$
    $\mathbf{z}_n^{(1)} = \text{tdl}(z_n^{(1)}, \mathbf{z}_{n-1}^{(1)})$
    $z_n^{(2)} = \mathbf{w}_n^{\mathrm{T}} \mathbf{z}_n^{(1)}$
    $\mathbf{i}_n^{(2)} = \text{tdl}(\text{disc}(z_n^{(2)}), \mathbf{i}_{n-1}^{(2)})$
    $\hat{y}_n = \sum_{r_2=1}^{R^{(2)}} \prod_{m_2=1}^{M^{(2)}} \mathbf{A}_{m_2,n}^{(2)}(i_{m_2,n}^{(2)}, r_2)$
    $e_n = y_n - \hat{y}_n$
    $\mathbf{u}_{m_2',n} = \circledast_{m_2 \neq m_2'} \mathbf{A}_{m_2,n}^{(2)}(i_{m_2,n}, :)$
    $\mathbf{A}_{m_2',n+1}^{(2)} = \mathbf{A}_{m_2',n}^{(2)} + 2\mu_2 e_n \mathbf{u}_{m_2',n}$
    $\mathbf{w}_{n+1} = \mathbf{w}_n + 2\mu_1 e_n \frac{1}{\Delta z^{(2)}} \sum_{m_2'=1}^{M} \mathbf{x}_{n-m_2'+1} \delta \mathbf{A}_{m_2',n}^{(2)} \mathbf{u}_{m_2',n}^{\mathrm{T}}, \qquad \delta \mathbf{A}_{m_2',n}^{(2)}$ see Eq. (27)
    $\mathbf{G}_{\text{TLMST},m_1',n}^{(1)} = -2e_n \sum_{m_2'=1}^{M^{(2)}} (\delta \mathbf{A}_{m_2',n}^{(2)})^{\mathrm{T}} \left[\circledast_{m_2 \neq m_2'} \mathbf{A}_{m_2,n}^{(2)}(i_{m_2,n}^{(2)}, :) \right]^{\mathrm{T}} \frac{1}{\Delta z^{(2)}} \sum_{p=1}^{P} w_{p,n-m_2'+1}$
    $\begin{pmatrix} \mathbf{0}_{i_{m_1',n-m_2'+1-p+1}^{(1)} - 1 \times R^{(1)}} \\ \circledast_{m_1 \neq m_1'} \mathbf{A}_{m_1,n-m_2'+1-p+1}^{(1)}(i_{m_1,n-m_2'+1-p+1}^{(1)}, :) \\ \mathbf{0}_{I_{m_1'}^{(1)} - i_{m_1',n-m_2'+1-p+1}^{(1)} \times R^{(1)}} \end{pmatrix} \quad \forall m_1' \in \{1, \dots, M^{(1)}\}$
    $\mathbf{A}_{m_1',n+1}^{(1)} = \mathbf{A}_{m_1',n}^{(1)} - \mu_3 \mathbf{G}_{\text{TLMST},m_1',n}^{(1)}$
  end for
  return $\hat{y}_{1,\dots,N}$
end function

LMST. To achieve this, we consider the a priori error e_n with e_{n+1} being approximated by the first order Taylor series (see [35]).

A. TENSOR-LMS NORMALIZATION

We start investigating the normalization of the tensor-LMS case. Therefore, we approximate the new error with the first

order Taylor expansion

$$e_{n+1} \approx e_n + \sum_{r=1}^{R} \frac{\partial \tilde{e}_n}{\partial \mathbf{A}_{m'}(:,r)} \Delta \mathbf{A}_{m'}(:,r) \tag{42}$$

with

$$\tilde{e}_n = y_n - \mathbf{w}_n^{\mathrm{T}} \tilde{\mathbf{z}}_n. \tag{43}$$

From equation (20) it follows that

$$\begin{aligned} &\Delta \mathbf{A}_{m'}(:,r) \\ &= 2\mu_2\, e_n \sum_{p=1}^{P} w_{n,p} \begin{pmatrix} \mathbf{0}_{i_{m',n-p+1}-1 \times 1} \\ \circledast_{m \neq m'} \mathbf{A}_{m,n-p+1}(i_{m,n-p+1},:) \\ \mathbf{0}_{I_{m'}-i_{m',n-p+1} \times 1} \end{pmatrix} \\ &= 2\mu_2\, e_n \mathbf{S}_{m',n}(:,r). \end{aligned} \tag{44}$$

With

$$\begin{aligned} \frac{\partial \tilde{e}_n}{\partial \mathbf{A}_{m'}(:,r)} &= -\sum_{p=1}^{P} w_{n,p} \begin{pmatrix} \mathbf{0}_{i_{m',n-p+1}-1 \times 1} \\ \circledast_{m \neq m'} \mathbf{A}_{m,n-p+1}(i_{m,n-p+1},:) \\ \mathbf{0}_{I_{m'}-i_{m',n-p+1} \times 1} \end{pmatrix}^{\mathrm{T}} \\ &= \mathbf{S}_{m',n}^{\mathrm{T}}(:,r) \end{aligned} \tag{45}$$

and (44) from (42) it follows

$$\begin{aligned} e_{n+1} &\approx e_n - 2\mu_2\, e_n \sum_{r=1}^{R} \mathbf{S}_{m',n}^{\mathrm{T}}(:,r) \mathbf{S}_{m',n}(:,r) \\ &= e_n - 2\mu_2\, e_n \sum_{r=1}^{R} \sum_{i=1}^{I_{m'}} |\mathbf{S}_{m',n}(i,r)|^2 \\ &= e_n - 2\mu_2\, e_n \left\| \mathbf{S}_{m',n} \right\|_{\mathrm{F}}^2 \end{aligned} \tag{46}$$

and therefore,

$$e_{n+1} \approx \left(1 - 2\mu_2 \left\| \mathbf{S}_{m',n} \right\|_{\mathrm{F}}^2\right) e_n. \tag{47}$$

In order to improve the convergence of the algorithm, the norm of the error e_{n+1} has to be smaller or equal than the norm of the right side of equation (47). This can be reached when

$$\left| 1 - 2\mu_2 \left\| \mathbf{S}_{m',n} \right\|_{\mathrm{F}}^2 \right| < 1. \tag{48}$$

Therefore, we get the following bound for the step size μ_2

$$0 < \mu_2 < \frac{1}{\left\| \mathbf{S}_{m',n} \right\|_{\mathrm{F}}^2}. \tag{49}$$

In Algorithm 1 this bound can be implemented by introducing the time-dependent step size

$$\mu_{2,n} = \frac{\mu_2}{\delta_2 + \left\| \mathbf{S}_{m',n} \right\|_{\mathrm{F}}^2}, \quad 0 < \mu_2 < 1. \tag{50}$$

The small regularization value $\delta_2 > 0$ limits the value of $\mu_{2,n}$ and thus helps to avoid numerical issues.

B. LMS-TENSOR NORMALIZATION

Similar to the previous approach, we develop the Taylor expansion

$$e_{n+1} \approx e_n + \frac{\partial e_n}{\partial \mathbf{w}_n} \Delta \mathbf{w}_n^{\mathrm{T}}. \tag{51}$$

Following the same reasoning as with deriving (28), the derivative in the above equation is given by

$$\frac{\partial e_n}{\partial \mathbf{w}_n} = -\sum_{m=1}^{M} \mathbf{u}_{m,n} \delta \mathbf{A}_{m,n} \frac{1}{\Delta x} \mathbf{x}_{n-m+1}^{\mathrm{T}}, \tag{52}$$

and

$$\Delta \mathbf{w}_n^{\mathrm{T}} = \mu_1 e_n \frac{1}{\Delta x} \sum_{m=1}^{M} \mathbf{x}_{n-m+1} \delta \mathbf{A}_{m,n}^{\mathrm{T}} \mathbf{u}_{m,n}^{\mathrm{T}}. \tag{53}$$

Then it follows that

$$\begin{aligned} e_{n+1} &\approx e_n - \mu_1\, e_n \overbrace{\left(\sum_{m=1}^{M} \mathbf{u}_{m,n} \delta \mathbf{A}_{m,n} \frac{1}{\Delta x} \mathbf{x}_{n-m+1}^{\mathrm{T}} \right)}^{=\mathbf{d}_n^{\mathrm{T}}} \\ &\quad \times \underbrace{\left(\frac{1}{\Delta x} \sum_{m=1}^{M} \mathbf{x}_{n-m+1} \delta \mathbf{A}_{m,n}^{\mathrm{T}} \mathbf{u}_{m,n}^{\mathrm{T}} \right)}_{=\mathbf{d}_n} \end{aligned} \tag{54}$$

$$= \left(1 - 2\mu_1 \left\| \mathbf{d}_n \right\|_2^2\right) e_n. \tag{55}$$

Hence, in order to improve convergence of the LMS filter, with the same reasoning as before, the step size μ_1 has to be bounded by

$$0 < \mu_1 < \frac{1}{\left\| \mathbf{d}_n \right\|_2^2}. \tag{56}$$

We normalize the step size in Algorithm 2 by introducing the time-dependent variable

$$\mu_{1,n} = \frac{\mu_1}{\delta_1 + \left\| \mathbf{d}_n \right\|_2^2}, \quad 0 < \mu_1 < 1, \tag{57}$$

where δ_1 is a small regularization value.

V. COMPLEXITY

Lastly, we are investigating the total complexity in terms of additions, multiplications and divisions for all methods covered in this paper. While it is well known that a normalized LMS update requires $3P + 1$ multiplications and $3P - 1$ additions per time step, the forward and backward passes of the tensors require a closer look to the equations derived in Section II.

The forward, i.e. estimation, step for a given tensor according to (4) requires RM multiplications and $R - 1$ additions. Further, a standard SGD update (backward pass), given in (11) requires R additions and MR multiplications. Similarly to the tensor-only case, the complexity for the proposed algorithms can be derived by taking a look at the corresponding update equations in Section III and are summarized in Table 1

TABLE 1. Complexity in terms of multiplications, additions and divisions of the considered algorithms depending on the parameters of the methods for both, estimation (forward) and update (backward) steps. In this table sandwich SAF 1 corresponds to the SAF with two linearities and sandwich SAF 2 denotes the SAF with two nonlinearities. The equations for all SAFs have been taken from [36], [37]. The parameters *P*, *R*, and *M* correspond to the length of the linear filter, rank and dimensonality of the tensors, respectively.

Algorithm		Mult.	Add.	Div.
LMS	Forward Path	P	$P-1$	–
	Backward Path	$3P-1$	$3P+1$	1
Tensor-Only	Forward Path	MR	$R-1$	–
	Backward Path	MR	R	–
TLMS	Forward Path	$P+RM$	$P+R-2$	–
	Backward Path	$4P+M(MR+P)$	$4P+M(2R-1)+M$	2
LMST	Forward Path	$P+RM$	$P+R-2$	–
	Backward Path	$MR+P+M(MR+P)+1$	$R+PR$	1
LMSTLMS	Forward Path	$P^{(1)}+P^{(2)}+RM+1$	$P^{(1)}+P^{(2)}+R-3$	–
	Backward Path	$M(MR+P^{(2)})+P^{(1)}+$ $P^{(2)}(4+P^{(1)}+M(MR+P^{(1)}))+1$	$5P^{(2)}+M(2R-1)+P^{(2)}(M-1)+$ $P^{(1)}M+P^{(1)}M(3R-1)-1$	3
TLMST	Forward Path	$P+R^{(1)}M^{(1)}+R^{(2)}M^{(2)}$	$P+R^{(1)}+R^{(2)}-3$	–
	Backward Path	$M^{(2)}R^{(2)}+P+M^{(2)}(M^{(2)}R^{(2)}+P)+R^{(1)}+$ $M^{(2)}(R^{(2)}+(M^{(2)}-1)R^{(2)}+R^{(1)}+(P-1)R^{(1)})+2$	$R^{(2)}+PR^{(2)}+R^{(1)}+M^{(2)}+$ $M^{(2)}((P-1)R^{(1)}+R^{(2)}-1)-1$	2
Hammerstein SAF	Total	$30P+14$	$23P+20$	1
Wiener SAF	Total	$2P+48$	$2P+67$	1
Sandwich SAF 1	Total	$2P^{(1)}P^{(2)}+P^{(1)}+22P^{(2)}+35$	$2P^{(1)}P^{(2)}+30P^{(2)}+38$	1
Sandwich SAF 2	Total	$23P+68$	$30P+81$	1

in general form. It can clearly be seen that the rank R and dimension M of the proposed architecture as well as of the tensor-only approach directly affect the required number of operations per sample. While for the tensor-only solution, the complexity grows linearly with rank R and dimension M, the complexity for the proposed architecture grows with M^2 and linearly with the rank R. The number of required operations in the forward and backward passes of all SAF architectures is given in [36], [37].

VI. SIMULATIONS

In order to compare the proposed concepts with a tensor-only approach as well as SAFs, the same system models used in [33] are used by utilizing the SAF toolbox [38]. Further, all previously discussed architectures (TLMS, LMST, LMSTLMS and TLMST) are evaluated for their performance. The settings used for all six simulations are shown in Table 2 and the performance curves are depicted in Fig. 5. The simulation parameters (step-size, rank, etc.) have been chosen empirically by trial and error so that all algorithms perform the best. All experiments use a signal-to-noise ration of 10 dB, meaning that the desired signal lies 10 dB above the noise floor, as in [33]. As mentioned in Section II-B, the overall goal of all considered algorithms is to minimize the error e_n, which however, includes the noise signal η_n by definition. Therefore the metric used for evaluation is the mean-square-error (MSE) defined as

$$\mathrm{MSE}_{\mathrm{dB}} = 10\log_{10}\left(\frac{1}{L}\sum_{l=1}^{L}\left(d_n^{(l)}-\hat{y}_n^{(l)}\right)^2\right), \tag{58}$$

where $d_n^{(l)}$ is the desired signal, $\hat{y}_n^{(l)}$ is the estimate at time n of the l-th run (i.e., l-th repetition) and L is the total number of runs of a given experiment. Therefore, the noise η_n is not included in the evaluation metric, and an ideal estimator would yield a value of $-\infty$ dB. Further, to show the adaptive nature of the considered algorithms, the linear part of the unknown system changes after half of the simulation time in the first four experiments. This can, for example, be caused by temperature drifts within the system or the need to switch to other linear filters during operation. Lastly, as mentioned in Algorithms 1 to 4, the weights and TDLs of all adaptive filters (including SAF) are initialized with zeros while the tensors are initialized with zero mean and unit variance elements (including the tensor-only approach).

A. EXPERIMENT 1

The first system follows a Hammerstein model, i.e. the system consists of a static nonlinearity followed by a linear part. Therefore, the TLMS architecture is used for this experiment and compared to the corresponding SAF architecture as well as a classical tensor-only approximation.

This system can be described by a static nonlinearity in the form of

$$y_n^{\mathrm{nl}} = \frac{2x_n}{1+|x_n|^2}, \tag{59}$$

which imitates the saturation behavior of a power amplifier in a satellite communications scenario [39] (like the second experiment in [33]). Followed by an unknown linear finite impulse response (FIR) filter $\mathbf{w}_0 \in \mathbb{R}^7$, which models a part of the transmission path. The input signal follows the relationship

$$x_n = ax_{n-1} + \sqrt{1-a^2}\nu_n, \tag{60}$$

where ν_n represents standard noise and $0 \leq a < 1$ determines the level of correlation between adjacent samples [33]. The performance of all considered approaches can be seen in Fig. 5a for the SAF, the TLMS approach with rank one and dimensionality one as well as for a tensor-only

TABLE 2. Simulation settings for all different algorithms used in the experiments, where len(·) denotes the length of a vector.

Experiment	Tensor 1			Tensor 2			LMS 1		LMS 2		Tensor-Only			SAF				
	μ	R	M	μ	R	M	μ	len($\mathbf{w}_1$)	μ	len($\mathbf{w}_2$)	μ	R	M	μ_{Spl}	μ_{LMS1}	μ_{LMS2}	len($\mathbf{w}_1$)	len($\mathbf{w}_2$)
1	0.009	1	1	-	-	-	0.009	7	-	-	0.05	50	7	0.001	0.001	-	7	-
2	0.1	1	1	-	-	-	0.0075	5	-	-	0.01	50	5	0.008	0.008	-	5	-
3	0.075	1	1	0.005	1	1	0.001	7	-	-	0.04	25	7	0.01	0.01	-	7	-
4	0.2	10	1	-	-	-	0.006	7	0.05	7	0.01	50	13	0.004	0.005	0.009	7	7
5	0.008	1	3	-	-	-	0.005	7	-	-	0.025	100	9	0.008	0.0001	-	7	-
6	0.05	3	3	-	-	-	0.002	7	-	-	0.05	200	8	0.001	0.001	-	7	-

model with rank 50 and dimensionality equal to the length of $\mathbf{w}_0$. As can be seen, the proposed solution significantly outperforms the SAF. Nevertheless, both approaches react quickly and as expected, to the changing linear subsystem. While the tensor-only approach not only requires significantly higher rank and dimensionality, but also performs worse than the other approaches. This algorithm does not adapt to the changed system appropriately and it performs worse after the change.

B. EXPERIMENT 2

The next experiment follows a Wiener model, where the general setup is the same as in the previous experiment with the exception that the nonlinearity occurs after the linear filter. Therefore, this simulation makes use of the LMST architecture with rank R and dimensionality M of the tensor set to one. The tensor-only approach, requires a rank of 50, as experiments demonstrated, and a dimensionality equal to the length of $\mathbf{w}_0$. The results of this simulation can be seen in Fig. 5b where it can be observed that the proposed solution again, achieves comparable performance to the SAF architecture and the tensor-only approach slightly worse performance. Furthermore, all three algorithms adapt to the changed system as expected and yield similar performance as before the change.

C. EXPERIMENT 3

In order to examine the performance of a combined Wiener-Hammerstein model, this experiment uses a system model where an FIR filter is placed between two nonlinearities. The nonlinear elements again are described by (59). The TLMST architecture is used in this simulation. The results in Fig. 5c show that the proposed approach achieves slightly worse performance than the SAF architecture (around 2 dB less) while the tensor-only solution requires significantly higher rank and dimensionality and again performs worse than the other two approaches. Furthermore, the tensor-only approach does not respond to the changed system like the other two architectures. Interestingly, the proposed solution seems to yield better performance after the linear part has changed and comes closer to the performance of the SAF.

D. EXPERIMENT 4

This simulation covers the case of a nonlinearity placed between two FIR filters. Again, the nonlinearity has the form as in (59) and the linear parts are defined by $\mathbf{w}_0$ and $\mathbf{w}_1$. This model fully covers the satellite communications scenario with two paths and one amplifier [33], [39]. Therefore, this problem requires the LMSTLMS approach. The results of this simulation can be seen in Fig. 5d where the dimensionality is set to one and the rank of the tensor is set to ten for the proposed solution. The proposed approach performs approximately 1 dB worse than the SAF before the linearity changes while yielding comparable performance after the change. Again, the tensor-only approach requires higher dimensionality and rank as the proposed architecture and performs significantly worse than the other two algorithms. However, the tensor-only approach is unaffected by the changing system, as it just continues converging to a solution.

E. EXPERIMENT 5

Having covered simple nonlinearities for both Wiener and Hammerstein models, this simulation has the same setup as experiment 2 with the exception that the nonlinearity has the form

$$y_n^{\text{nl}} = \sin^2(x_n) + \sin^3(x_{n-1}) + \sin^4(x_{n-2}), \tag{61}$$

i.e., the output of the nonlinearity at time n depends on the current and last two input values of the system's nonlinear part. In other words, the nonlinear part has some memory. This model shall show the performance on a system that has not been modeled in a way that is ideal for SAFs. The input signal x_n is modeled according to (60). The linear filter of the system remains the same as before. Experiments have shown that a single SAF, like in experiments 1 and 2, cannot handle a nonlinearity with memory, therefore the sandwich approach from experiment three has been used in the evaluation. It can be seen, that all architectures achieve around equal performance, and that the LMST requires significantly less parameters than a tensor-only approach (cf. Table 2).

F. EXPERIMENT 6

The last experiment has the same setup as experiment 5, but the location of linear and nonlinear parts has been swapped to model the Hammerstein case. Again, a simple SAF with one linearity and one nonlinearity cannot handle a nonlinearity with memory, therefore, the evaluation shown in Fig. 5f uses a sandwich SAF approach with two nonlinearities. Also, for this case, it can be seen that the proposed method outperforms the SAF architecture significantly, while requiring less complexity. The tensor-only approach yields comparable performance to the proposed method, although the convergence speed is significantly slower.

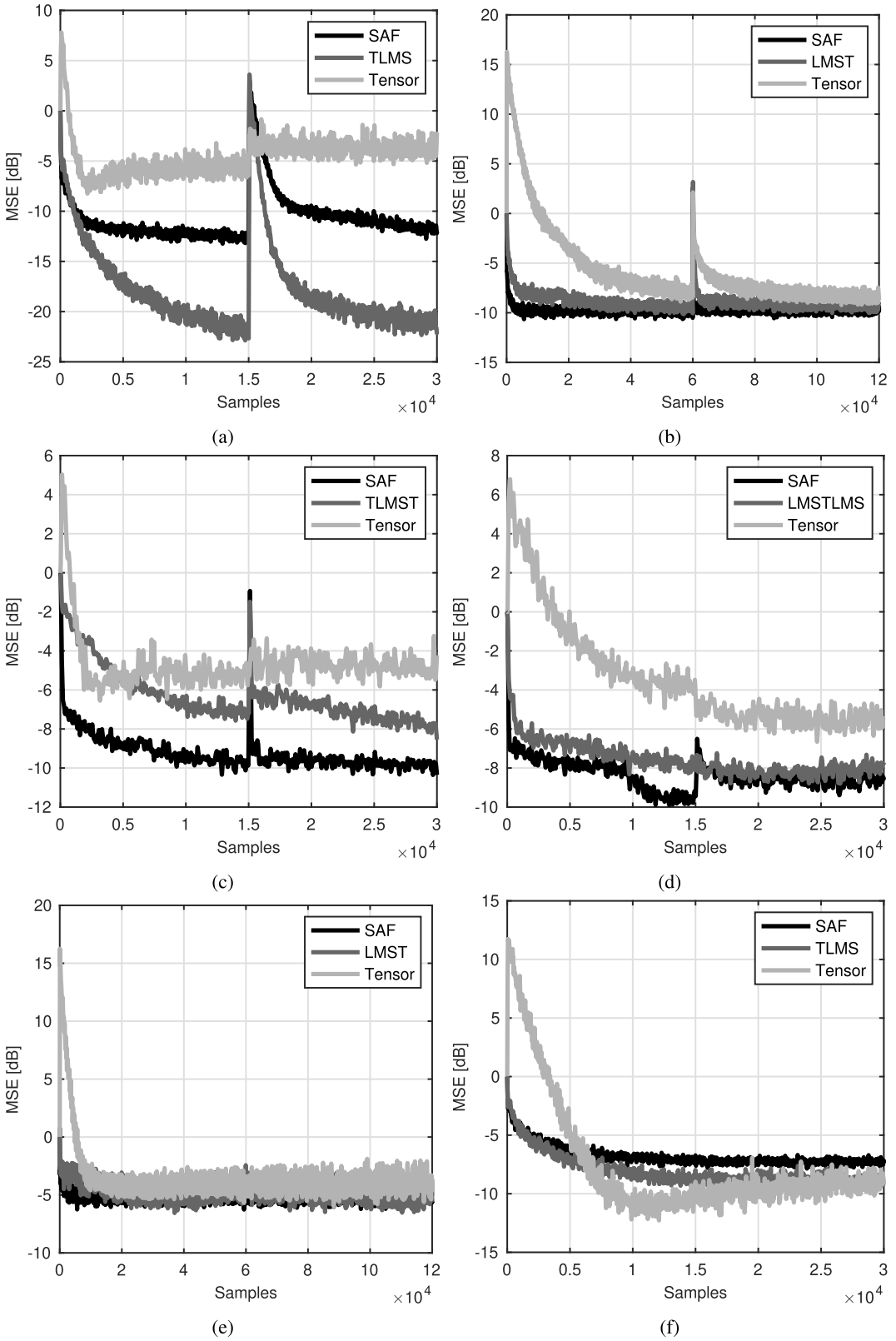

FIGURE 5. MSE convergence for all considered algorithms for (a) the Hammerstein case, (b) the Wiener case, (c) a linear filter between two nonlinearities, (d) a nonlinearity between two linear filters, (e) the Wiener case with a nonlinearity depending on past samples, and (f) the Hammerstein case with a nonlinearity containing memory.

TABLE 3. Complexity in terms of multiplications, additions and divisions of the considered algorithms in all experiments conducted.

Experiment	Operation	Tensor-only	Proposed	SAF
1	Mult.	700	45	224
	Add.	99	36	181
	Div.	-	2	1
2	Mult.	500	19	58
	Add.	99	11	30
	Div.	-	1	1
3	Mult.	350	36	229
	Add.	49	21	291
	Div.	-	2	1
4	Mult.	650	146	294
	Add.	49	284	346
	Div.	-	3	1
5	Mult.	1800	51	229
	Add.	199	15	291
	Div.	-	1	1
6	Mult.	2800	93	229
	Add.	399	54	291
	Div.	-	2	1

G. COMPARISON

Table 3 shows the required number of operations for the specific cases considered in the experiments. It can be seen that the tensor-only model requires the most operations in all simulations while all proposed architectures are the cheapest architecture to implement and the SAFs lie somewhere in between. This comparison shows how drastically the required number of operations can be reduced compared to a tensor-only approach, with up to 18 times less multiplications and 13 times less additions.

In terms of convergence speed, our approaches either outperform or yield similar convergence time than SAFs and result in similar end-performance. Especially in cases where the nonlinearity has memory, the advantage of using tensor-based methods can be seen over SAFs. This results from the latter not being able to cope with memory in the nonlinearities. Futher, it has to be noted, that the systems in experiments one through four have been chosen in a way to be an ideal system do model via SAFs, while the last two experiments make use of a more complex nonlinearity that the SAFs are not specifically designed to cope with. Additionally, the slightly worse performance, compared to SAFs could yield from the approximations made when deriving the gradient through a tensor. If a system yields in relatively smooth matrices $\mathbf{A}_n$, or if a smoothness constraint is enforced (e.g. [17]), the derivative (27) could be more accurate and yield better performance overall. However, this is subject of future work.

Overall it can be said, that our proposed method is a low-cost architecture that can achieve on-par performance with state-of-the-art methods for all conducted experiments.

H. DISCUSSION: RANK AND DIMENSION

As mentioned before, the dimension M of the tensors (in the proposed as well as the tensor-only architectures) directly correlates with the memory length of the system and therefore should always be chosen as such (if the memory length is known). As can be seen in Figure 6, if the dimension of the tensor is chosen poorly the performance drastically decreases. Since the dimension of the tensor-only approach is big enough to detect changes around the perfect value of 8 clearer, this simulation was carried out for this algorithm. The results however directly transfer to the proposed methods. Due to the overall size of the tensor, this approach compensates for too low values M to some extent. Choosing it too high rapidly leads to very high MSE values.

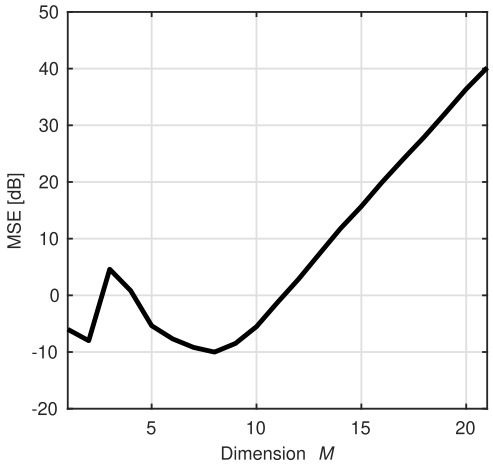

FIGURE 6. The influence of improperly chosen dimension M on the performance of the tensor-only architecture in the scenario considered in experiment 6.

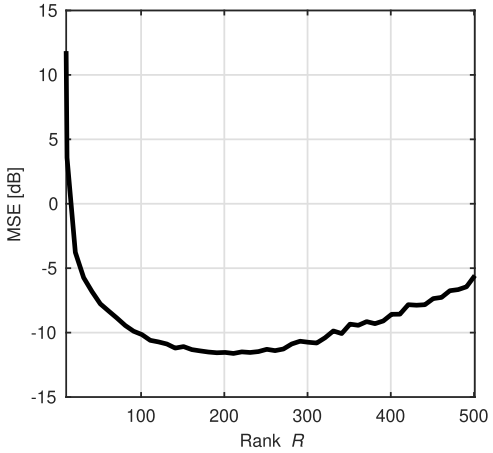

FIGURE 7. The influence of improperly chosen rank R on the performance of the tensor-only architecture in the scenario considered in experiment 6.

In comparison, the rank R of the tensors cannot be easily indicated by a parameter of the system and must be found via trial and error. Therefore, it is considered as a hyper-parameter of the model. This aspect can be seen nicely in Figure 7. Therefore, the tensor-only approach was evaluated on the setup from experiment 6 with the rank taking values from 10 to 500 in steps of 10. As can be seen, choosing the parameter R too high or too low results in a bad performance, and the goal is to find the right value. The tensor-only

approach was chosen for this analysis. Since the required rank for a good performance in experiment 6 is quite high, namely 200, there is still space on both sides to perform a meaningful sweep. However, our investigation showed, that this also yields for our proposed methods. If the tensor is required to be bigger in size (and rank) it is mentioned explicitly in the considered scenarios.

VII. CONCLUSION

In this paper we reviewed state-of-the-art algorithms to learn tensors the representation of a system and introduced a novel concept to reduce both, the required rank and dimensionality of the tensor. This approach not only greatly reduces the complexity of the architecture, but also considerably outperforms a tensor-only model in most considered test cases. We further showed that the proposed architecture can either surpass or approximately match the performance of SAF in all considered constellations. Additionally, a complexity analysis has been carried out for all considered architectures and it has been shown that the proposed approaches require significantly less operations per time step than all other methods. Future interests include a hardware implementation and further investigations on more complex signals and systems. Further, it shall be investigated if regularization and smoothness constraints on the $\mathbf{A}_n$ matrices (like e.g. in [17]) could improve performance and complexity of the proposed method.

ACKNOWLEDGMENT

Christina Auer and Oliver Ploder contributed equally to this work.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[2] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[3] K. Burse, R. N. Yadav, and S. C. Shrivastava, "Channel equalization using neural networks: A review," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 40, no. 3, pp. 352–357, May 2010.

[4] O. Ploder, O. Lang, T. Paireder, and M. Huemer, "An adaptive machine learning based approach for the cancellation of second-order-intermodulation distortions in 4G/5G transceivers," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Honolulu, HI, USA, Sep. 2019, pp. 1–7.

[5] A. Balatsoukas-Stimming, "Non-linear digital self-interference cancellation for in-band full-duplex radios using neural networks," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Kalamata, Greece, Jun. 2018, pp. 1–5.

[6] O. Ploder, C. Motz, T. Paireder, and M. Huemer, "A neural network approach for the cancellation of the second-order-intermodulation distortion in future cellular RF transceivers," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Nov. 2019, pp. 1144–1148.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2001.

[8] M. A. H. Shaikh and K. Barbe, "Initial estimation of wiener-hammerstein system with random forest," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, Auckland, New Zealand, May 2019, pp. 1–6.

[9] C. Auer, K. Kostoglou, T. Paireder, O. Ploder, and M. Huemer, "Support vector machines for self-interference cancellation in mobile communication transceivers," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, Antwerp, Belgium, May 2020.

[10] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004. [Online]. Available: https://doi.org/10.1023/B:STCO.0000035301.49549.88

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

[12] C. Auer, A. Gebhard, C. Motz, T. Paireder, O. Ploder, R. S. Kanumalli, A. Melzer, O. Lang, and M. Huemer, "Kernel adaptive filters: A panacea for self-interference cancellation in mobile communication transceivers?" in *Proc. Comput. Aided Syst. Theory (EUROCAST)* (Lecture Notes in Computer Science). Las Palmas de Gran Canaria, Spain: Springer, Feb. 2019, pp. 36–43.

[13] W. Liu, J. Principe, and S. Haykin, *Kernel Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2010.

[14] R. A. Kennedy and P. Sadeghi, *Hilbert Space Methods in Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2013.

[15] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, and C. Caiafa, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[16] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017. [Online]. Available: http://arxiv.org/abs/1607.01668.

[17] N. Kargas and N. D. Sidiropoulos, "Nonlinear system identification via tensor completion," 2019, *arXiv:1906.05746*. [Online]. Available: http://arxiv.org/abs/1906.05746

[18] M. Bousse, O. Debals, and L. De Lathauwer, "Tensor-based large-scale blind system identification using segmentation," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5770–5784, Nov. 2017.

[19] G. Favier and A. Y. Kibangou, "Tensor-based methods for system identification," *Int. J. Sci. Techn. Autom. Control Comput. Eng. (STA)*, vol. 3, no. 1, pp. 840–869, Jul. 2008.

[20] M. Sørensen and L. De Lathauwer, "Tensor decompositions with block-Toeplitz structure and applications in signal processing," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Pacific Grove, CA, USA, Nov. 2011, pp. 454–458.

[21] C. A. Fernandes, G. Favier, and J. C. M. Mota, "Blind identification of multiuser nonlinear channels using tensor decomposition and precoding," *Signal Process.*, vol. 89, no. 12, pp. 2644–2656, Dec. 2009.

[22] A. Kibangou and G. Favier, "Blind joint identification and equalization of Wiener-Hammerstein communication channels using PARATUCK-2 tensor decomposition," in *Proc. 15th Eur. Signal Process. Conf.*, Poznan, Poland, Sep. 2007, pp. 1516–1520.

[23] D. Nion, K. N. Mokios, N. D. Sidiropoulos, and A. Potamianos, "Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 6, pp. 1193–1207, Aug. 2010.

[24] D. Nion and N. D. Sidiropoulos, "Tensor algebra and multidimensional harmonic retrieval in signal processing for MIMO radar," *IEEE Trans. Signal Process.*, vol. 58, no. 11, pp. 5693–5705, Nov. 2010.

[25] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2377–2388, Aug. 2000.

[26] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," in *Proc. 7th Eur. Conf. Comput. Vis. (ECCV)*, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Copenhagen, Denmark: Springer, May 2002, pp. 447–460.

[27] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases (ECML PKDD)*, Bristol, U.K., Sep. 2012, pp. 521–536.

[28] C.-Y. Ko, K. Batselier, W. Yu, and N. Wong, "Fast and accurate tensor completion with total variation regularized tensor trains," 2018, *arXiv:1804.06128*. [Online]. Available: http://arxiv.org/abs/1804.06128

[29] K. Batselier, C.-Y. Ko, and N. Wong, "Extended Kalman filtering with low-rank tensor networks for MIMO volterra system identification," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Nice, France, Dec. 2019, pp. 7148–7153.

[30] K. Batselier and N. Wong, "Matrix output extension of the tensor network Kalman filter with an application in MIMO volterra system identification," 2017, *arXiv:1708.05156*. [Online]. Available: http://arxiv.org/abs/1708.05156
[31] R. Karagoz and K. Batselier, "Nonlinear system identification with regularized tensor network B-splines," 2020, *arXiv:2003.07594*. [Online]. Available: http://arxiv.org/abs/2003.07594
[32] N. Wiener, *Nonlinear Problems in Random Theory*. Cambridge, MA, USA: MIT Press, 1958.
[33] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Nonlinear spline adaptive filtering," *Signal Process.*, vol. 93, no. 4, pp. 772–783, Apr. 2013.
[34] A. Gebhard, "Self-interference cancellation and rejection in FDD RF-transceivers," Ph.D. dissertation, Inst. Signal Process., Johannes Kepler Univ. Linz, Linz, Austria, 2019.
[35] A. I. Hanna and D. P. Mandic, "A fully adaptive normalized nonlinear gradient descent algorithm for complex-valued nonlinear adaptive filters," *IEEE Trans. Signal Process.*, vol. 51, no. 10, pp. 2540–2549, Oct. 2003.
[36] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Novel cascade spline architectures for the identification of nonlinear systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 7, pp. 1825–1835, Jul. 2015.
[37] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Spline adaptive filters: Theory and applications," in *Adaptive Learning Methods for Nonlinear System Modeling*, D. Comminiello and J. C. Príncipe, Eds. Oxford, U.K.: Butterworth-Heinemann, 2018, ch. 3, pp. 47–69. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978012812976000004X
[38] M. Scarpiniti. (Apr. 2020). *SAF-Toolbox*. [Online]. Available: https://github.com/ispamm/SAF-Toolbox
[39] T. M. Panicker, V. J. Mathews, and G. L. Sicuranza, "Adaptive parallel-cascade truncated volterra filters," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2664–2673, Oct. 1998.

CHRISTINA AUER (Member, IEEE) was born in Linz, Austria, in 1987. She received the master's degree in applied mathematics from Johannes Kepler University Linz, in 2012. She is currently pursuing the Ph.D. degree with Johannes Kepler University Linz. She partly wrote her master thesis in image denoising at UCLA. After graduation, she joined Linz Center of Mechatronics (LCM), where she was working in the research and development, especially with Kalman Filtering for localization. Since September 2017, she has been a member of the Institute of Signal Processing, Johannes Kepler University Linz. She is also a member of the CD Laboratory for Digitally Assisted RF Transceivers for Future Mobile Communications. Her research interests include self-interference cancellation using kernel adaptive filtering and machine learning.

OLIVER PLODER (Graduate Student Member, IEEE) was born in Graz, Austria, in 1992. He received the bachelor's degree in information electronics from Johannes Kepler University Linz, in 2016, and the master's degree in telecommunication engineering with a focus on wireless communications from the Universitat Politècnica de Catalunya (BarcelonaTech), in 2018. He is currently pursuing the Ph.D. degree with Johannes Kepler University Linz, focusing his research on receiver interference cancellation for LTE and LTE-A RF transceiver systems by means of machine learning. His master thesis was written in cooperation with the Networked and Embedded Systems Laboratory (NESL), UCLA, on the topic of secure state estimation in cyber-physical systems (CPS). Since September 2018, he has been a member of the Institute of Signal Processing, Johannes Kepler University Linz.

THOMAS PAIREDER (Graduate Student Member, IEEE) received the bachelor's degree (Hons.) in information electronics and the master's degree (Hons.) in electronics and information technology from Johannes Kepler University Linz (JKU), Linz, Austria, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Institute of Signal Processing. His master's thesis was focused on signal processing in cooperation with the Research Center for Non Destructive Testing GmbH. In his thesis, he implemented a real-time processing system for laser-ultrasonic signals. His topic is receiver interference cancellation by means of adaptive signal processing methods.

PÉTER KOVÁCS received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Eötvös Loránd University (ELTE), Budapest, Hungary, in 2008, 2010, and 2016, respectively. During his Ph.D. studies, he spent five months as a Visiting Researcher at the Department of Signal Processing, Tampere University of Technology, Finland. In 2016, he was promoted to an Assistant Professor at the Department of Numerical Analysis, ELTE. Since March 2018, he has been a Postdoctoral Researcher at the Institute of Signal Processing, Johannes Kepler University Linz, Austria. His research interests include optimization, biomedical signal modeling, and analysis and implementation of numerical algorithms. He received the Farkas Gyula Prize in applied mathematics from János Bolyai Mathematical Society, in 2016.

OLIVER LANG (Member, IEEE) was born in Schärding, Austria, in 1987. He received the bachelor's degree in electrical engineering and the master's degree (Hons.) in microelectronics from Vienna University of Technology, in 2011, and the Ph.D. degree from Johannes Kepler University Linz, in 2018. The topic of his master's thesis was the development and analysis of models for a scanning microwave microscope. From 2014 to 2018, he was a University Assistant at the Institute of Signal Processing, Johannes Kepler University Linz. The title of his dissertation is "Knowledge-Aided Methods in Estimation Theory and Adaptive Filtering," where he invented several interesting estimators and adaptive filters for special applications. From 2018 to 2019, he worked at DICE GmbH, Linz, which is a subsidiary company of Infineon Austria GmbH. Since March 2019, he has been a University Assistant with Ph.D. at the Institute of Signal Processing.

MARIO HUEMER (Senior Member, IEEE) received the Dipl.-Ing. and Dr.Techn. degrees from Johannes Kepler University (JKU) Linz, Austria, in 1996 and 1999, respectively. After holding positions in industry and academia he became an Associate Professor at the University of Erlangen–Nuremberg, Germany, from 2004 to 2007, and a Full Professor at Klagenfurt University, Austria, from 2007 to 2013. Since September 2013, he has been heading the Institute of Signal Processing, JKU, Linz, as a Full Professor. Since 2017, he has been a Co-Head of the Christian Doppler Laboratory for Digitally Assisted RF Transceivers for Future Mobile Communications. His research interests include statistical and adaptive signal processing, signal processing architectures, and mixed signal processing with applications in information and communications engineering, radio frequency transceivers for communications and radar, sensor and biomedical signal processing. He received the dissertation awards from the German Society of Information Technology (ITG) and the Austrian Society of Information and Communications Technology (GIT), the Austrian Kardinal Innitzer Award in natural sciences, and the German ITG Award.