algorithms

Article

A Recursive Least-Squares Algorithm for the Identification of Trilinear Forms

Camelia Elisei-Iliescu [1], Laura-Maria Dogariu [1], Constantin Paleologu [1,*], Jacob Benesty [2], Andrei-Alexandru Enescu [1] and Silviu Ciochină [1]

[1] Department of Telecommunications, University Politehnica of Bucharest, 1-3, Iuliu Maniu Blvd., 061071 Bucharest, Romania; cameliailiescu27@gmail.com (C.E.-I.); ldogariu@comm.pub.ro (L.-M.D.); aenescu@gmail.com (A.-A.E.); silviu@comm.pub.ro (S.C.)

[2] INRS-EMT, University of Quebec, Montreal, QC H5A 1K6, Canada; benesty@emt.inrs.ca

* Correspondence: pale@comm.pub.ro

Received: 23 March 2020; Accepted: 27 May 2020; Published: 1 June 2020

Abstract: High-dimensional system identification problems can be efficiently addressed based on tensor decompositions and modelling. In this paper, we design a recursive least-squares (RLS) algorithm tailored for the identification of trilinear forms, namely RLS-TF. In our framework, the trilinear form is related to the decomposition of a third-order tensor (of rank one). The proposed RLS-TF algorithm acts on the individual components of the global impulse response, thus being efficient in terms of both performance and complexity. Simulation results indicate that the proposed solution outperforms the conventional RLS algorithm (which handles only the global impulse response), but also the previously developed trilinear counterparts based on the least-mean-squares algorithm.

Keywords: adaptive filters; recursive least-squares (RLS) algorithm; system identification; tensor decomposition; trilinear forms

1. Introduction

Currently, there is an increasing interest in developing methods and algorithms that exploit tensor decompositions and modelling [1,2]. These techniques become of significant importance in many real-world scenarios, e.g., when dealing with large amounts of data, processing multidimensional signals, or solving high-dimensional system identification problems. Many important applications rely on such tensor-based techniques, which can be successfully used in the fields of big data [3], source separation [4], machine learning [5], multiple-input multiple-output (MIMO) communication systems [6], and beamforming [7].

Tensor decompositions and their related applications are frequently addressed based on multilinear signal processing techniques [8,9]. For example, in the context of system identification scenarios, the problems can be formulated in terms of identifying multilinear forms. As particular cases, we can mention the bilinear and trilinear forms, where the decomposition is performed using two and three components, respectively. Since the Wiener filter and adaptive algorithms represent popular methods to address system identification problems, their applicability was also extended to the multilinear framework. Among the recent related works, we can mention the iterative Wiener filter for bilinear forms [10] and the subsequent adaptive filtering methods [11–13], together with their extensions to trilinear forms [14–17].

In this context, the work in [17] provided a system identification framework based on tensor decomposition, which was suitable for the trilinear approach. This work presents the iterative Wiener filter and least-mean-squares (LMS) adaptive algorithms tailored for trilinear forms. Among those, the normalized LMS (NLMS) algorithm for trilinear forms, namely NLMS-TF, represents a practical solution in terms of both performance and complexity. Nevertheless, it is known that the recursive least-squares (RLS) algorithm [18] could provide improved performance as compared to the LMS-based algorithms, especially in terms of the convergence rate. This represents the motivation behind the solution proposed in this paper, which targets the development of the RLS algorithm for trilinear forms, namely RLS-TF. Therefore, the goal of this work is mainly twofold. First, we aim to design a faster converging algorithm as compared to the recently developed NLMS-TF. Second, we intend to outline the performance features of the RLS-TF algorithm as compared to the conventional RLS benchmark.

Besides the conventional adaptive algorithms (which usually act as supervised methods), we should also note that unsupervised algorithms can be used in conjunction with tensor-based approaches, e.g., [19,20]. In this context, we can mention the single-channel blind source separation problem, which targets the identification of individual source signals from a single mixture recording. Such an approach can be found in [19], where the parameters of a so-called "imitated-stereo" mixture model (i.e., one real and the other virtual microphones, which results in an artificial mixing system of dual channels) were found by applying tensor estimation and exploiting sparsity features. The method proposed in [20] also exploits sparsity (using the Gibbs distribution) for multichannel source separation, by solving the underdetermined convolutive mixture separation, while considering the reverberations of the surrounding environment. Another appealing field where such tensor-based techniques can be applied is video processing. For example, unsupervised algorithms can be used for detecting anomalies in a video sequence, e.g., detecting micro defects while employing a thermography imaging system [21].

Currently, in many real-world applications related to MIMO systems, speech processing, and image/video processing, the received signals are tensors or can be grouped in a tensorial form. Consequently, as outlined before, utilizing estimation techniques from tensor algebra is beneficial. Moreover, in most of these applications, the underlying parameters to be estimated are sparse, so that specific features could be exploited, thus bringing additional advantages, especially for systems of large dimensions. The tensor-based RLS algorithm proposed in this paper represents an improved solution (in terms of convergence rate) as compared to the existing solution based on the NLMS algorithm [17], but also an efficient version (in terms of computational complexity) as compared to the conventional RLS algorithm [18].

The rest of the paper is organized as follows. Section 2 provides a background on third-order tensors, which represents the framework of trilinear forms. The proposed RLS-TF algorithm is developed in Section 3. Simulation results are provided in Section 4, outlining the main performance features of the designed solution. Finally, Section 5 concludes this paper and discusses several perspectives for future works.

2. Third-Order Tensors

In this section, we provide a brief summary on tensors, outlining the main definitions and operations, while also establishing the notation. A tensor can be defined as a multidimensional array of data [22,23]. For example, a matrix and a vector can be referred to as second- and first-order tensors, respectively. Tensors of order three or higher are called higher order tensors. In the following, the notation used for a tensor, a matrix, a vector, and a scalar is $\boldsymbol{\mathcal{A}}$, $\mathbf{A}$, $\mathbf{a}$, and a, respectively. In this work, we are focusing only on real-valued third-order tensors, i.e., $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{L_1 \times L_2 \times L_3}$, so that the array dimension is $L_1 \times L_2 \times L_3$.

The entries of a tensor can be referred to by using multiple indices. In our case, for a third-order tensor, the first and second indices l_1 (with $l_1 = 1, 2, \ldots, L_1$) and l_2 (with $l_2 = 1, 2, \ldots, L_2$) correspond to the row and column, respectively (like in a matrix); in addition, the third index l_3 (with $l_3 = 1, 2, \ldots, L_3$) corresponds to the tube and describes its depth. Consequently, these three indices describe the three different modes. In terms of notation, the entries of the different order tensors are denoted by $(\boldsymbol{\mathcal{A}})_{l_1 l_2 l_3} = a_{l_1 l_2 l_3}$, $(\mathbf{A})_{l_1 l_2} = a_{l_1 l_2}$, and $(\mathbf{a})_{l_1} = a_{l_1}$.

In the case of a matrix, the vectorization operation leads to a vector that contains the concatenated columns of the original matrix. In the case of a third-order tensor, the so-called matricization operation concatenates the "slices" of the tensor and produces a large matrix. Of course, the result depends on which index, all the elements of which are considered first. Thus, the matricization can be performed along three different modes [8,9]. For example, considering the mode row and then varying the columns and the tubes, we obtain:

$$\begin{aligned}\mathbf{A}_{[1]} &= \mathbf{A}_{:,1:L_2,1:L_3} \\ &= \begin{bmatrix} \mathbf{A}_{::1} & \cdots & \mathbf{A}_{::L_3} \end{bmatrix},\end{aligned}$$

where $\mathbf{A}_{[1]} \in \mathbb{R}^{L_1 \times L_2 L_3}$ and the matrices $\mathbf{A}_{::l_3} \in \mathbb{R}^{L_1 \times L_2}$ with $l_3 = 1, 2 \ldots, L_3$ denote the frontal slices. Similarly, we can take the mode column and then vary the lines and the tubes, which results in $\mathbf{A}_{[2]} = \mathbf{A}_{1:L_1,:,1:L_3}$, with $\mathbf{A}_{[2]} \in \mathbb{R}^{L_2 \times L_1 L_3}$. Finally, we can take the mode tube and then vary the rows and the columns, in order to obtain $\mathbf{A}_{[3]} = \mathbf{A}_{1:L_1,1:L_2,:}$, where $\mathbf{A}_{[3]} \in \mathbb{R}^{L_3 \times L_1 L_2}$. The ranks of $\mathbf{A}_{[1]}$, $\mathbf{A}_{[2]}$, and $\mathbf{A}_{[3]}$ represent the mode-1, mode-2, and mode-3 ranks of the tensor, respectively. Furthermore, the vectorization of a tensor (e.g., following mode-1) is

$$\begin{aligned}\mathrm{vec}\left(\boldsymbol{\mathcal{A}}\right) &= \mathrm{vec}\left(\mathbf{A}_{[1]}\right) \\ &= \begin{bmatrix} \mathrm{vec}\left(\mathbf{A}_{::1}\right) \\ \vdots \\ \mathrm{vec}\left(\mathbf{A}_{::L_3}\right) \end{bmatrix} \in \mathbb{R}^{L_1 L_2 L_3}.\end{aligned}$$

Nevertheless, there are some fundamental differences between the rank of a matrix $\mathbf{A} \in \mathbb{R}^{L_1 \times L_2}$ and the rank of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{L_1 \times L_2 \times L_3}$. For example, the rank of $\mathbf{A}$ can never be larger than $\min\{L_1, L_2\}$, while the rank of $\boldsymbol{\mathcal{A}}$ can be greater than $\min\{L_1, L_2, L_3\}$. A rank-1 tensor (of dimension $L_1 \times L_2 \times L_3$) is defined as:

$$\boldsymbol{\mathcal{B}} = \mathbf{b}_1 \circ \mathbf{b}_2 \circ \mathbf{b}_3, \tag{1}$$

where $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ are vectors of lengths L_1, L_2, and L_3, respectively, $\circ$ is the vector outer product, and the elements of $\boldsymbol{\mathcal{B}}$ are given by $(\boldsymbol{\mathcal{B}})_{l_1 l_2 l_3} = b_{1,l_1} b_{2,l_2} b_{3,l_3}$, where b_{i,l_i} are the elements of the vector $\mathbf{b}_i$, with $i = 1, 2, 3$ and $l_i = 1, 2, \ldots, L_i$. In this case, it can be easily verified that:

$$\mathrm{vec}\left(\boldsymbol{\mathcal{B}}\right) = \mathbf{b}_3 \otimes \mathbf{b}_2 \otimes \mathbf{b}_1, \tag{2}$$

where $\otimes$ denotes the Kronecker product [24]. In this context, the rank of a tensor $\boldsymbol{\mathcal{A}}$, denoted $\mathrm{rank}\left(\boldsymbol{\mathcal{A}}\right)$, is defined as the minimum number of rank-1 tensors that generate $\boldsymbol{\mathcal{A}}$ as their sum. For example, if:

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \mathbf{a}_{3r}, \tag{3}$$

where $\mathbf{a}_{1r}$, $\mathbf{a}_{2r}$, and $\mathbf{a}_{3r}$ are vectors of lengths L_1, L_2, and L_3, respectively, then $\mathrm{rank}\left(\boldsymbol{\mathcal{A}}\right) = R$ when R is minimal and (3) is called the canonical polyadic decomposition (CPD) of $\boldsymbol{\mathcal{A}}$.

Another important operation is the multiplication of a tensor with a matrix [1,2], which can also be defined in different ways. For example, the mode-1 product between the tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{L_1 \times L_2 \times L_3}$ and the matrix $\mathbf{M}_1 \in \mathbb{R}^{M_1 \times L_1}$ gives the tensor:

$$\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{A}} \times_1 \mathbf{M}_1,\ \boldsymbol{\mathcal{U}} \in \mathbb{R}^{M_1 \times L_2 \times L_3}, \tag{4}$$

whose entries are $u_{m_1 l_2 l_3} = \sum_{l_1=1}^{L_1} a_{l_1 l_2 l_3} m_{m_1 l_1}$ (for $m_1 = 1, 2, \ldots, M_1$) and $\mathbf{U}_{[1]} = \mathbf{M}_1 \mathbf{A}_{[1]}$. Similarly, the mode-2 product between the same tensor $\boldsymbol{\mathcal{A}}$ and the matrix $\mathbf{M}_2 \in \mathbb{R}^{M_2 \times L_2}$ leads to the tensor:

$$\underline{\boldsymbol{\mathcal{U}}} = \boldsymbol{\mathcal{A}} \times_2 \mathbf{M}_2,\ \underline{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{L_1 \times M_2 \times L_3}, \tag{5}$$

with the entries $\underline{u}_{l_1 m_2 l_3} = \sum_{l_2=1}^{L_2} a_{l_1 l_2 l_3} m_{m_2 l_2}$ (for $m_2 = 1, 2, \ldots, M_2$) and $\underline{\mathbf{U}}_{[2]} = \mathbf{M}_2 \mathbf{A}_{[2]}$. Finally, the mode-3 product between the tensor $\boldsymbol{\mathcal{A}}$ and the matrix $\mathbf{M}_3 \in \mathbb{R}^{M_3 \times L_3}$ results in the tensor:

$$\overline{\boldsymbol{\mathcal{U}}} = \boldsymbol{\mathcal{A}} \times_3 \mathbf{M}_3,\ \overline{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{L_1 \times L_2 \times M_3}, \tag{6}$$

having the entries $\overline{u}_{l_1 l_2 m_3} = \sum_{l_3=1}^{L_3} a_{l_1 l_2 l_3} m_{m_3 l_3}$ (for $m_3 = 1, 2, \ldots, M_3$), while $\overline{\mathbf{U}}_{[3]} = \mathbf{M}_3 \mathbf{A}_{[3]}$. Clearly, we can multiply the tensor $\boldsymbol{\mathcal{A}}$ with the three previously defined matrices $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$. In this case, we get the tensor:

$$\begin{aligned}\boldsymbol{\mathcal{T}} &= \boldsymbol{\mathcal{A}} \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \times_3 \mathbf{M}_3 \\ &= \boldsymbol{\mathcal{A}} \times_1 \mathbf{M}_1 \times_3 \mathbf{M}_3 \times_2 \mathbf{M}_2 \\ &= \boldsymbol{\mathcal{A}} \times_2 \mathbf{M}_2 \times_1 \mathbf{M}_1 \times_3 \mathbf{M}_3 \\ &= \boldsymbol{\mathcal{A}} \times_2 \mathbf{M}_2 \times_3 \mathbf{M}_3 \times_1 \mathbf{M}_1 \\ &= \boldsymbol{\mathcal{A}} \times_3 \mathbf{M}_3 \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \\ &= \boldsymbol{\mathcal{A}} \times_3 \mathbf{M}_3 \times_2 \mathbf{M}_2 \times_1 \mathbf{M}_1,\end{aligned} \tag{7}$$

where $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{M_1 \times M_2 \times M_3}$. As a consequence, considering $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ three vectors of lengths L_1, L_2, and L_3, respectively, the multiplication of the tensor $\boldsymbol{\mathcal{A}}$ with the transposed vectors gives the scalar:

$$\begin{aligned}c &= \boldsymbol{\mathcal{A}} \times_1 \mathbf{b}_1^T \times_2 \mathbf{b}_2^T \times_3 \mathbf{b}_3^T \\ &= \sum_{l_1=1}^{L_1} \sum_{l_2=1}^{L_2} \sum_{l_3=1}^{L_3} a_{l_1 l_2 l_3} b_{1 l_1} b_{2 l_2} b_{3 l_3},\end{aligned} \tag{8}$$

where ^{T} is the transpose operator. It is easy to check that (8) is trilinear with respect to $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$.

At this point, we can also define the inner product between two tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ of the same dimension ($L_1 \times L_2 \times L_3$), which is:

$$\begin{aligned}\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle &= \sum_{l_1=1}^{L_1} \sum_{l_2=1}^{L_2} \sum_{l_3=1}^{L_3} a_{l_1 l_2 l_3} b_{l_1 l_2 l_3} \\ &= \mathrm{vec}^T(\boldsymbol{\mathcal{B}})\, \mathrm{vec}(\boldsymbol{\mathcal{A}}).\end{aligned} \tag{9}$$

Therefore, Expression (8) can also be written in a more convenient way, i.e.,

$$\begin{aligned}c &= \langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle \\ &= \mathrm{vec}^T(\boldsymbol{\mathcal{B}})\, \mathrm{vec}(\boldsymbol{\mathcal{A}}) \\ &= \mathrm{vec}^T(\mathbf{b}_1 \circ \mathbf{b}_2 \circ \mathbf{b}_3)\, \mathrm{vec}(\boldsymbol{\mathcal{A}}) \\ &= (\mathbf{b}_3 \otimes \mathbf{b}_2 \otimes \mathbf{b}_1)^T\, \mathrm{vec}(\boldsymbol{\mathcal{A}}),\end{aligned} \tag{10}$$

where $\boldsymbol{\mathcal{B}} = \mathbf{b}_1 \circ \mathbf{b}_2 \circ \mathbf{b}_3$ (see (1)). Moreover, if $\boldsymbol{\mathcal{A}}$ is also a rank-1 tensor, i.e., $\boldsymbol{\mathcal{A}} = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \mathbf{a}_3$, where the vectors $\mathbf{a}_i$ have the lengths L_i (with $i = 1, 2, 3$), then:

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \mathbf{b}_1^T \mathbf{a}_1 \times \mathbf{b}_2^T \mathbf{a}_2 \times \mathbf{b}_3^T \mathbf{a}_3. \tag{11}$$

Furthermore, it is easy to check that:

$$\begin{aligned}\boldsymbol{\mathcal{B}} \times_1 \mathbf{M}_1 &= \mathbf{M}_1\mathbf{b}_1 \circ \mathbf{b}_2 \circ \mathbf{b}_3, \\ \boldsymbol{\mathcal{B}} \times_2 \mathbf{M}_2 &= \mathbf{b}_1 \circ \mathbf{M}_2\mathbf{b}_2 \circ \mathbf{b}_3, \\ \boldsymbol{\mathcal{B}} \times_3 \mathbf{M}_3 &= \mathbf{b}_1 \circ \mathbf{b}_2 \circ \mathbf{M}_3\mathbf{b}_3,\end{aligned}$$

where the matrices $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ were previously defined (related to (4)–(6)).

The short background on tensors provided before and the main related operations (e.g., matricization, vectorization, rank, and different types of product) aim to facilitate the development that follows in Section 3. It is also important to outline that the trilinear forms result in the context of the decomposition of third-order tensors. Extension to higher order tensors and multilinear forms could be straightforward when dealing with rank-1 tensors. Otherwise, in the general case, decomposing higher rank higher order tensors (see (3)) raises additional difficulties, as will be briefly pointed out at the end of Section 5.

3. RLS Algorithm for Trilinear Forms

In the following, for the sake of consistency with the development of the NLMS-TF algorithm, we will keep the framework and notation from [17]. Therefore, let us consider the output of a multiple-input single-output (MISO) system (with real-valued data) at the discrete-time index n defined as:

$$\begin{aligned} y(n) &= \boldsymbol{\mathcal{X}}(n) \times_1 \mathbf{h}_1^T \times_2 \mathbf{h}_2^T \times_3 \mathbf{h}_3^T \\ &= \sum_{l_1=1}^{L_1} \sum_{l_2=1}^{L_2} \sum_{l_3=1}^{L_3} x_{l_1 l_2 l_3}(n) h_{1,l_1} h_{2,l_2} h_{3,l_3}, \end{aligned} \tag{12}$$

where the tensorial form $\boldsymbol{\mathcal{X}}(n) \in \mathbb{R}^{L_1 \times L_2 \times L_3}$ groups the input signals, with:

$$[\boldsymbol{\mathcal{X}}(n)]_{l_1 l_2 l_3} = x_{l_1 l_2 l_3}(n),\ l_i = 1, 2, \ldots, L_i,\ i = 1, 2, 3,$$

and the vectors $\mathbf{h}_i,\ i = 1, 2, 3$ (of lengths L_1, L_2, and L_3, respectively) define the three impulse responses, i.e.,

$$\mathbf{h}_i = \begin{bmatrix} h_{i,1} & h_{i,2} & \cdots & h_{i,L_i} \end{bmatrix}^T,\ i = 1, 2, 3.$$

As we can notice, $y(n)$ represents a trilinear form (see (12) as compared to (8)), because it is a linear function of each of the vectors $\mathbf{h}_i,\ i = 1, 2, 3$, if the other two are fixed.

Next, we can also introduce a rank-1 tensor of dimension $L_1 \times L_2 \times L_3$, using the three impulse responses of the MISO system:

$$\boldsymbol{\mathcal{H}} = \mathbf{h}_1 \circ \mathbf{h}_2 \circ \mathbf{h}_3, \tag{13}$$

whose elements are:

$$(\boldsymbol{\mathcal{H}})_{l_1 l_2 l_3} = h_{1,l_1} h_{2,l_2} h_{3,l_3},\ l_i = 1, 2, \ldots, L_i,\ i = 1, 2, 3.$$

Consequently, the output signal results in:

$$y(n) = \mathrm{vec}^T(\boldsymbol{\mathcal{H}})\, \mathrm{vec}\left[\boldsymbol{\mathcal{X}}(n)\right], \tag{14}$$

where:

$$\text{vec}\left(\boldsymbol{\mathcal{H}}\right) = \begin{bmatrix} \text{vec}\left(\mathbf{H}_{::1}\right) \\ \vdots \\ \text{vec}\left(\mathbf{H}_{::L_3}\right) \end{bmatrix}, \tag{15}$$

$$\text{vec}\left[\boldsymbol{\mathcal{X}}(n)\right] = \begin{bmatrix} \text{vec}\left[\mathbf{X}_{::1}(n)\right] \\ \vdots \\ \text{vec}\left[\mathbf{X}_{::L_3}(n)\right] \end{bmatrix}, \tag{16}$$

with $\mathbf{H}_{::l_3}$ and $\mathbf{X}_{::l_3}(n)$ $(l_3 = 1, 2 \ldots, L_3)$ being the frontal slices of $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{X}}(n)$, respectively. At this point, we can introduce the notation:

$$\mathbf{h} \triangleq \text{vec}\left(\boldsymbol{\mathcal{H}}\right) = \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1, \tag{17}$$

$$\mathbf{x}(n) \triangleq \text{vec}\left[\boldsymbol{\mathcal{X}}(n)\right], \tag{18}$$

where $\mathbf{h}$ and $\mathbf{x}(n)$ are two long vectors, each of them having $L_1L_2L_3$ elements. Thus, the output signal can also be expressed as:

$$y(n) = \mathbf{h}^T\mathbf{x}(n). \tag{19}$$

The main goal is to estimate the output of the MISO system, which is usually corrupted by an additive noise. Hence, the reference signal results in:

$$\begin{aligned} d(n) &= y(n) + w(n) \\ &= \mathbf{h}^T\mathbf{x}(n) + w(n), \end{aligned} \tag{20}$$

where $w(n)$ is a zero-mean additive noise, which is uncorrelated with the input signals. Alternatively, we could estimate the global impulse response $\mathbf{h}$, using an adaptive filter $\widehat{\mathbf{h}}(n)$ (of length $L_1L_2L_3$). At this point, we may also define the error signal:

$$\begin{aligned} e(n) &= d(n) - \widehat{y}(n) \\ &= d(n) - \widehat{\mathbf{h}}^T(n-1)\mathbf{x}(n), \end{aligned} \tag{21}$$

which represents the difference between the reference signal and the estimated signal, $\widehat{y}(n)$.

Based on (17), we can notice that the global impulse response $\mathbf{h}$ (of length $L_1L_2L_3$) results based on a combination of the shorter impulse responses $\mathbf{h}_i,\ i = 1, 2, 3$, of lengths L_1, L_2, and L_3, respectively. Consequently, the estimated impulse response can also be decomposed as:

$$\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}_3(n) \otimes \widehat{\mathbf{h}}_2(n) \otimes \widehat{\mathbf{h}}_1(n), \tag{22}$$

where $\widehat{\mathbf{h}}_i(n),\ i = 1, 2, 3$ are three adaptive filters (with L_1, L_2, and L_3 coefficients, respectively), which aim to model the individual impulse responses $\mathbf{h}_i,\ i = 1, 2, 3$. Nevertheless, we should notice that there is no unique solution related to the decomposition in (22). It is obvious that, for any constants $\eta_1,\ \eta_2$, and η_3, with $\eta_1\eta_2\eta_3 = 1$, we have:

$$\begin{aligned} \mathbf{h} &= \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1 \\ &= \eta_3\mathbf{h}_3 \otimes \eta_2\mathbf{h}_2 \otimes \eta_1\mathbf{h}_1. \end{aligned} \tag{23}$$

Hence, $\eta_i\mathbf{h}_i,\ i = 1, 2, 3$ also represent a set of solutions. However, the global impulse response $\mathbf{h}$ is identified with no scaling ambiguity.

Following the decomposition from (22), we can easily verify that:

$$\widehat{\mathbf{h}}(n) = \widehat{\mathbf{H}}_{32}(n)\widehat{\mathbf{h}}_1(n) \tag{24}$$
$$= \widehat{\mathbf{H}}_{31}(n)\widehat{\mathbf{h}}_2(n) \tag{25}$$
$$= \widehat{\mathbf{H}}_{21}(n)\widehat{\mathbf{h}}_3(n), \tag{26}$$

where:

$$\widehat{\mathbf{H}}_{32}(n) = \widehat{\mathbf{h}}_3(n) \otimes \widehat{\mathbf{h}}_2(n) \otimes \mathbf{I}_{L_1}, \tag{27}$$
$$\widehat{\mathbf{H}}_{31}(n) = \widehat{\mathbf{h}}_3(n) \otimes \mathbf{I}_{L_2} \otimes \widehat{\mathbf{h}}_1(n), \tag{28}$$
$$\widehat{\mathbf{H}}_{21}(n) = \mathbf{I}_{L_3} \otimes \widehat{\mathbf{h}}_2(n) \otimes \widehat{\mathbf{h}}_1(n), \tag{29}$$

and $\mathbf{I}_{L_i}$, $i = 1, 2, 3$ are the identity matrices of sizes $L_1 \times L_1$, $L_2 \times L_2$, and $L_3 \times L_3$, respectively. Moreover, introducing the notation:

$$\mathbf{x}_{32}(n) = \widehat{\mathbf{H}}_{32}^T(n-1)\mathbf{x}(n), \tag{30}$$
$$\mathbf{x}_{31}(n) = \widehat{\mathbf{H}}_{31}^T(n-1)\mathbf{x}(n), \tag{31}$$
$$\mathbf{x}_{21}(n) = \widehat{\mathbf{H}}_{21}^T(n-1)\mathbf{x}(n), \tag{32}$$

the error signal from (21) can be expressed in three equivalent ways as:

$$e(n) = d(n) - \widehat{\mathbf{h}}_1^T(n-1)\mathbf{x}_{32}(n), \tag{33}$$
$$= d(n) - \widehat{\mathbf{h}}_2^T(n-1)\mathbf{x}_{31}(n), \tag{34}$$
$$= d(n) - \widehat{\mathbf{h}}_3^T(n-1)\mathbf{x}_{21}(n). \tag{35}$$

Based on the least-squares error criterion [18] applied in the context of (20) and (21), the conventional RLS algorithm is derived from:

$$J\left[\widehat{\mathbf{h}}(n)\right] = \sum_{i=1}^{n} \lambda^{n-i} \left[d(i) - \widehat{\mathbf{h}}^T(n)\mathbf{x}(i)\right]^2, \tag{36}$$

where $\lambda \leq 1$ is a positive constant known as the forgetting factor. On the other hand, based on (24)–(26), the cost function from (36) can be expressed in three different ways, targeting the optimization of the individual components, i.e.,

$$J_{\widehat{\mathbf{h}}_3,\widehat{\mathbf{h}}_2}\left[\widehat{\mathbf{h}}_1(n)\right] = \sum_{i=1}^{n} \lambda_1^{n-i} \left[d(i) - \widehat{\mathbf{h}}_1^T(n)\mathbf{x}_{32}(i)\right]^2, \tag{37}$$
$$J_{\widehat{\mathbf{h}}_3,\widehat{\mathbf{h}}_1}\left[\widehat{\mathbf{h}}_2(n)\right] = \sum_{i=1}^{n} \lambda_2^{n-i} \left[d(i) - \widehat{\mathbf{h}}_2^T(n)\mathbf{x}_{31}(i)\right]^2, \tag{38}$$
$$J_{\widehat{\mathbf{h}}_2,\widehat{\mathbf{h}}_1}\left[\widehat{\mathbf{h}}_3(n)\right] = \sum_{i=1}^{n} \lambda_3^{n-i} \left[d(i) - \widehat{\mathbf{h}}_3^T(n)\mathbf{x}_{21}(i)\right]^2, \tag{39}$$

where λ_1, λ_2, and λ_3 are the individual forgetting factors. The previous cost functions suggest a "trilinear" optimization strategy [25], where we assume that two components are fixed during the optimization of the third one. Consequently, based on the minimization of (37)–(39) with respect to $\widehat{\mathbf{h}}_1(n)$, $\widehat{\mathbf{h}}_2(n)$, and $\widehat{\mathbf{h}}_3(n)$, respectively, the following set of normal equations are obtained:

$$\mathbf{R}_{32}(n)\widehat{\mathbf{h}}_1(n) = \mathbf{p}_{32}(n), \tag{40}$$
$$\mathbf{R}_{31}(n)\widehat{\mathbf{h}}_2(n) = \mathbf{p}_{31}(n), \tag{41}$$
$$\mathbf{R}_{21}(n)\widehat{\mathbf{h}}_3(n) = \mathbf{p}_{21}(n), \tag{42}$$

where:

$$\begin{aligned}
\mathbf{R}_{32}(n) &= \sum_{i=1}^{n} \lambda_1^{n-i}\mathbf{x}_{32}(i)\mathbf{x}_{32}^T(i) \\
&= \lambda_1\mathbf{R}_{32}(n-1) + \mathbf{x}_{32}(n)\mathbf{x}_{32}^T(n), \\
\mathbf{p}_{32}(n) &= \sum_{i=1}^{n} \lambda_1^{n-i}\mathbf{x}_{32}(i)d(i) \\
&= \lambda_1\mathbf{p}_{32}(n-1) + \mathbf{x}_{32}(n)d(n), \\
\mathbf{R}_{31}(n) &= \sum_{i=1}^{n} \lambda_2^{n-i}\mathbf{x}_{31}(i)\mathbf{x}_{31}^T(i) \\
&= \lambda_2\mathbf{R}_{31}(n-1) + \mathbf{x}_{31}(n)\mathbf{x}_{31}^T(n), \\
\mathbf{p}_{31}(n) &= \sum_{i=1}^{n} \lambda_2^{n-i}\mathbf{x}_{31}(i)d(i) \\
&= \lambda_2\mathbf{p}_{31}(n-1) + \mathbf{x}_{31}(n)d(n), \\
\mathbf{R}_{21}(n) &= \sum_{i=1}^{n} \lambda_3^{n-i}\mathbf{x}_{21}(i)\mathbf{x}_{21}^T(i) \\
&= \lambda_2\mathbf{R}_{21}(n-1) + \mathbf{x}_{21}(n)\mathbf{x}_{21}^T(n), \\
\mathbf{p}_{21}(n) &= \sum_{i=1}^{n} \lambda_3^{n-i}\mathbf{x}_{21}(i)d(i) \\
&= \lambda_3\mathbf{p}_{21}(n-1) + \mathbf{x}_{21}(n)d(n).
\end{aligned}$$

Solving (40)–(42), the updates of the individual filters result in:

$$\begin{aligned}
\widehat{\mathbf{h}}_1(n) &= \widehat{\mathbf{h}}_1(n-1) + \mathbf{R}_{32}^{-1}(n)\mathbf{x}_{32}(n)e(n) \\
&= \widehat{\mathbf{h}}_1(n-1) + \mathbf{k}_{32}(n)e(n), \qquad (43)\\
\widehat{\mathbf{h}}_2(n) &= \widehat{\mathbf{h}}_2(n-1) + \mathbf{R}_{31}^{-1}(n)\mathbf{x}_{31}(n)e(n) \\
&= \widehat{\mathbf{h}}_2(n-1) + \mathbf{k}_{31}(n)e(n), \qquad (44)\\
\widehat{\mathbf{h}}_3(n) &= \widehat{\mathbf{h}}_3(n-1) + \mathbf{R}_{21}^{-1}(n)\mathbf{x}_{21}(n)e(n) \\
&= \widehat{\mathbf{h}}_3(n-1) + \mathbf{k}_{21}(n)e(n), \qquad (45)
\end{aligned}$$

where $\mathbf{k}_{32}(n) = \mathbf{R}_{32}^{-1}(n)\mathbf{x}_{32}(n)$, $\mathbf{k}_{31}(n) = \mathbf{R}_{31}^{-1}(n)\mathbf{x}_{31}(n)$, and $\mathbf{k}_{21}(n) = \mathbf{R}_{21}^{-1}(n)\mathbf{x}_{21}(n)$ are the Kalman gain vectors, while the error signal can be computed based on (33). At this point, the main task is to update the inverse of the matrices $\mathbf{R}_{32}(n)$, $\mathbf{R}_{31}(n)$, and $\mathbf{R}_{21}(n)$ efficiently. The solution relies on the matrix inversion lemma [18], which leads to the following updates:

$$\mathbf{R}_{32}^{-1}(n) = \frac{1}{\lambda_1}\left[\mathbf{I}_{L_1} - \mathbf{k}_{32}(n)\mathbf{x}_{32}^T(n)\right]\mathbf{R}_{32}^{-1}(n-1), \tag{46}$$

$$\mathbf{R}_{31}^{-1}(n) = \frac{1}{\lambda_2}\left[\mathbf{I}_{L_2} - \mathbf{k}_{31}(n)\mathbf{x}_{31}^T(n)\right]\mathbf{R}_{31}^{-1}(n-1), \tag{47}$$

$$\mathbf{R}_{21}^{-1}(n) = \frac{1}{\lambda_3}\left[\mathbf{I}_{L_3} - \mathbf{k}_{21}(n)\mathbf{x}_{21}^T(n)\right]\mathbf{R}_{21}^{-1}(n-1). \tag{48}$$

Therefore, the Kalman gain vectors are evaluated as:

$$\mathbf{k}_{32}(n) = \frac{\mathbf{R}_{32}^{-1}(n-1)\mathbf{x}_{32}(n)}{\lambda_1 + \mathbf{x}_{32}^T(n)\mathbf{R}_{32}^{-1}(n-1)\mathbf{x}_{32}(n)}, \tag{49}$$

$$\mathbf{k}_{31}(n) = \frac{\mathbf{R}_{31}^{-1}(n-1)\mathbf{x}_{31}(n)}{\lambda_2 + \mathbf{x}_{31}^T(n)\mathbf{R}_{31}^{-1}(n-1)\mathbf{x}_{31}(n)}, \tag{50}$$

$$\mathbf{k}_{21}(n) = \frac{\mathbf{R}_{21}^{-1}(n-1)\mathbf{x}_{21}(n)}{\lambda_3 + \mathbf{x}_{21}^T(n)\mathbf{R}_{21}^{-1}(n-1)\mathbf{x}_{21}(n)}. \tag{51}$$

For initialization, we can choose:

$$\widehat{\mathbf{h}}_1(0) = \begin{bmatrix} 1 \\ \mathbf{0}_{L_1-1} \end{bmatrix}, \tag{52}$$

$$\widehat{\mathbf{h}}_2(0) = \frac{1}{L_2}\mathbf{1}_{L_2}, \tag{53}$$

$$\widehat{\mathbf{h}}_3(0) = \frac{1}{L_3}\mathbf{1}_{L_3}, \tag{54}$$

where $\mathbf{0}_N$ and $\mathbf{1}_N$ are two vectors of length N, all elements of which are equal to zero and one, respectively.

The proposed RLS algorithm for trilinear forms, namely RLS-TF, is summarized in Table 1. It could also be interpreted as an extension of the RLS-based algorithm tailored for bilinear forms, which was presented in [12]. However, if the MISO system identification problem results based on (12), it is natural to use the RLS-TF algorithm, which is designed for the identification of third-order (rank-1) tensors.

In terms of computational complexity, it can be noticed that the proposed RLS-TF algorithm combines the solutions provided by three RLS-based filters, i.e., $\widehat{\mathbf{h}}_1(n)$ (of length L_1), $\widehat{\mathbf{h}}_2(n)$ (of length L_2), and $\widehat{\mathbf{h}}_3(n)$ (of length L_3). Since the complexity of a regular RLS-based algorithm is proportional to the square of the filter length, the overall complexity of the RLS-TF algorithm roughly results in $\mathcal{O}(L_1^2 + L_2^2 + L_3^2)$. On the other hand, the system identification problem can also be handled based on the conventional RLS algorithm, which results following (20) and (21), together with the cost function from (36). However, in this case, there is a single adaptive filter $\widehat{\mathbf{h}}(n)$, of length $L = L_1L_2L_3$, so that the computational complexity is of the order of $\mathcal{O}(L^2)$. This could be much more computationally expensive as compared to the proposed RLS-TF algorithm.

Table 1. RLS algorithm for trilinear forms (RLS-TF).

```
Initialization:
Set $\widehat{\mathbf{h}}_1(0)$, $\widehat{\mathbf{h}}_2(0)$, and $\widehat{\mathbf{h}}_3(0)$ based on (52)–(54)
$\mathbf{R}_{32}^{-1}(0) = \frac{1}{\delta_1}\mathbf{I}_{L_1}$, $\mathbf{R}_{31}^{-1}(0) = \frac{1}{\delta_2}\mathbf{I}_{L_2}$, $\mathbf{R}_{21}^{-1}(0) = \frac{1}{\delta_3}\mathbf{I}_{L_3}$
(Regularization parameters: $\delta_1 > 0$, $\delta_2 > 0$, $\delta_3 > 0$)
$\lambda_1 = 1 - \frac{1}{KL_1}$, $\lambda_2 = 1 - \frac{1}{KL_2}$, $\lambda_3 = 1 - \frac{1}{KL_3}$
(Tuning constant : $K \geq 1$)
For $n = 1, 2, \ldots,$ number of iterations :
Compute $\widehat{\mathbf{H}}_{32}(n-1)$, $\widehat{\mathbf{H}}_{31}(n-1)$, and $\widehat{\mathbf{H}}_{21}(n-1)$ based on (27)–(29)
Compute $\mathbf{x}_{32}(n)$, $\mathbf{x}_{31}(n)$, and $\mathbf{x}_{21}(n)$ based on (30)–(32)
Evaluate the error signal $e(n)$ based on (33)
Compute $\mathbf{k}_{32}(n)$, $\mathbf{k}_{31}(n)$, and $\mathbf{k}_{21}(n)$ based on (49)–(51)
Update $\widehat{\mathbf{h}}_1(n)$, $\widehat{\mathbf{h}}_2(n)$, and $\widehat{\mathbf{h}}_3(n)$ based on (43)–(45)
Update $\mathbf{R}_{32}^{-1}(n)$, $\mathbf{R}_{31}^{-1}(n)$, and $\mathbf{R}_{21}^{-1}(n)$ based on (46)–(48)
Evaluate $\widehat{\mathbf{h}}(n)$ based on (22)
```

Basically, the RLS-TF algorithm "transforms" a large system identification problem of length $L = L_1L_2L_3$ into three "smaller" problems of lengths L_1, L_2, and L_3, respectively, with advantages in terms of both performance (as will be shown in simulations) and complexity. As outlined before, the proposed RLS-TF algorithm combines the solutions provided by three adaptive filters of lengths L_1, L_2, and L_3, respectively, while the conventional RLS algorithm deals with a single filter of length $L = L_1L_2L_3$, which is usually much longer. Since the length of the filter highly influences the main performance criteria, i.e., convergence rate and misadjustment [18], the proposed algorithm is able to outperform the conventional one in terms of both criteria. In other words, the shorter the length, the faster the convergence and the lower the misadjustment. This expected behaviour will be supported by the simulation results provided in the next section.

Finally, we should observe that there are some extra operations specific to the RLS-TF algorithm. For example, the "input" signals $\mathbf{x}_{32}(n)$, $\mathbf{x}_{31}(n)$, and $\mathbf{x}_{21}(n)$ result based on (30)–(32), which rely on (27)–(29). Furthermore, the global impulse response (if required within the application) can be evaluated based on (22). These operations require Kronecker products, but the related computational complexity is moderate, i.e., of the order of $\mathcal{O}(L_1L_2L_3) = \mathcal{O}(L)$.

The detailed computational complexities of the proposed RLS-TF algorithm and other benchmark algorithms (i.e., the conventional RLS and NLMS algorithms) are summarized in Table 2. For a better visualization, the computational complexities are also illustrated in Figure 1, in terms of the number of multiplications and additions (per iteration), for different values of L_1; the other lengths are fixed to $L_2 = 8$ and $L_3 = 4$ (similar to the experimental setup from Section 4). As we can notice, the computational complexity of the conventional RLS algorithm was significantly greater, while the computational amount of the proposed RLS-TF algorithm was closer to the conventional NLMS algorithm, especially for higher lengths.

Table 2. Computational complexity of the RLS-TF algorithm, as compared to the conventional RLS and NLMS algorithms.

Algorithms	$\times$	$+$	$\div$
RLS	$2L^2 + 2L$	$2L^2 + L + 1$	1
RLS-TF	$4L + 3\left(L_1^2 + L_2^2 + L_3^2\right) + 3\left(L_1 + L_2 + L_3\right) + \min(L_1, L_2, L_3)$	$3L + 2\left(L_1^2 + L_2^2 + L_3^2\right) + L_1 + L_2 + L_3 + \min(L_1, L_2, L_3)$	3
NLMS	$2L + 2$	$2L + 3$	1

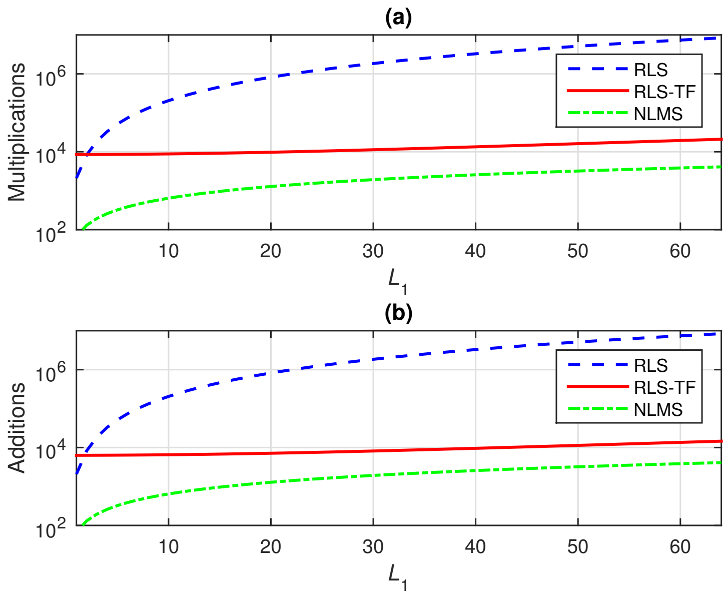

Figure 1. Computational complexity of the proposed RLS-TF algorithm, as compared to the conventional RLS and NLMS algorithms, as a function of L_1; the other dimensions are set to $L_2 = 8$ and $L_3 = 4$: (**a**) number of multiplications per iteration and (**b**) number of additions per iteration.

4. Simulation Results

Simulations were performed in the framework of a tensor-based system identification problem, which resulted following the MISO model defined by (12) and (20) and was similar to the setup used in [17]. The input signals that form the third-order tensor $\boldsymbol{\mathcal{X}}(n)$ are AR(1) processes; they are generated by filtering white Gaussian noises through a first-order system $1/\left(1 - 0.9z^{-1}\right)$. The additive noise $w(n)$ is white and Gaussian; its variance was set to $\sigma_w^2 = 0.01$.

The third-order system used in the simulations and its components ($\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$) are depicted in Figure 2. First, the component $\mathbf{h}_1$ is an impulse response from the G168 Recommendation [26], of length $L_1 = 64$; it is provided in Figure 2a. Second, in Figure 2b, the component $\mathbf{h}_2$ is a random impulse response (with Gaussian distribution) of length $L_2 = 8$. Third, the coefficients of the last component, i.e., the impulse response $\mathbf{h}_3$, are depicted in Figure 2c; those were evaluated as $h_{3,l_3} = 0.5^{l_3 - 1}$, $l_3 = 1, 2, \ldots, L_3$, using the length $L_3 = 4$. Therefore, the global impulse response from Figure 2d resulted as $\mathbf{h} = \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1$, and its length was $L = L_1 L_2 L_3 = 2048$. This global impulse response resembled a channel with echoes, e.g., like an acoustic echo path [27]. Finally, the third-order (rank-1) tensor $\boldsymbol{\mathcal{H}}$ of dimension $L_1 \times L_2 \times L_3$ could be formed according to (13). In order to test the tracking capabilities of the algorithms, an abrupt change of the system was introduced in the middle of each experiment, by changing the sign of the coefficients of each impulse response.

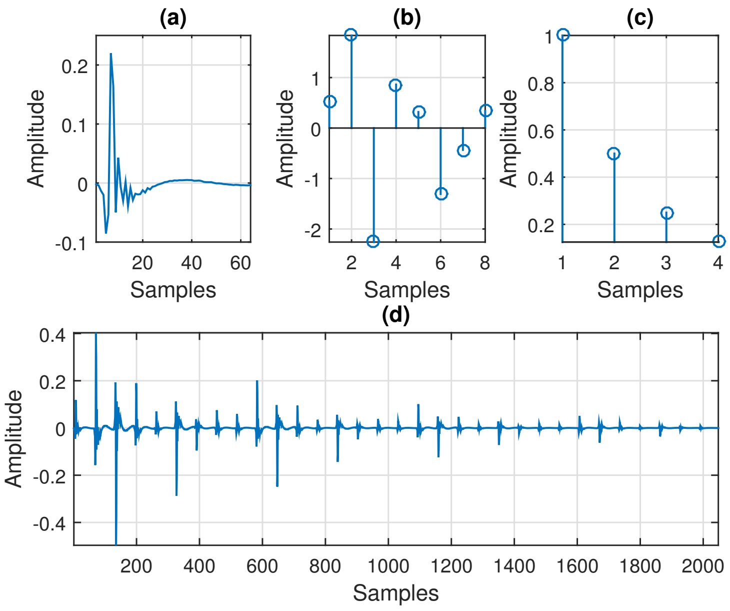

Figure 2. The components of the third-order system used in simulations: (**a**) $\mathbf{h}_1$ is the first impulse response (of length $L_1 = 64$) from the G168 Recommendation [26]; (**b**) $\mathbf{h}_2$ is a randomly generated impulse response (of length $L_2 = 8$), with Gaussian distribution; (**c**) the impulse response $\mathbf{h}_3$ (of length $L_3 = 4$), with the coefficients computed as $h_{3,l_3} = 0.5^{l_3-1}$, with $l_3 = 1, 2, \ldots, L_3$; and (**d**) the global impulse response (of length $L = L_1L_2L_3 = 2048$) results based on (17), $\mathbf{h} = \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1$.

As shown in Section 3, the proposed RLS-TF algorithm was designed to estimate the individual components of the global system. However, we could identify $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$ up to some scaling factors, as explained using (23). Therefore, to evaluate the identification of these individual impulse responses, a proper performance measure is the normalized projection misalignment (NPM) [28]:

$$\text{NPM}\left[\mathbf{h}_1, \widehat{\mathbf{h}}_1(n)\right] = 1 - \left[\frac{\mathbf{h}_1^T\widehat{\mathbf{h}}_1(n)}{\|\mathbf{h}_1\|_2 \left\|\widehat{\mathbf{h}}_1(n)\right\|_2}\right]^2, \tag{55}$$

$$\text{NPM}\left[\mathbf{h}_2, \widehat{\mathbf{h}}_2(n)\right] = 1 - \left[\frac{\mathbf{h}_2^T\widehat{\mathbf{h}}_2(n)}{\|\mathbf{h}_2\|_2 \left\|\widehat{\mathbf{h}}_2(n)\right\|_2}\right]^2, \tag{56}$$

$$\text{NPM}\left[\mathbf{h}_3, \widehat{\mathbf{h}}_3(n)\right] = 1 - \left[\frac{\mathbf{h}_3^T\widehat{\mathbf{h}}_3(n)}{\|\mathbf{h}_3\|_2 \left\|\widehat{\mathbf{h}}_3(n)\right\|_2}\right]^2, \tag{57}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. On the other hand, the global impulse response $\mathbf{h}$ results without any scaling ambiguity. Consequently, we can use a performance measure based on the regular normalized misalignment (NM):

$$\text{NM}\left[\mathbf{h}, \widehat{\mathbf{h}}(n)\right] = \frac{\left\|\mathbf{h} - \widehat{\mathbf{h}}(n)\right\|_2^2}{\|\mathbf{h}\|_2^2}, \tag{58}$$

which is also equivalent to $\left\|\boldsymbol{\mathcal{H}} - \widehat{\boldsymbol{\mathcal{H}}}(n)\right\|_F^2 / \|\boldsymbol{\mathcal{H}}\|_F^2$, where $\widehat{\boldsymbol{\mathcal{H}}}(n) = \widehat{\mathbf{h}}_1(n) \circ \widehat{\mathbf{h}}_2(n) \circ \widehat{\mathbf{h}}_3(n)$ and $\|\cdot\|_F$ denotes the Frobenius norm (the Frobenius norm of a third-order tensor $\boldsymbol{\mathcal{A}}$ is defined as $\|\boldsymbol{\mathcal{A}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle} = \left\|\mathbf{A}_{[1]}\right\|_F = \left\|\mathbf{A}_{[2]}\right\|_F = \left\|\mathbf{A}_{[3]}\right\|_F$).

The simulation results should provide answers to several important questions, as follows. (i) What is the influence of the forgetting factors on the performance of the proposed RLS-TF algorithm? (ii) What are the advantages of the RLS-TF algorithm over the previously developed NLMS-TF counterpart [17]? (iii) What are the advantages of the RLS-TF algorithm over the conventional RLS benchmark? The following three experiments are designed to address these issues.

In the first experiment, the performance of the proposed RLS-TF algorithm was analysed with respect to its main parameters, i.e., the forgetting factors λ_1, λ_2, and λ_3. In the case of an RLS-based algorithm, the value of the forgetting factor is usually related to the filter length, following a well-known rule of thumb, as shown in Table 1 (see "Initialization"). In our case, the forgetting factors of the RLS-TF algorithm were set to $\lambda_1 = 1 - 1/(KL_1)$, $\lambda_2 = 1 - 1/(KL_2)$, and $\lambda_3 = 1 - 1/(KL_3)$. As we can notice, the value of each forgetting factor depended on the length of its related filter (i.e., L_1, L_2, or L_3), but also on the constant K. This tuning parameter could be used to adjust the values of the forgetting factors, as indicated in Figures 3 and 4. Clearly, a higher value of K would result in a higher value of the forgetting factor (i.e., closer to one). We could expect that a higher value of the forgetting factor would reduce the misalignment, but slowing down the convergence/tracking [29]. On the other hand, reducing the forgetting factor improves the convergence/tracking, but increasing the misalignment. This behaviour was supported by the results depicted in Figures 3 and 4, in terms of NPM and NM, respectively. As we can notice, the value $K = 20$ lead to a good compromise between the performance criteria, so that it would be used in the following experiments.

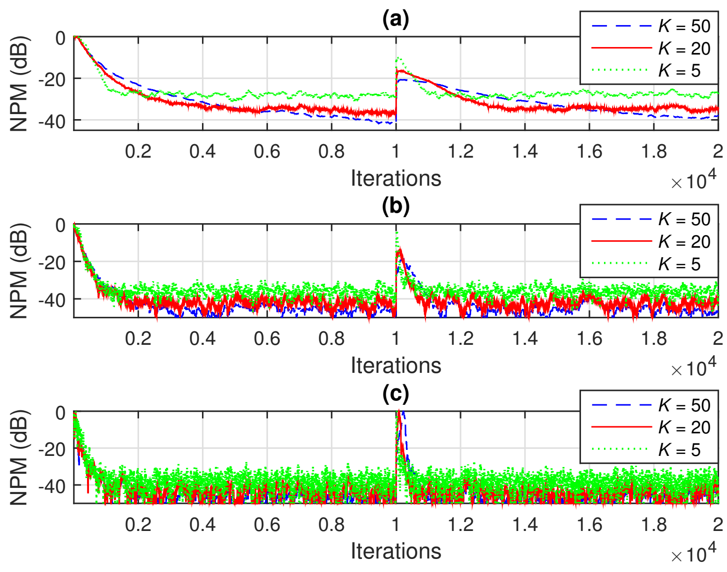

Figure 3. Normalized projection misalignment (NPM) evaluated based on (55)–(57), in dB, for the identification of the individual impulse responses from Figure 2a–c, using the RLS-TF algorithm with different values of the forgetting factors $\lambda_i = 1 - 1/(KL_i)$, $i = 1, 2, 3$ (varying the value of K): (**a**) NPM $\left[\mathbf{h}_1, \widehat{\mathbf{h}}_1(n)\right]$, (**b**) NPM $\left[\mathbf{h}_2, \widehat{\mathbf{h}}_2(n)\right]$, and (**c**) NPM $\left[\mathbf{h}_3, \widehat{\mathbf{h}}_3(n)\right]$.

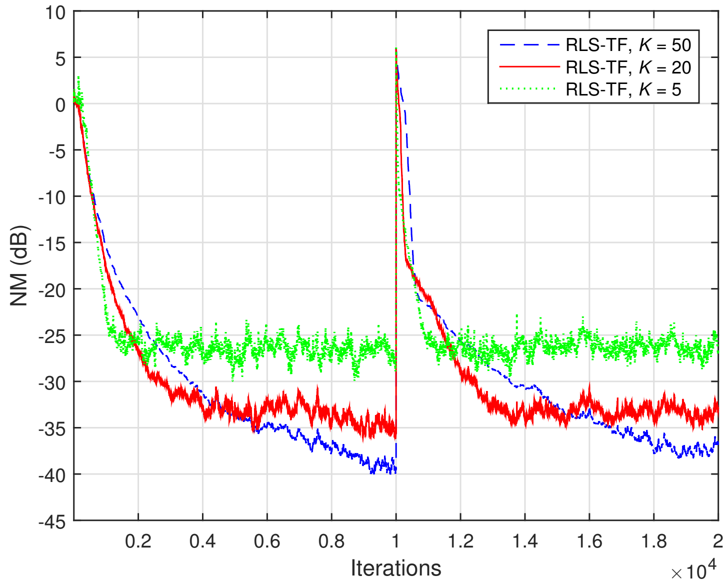

Figure 4. Normalized misalignment (NM) evaluated based on (58), in dB, for the identification of the global impulse response $\mathbf{h}$ (of length $L = 2048$) from Figure 2d, using the RLS-TF algorithm with different values of the forgetting factors $\lambda_i = 1 - 1/(KL_i)$, $i = 1, 2, 3$ (varying the value of K).

Next, we compare the performance of the proposed RLS-TF algorithm with its previously developed counterpart based on the NLMS algorithm, i.e., NLMS-TF [17]. The overall performance of this algorithm is mainly controlled by its normalized step-sizes, which are positive constants smaller than one. Using notation similar to that involved in [17], we set equal values for these parameters, i.e., $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$. In the case of the NLMS-TF algorithm, the fastest convergence mode was obtained when $\alpha_1 + \alpha_2 + \alpha_3 \approx 1$, so that we could use $\alpha = 0.33$. Smaller values of the normalized step-sizes (e.g., $\alpha = 0.1$) reduced the convergence/tracking, but led to a lower misalignment. As shown in Figures 5 and 6 (in terms of NPM and NM, respectively), the RLS-TF algorithm clearly outperformed the NLMS-TF counterpart, achieving a faster convergence rate and tracking, together with low misalignment.

Finally, in the last experiment, we investigated the comparison between the RLS-TF solution and the conventional RLS algorithm. As explained in the last part of Section 3 (related to the computational complexity), the conventional RLS algorithm could also be used for the identification of the global impulse response of length L, based on the cost function from (36). This algorithm uses a single forgetting factor, which can also be set as $\lambda = 1 - 1/(KL)$, where K is the same tuning constant. The influence of the value of λ on the performance of the algorithm is also related to the well-known compromise between low misalignment and fast tracking. In the experiment reported in Figure 7, the conventional RLS algorithm uses two values of the forgetting factor, which were set by varying the tuning constant to $K = 1$ and $K = 20$. As we can notice, even if the largest value of the forgetting factor (obtained for $K = 20$) led to a lower misalignment, the tracking capability of the conventional RLS algorithm was significantly reduced. Clearly, the tracking was improved when using a smaller forgetting factor (corresponding to $K = 1$), but the misalignment of the conventional RLS algorithm was much higher in this case. On the other hand, the RLS-TF algorithm outperformed by far its conventional counterpart, in terms of both performance criteria. Moreover, the complexity of the conventional RLS algorithm, i.e., $\mathcal{O}(L^2)$, was prohibitive for practical implementations, due to the long length of the global filter ($L = 2048$). On the other hand, the RLS-TF algorithm worked on the individual components and combined the solutions of three shorter filters, of lengths L_1, L_2, and L_3 (with $L_1 L_2 L_3 = L$); thus, it was much more computationally efficient. As a trivial example related to the last experiment given in Figure 7, we could mention that the simulation time (using MATLAB) of the RLS-TF algorithm was less than one minute, while the conventional RLS algorithm took hours to reach the final result.

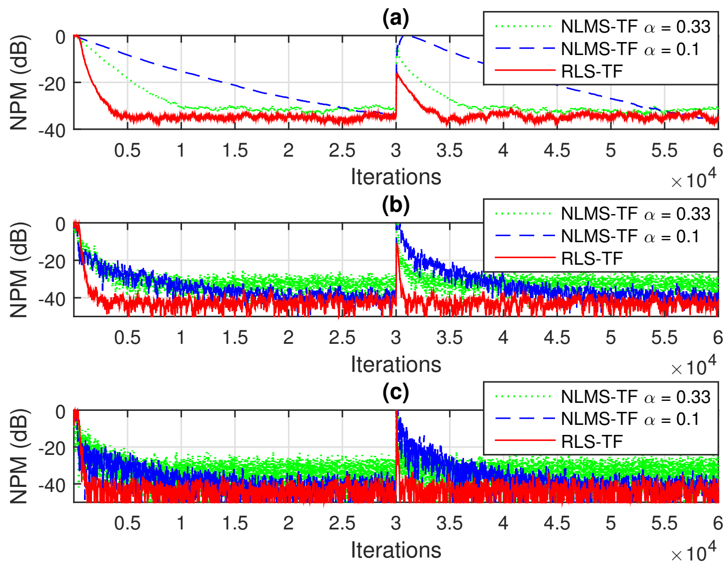

Figure 5. Normalized projection misalignment (NPM) evaluated based on (55)–(57), in dB, for the identification of the individual impulse responses from Figure 2a–c, using the NLMS-TF algorithm [17] (with different normalized step-sizes $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$) and the RLS-TF algorithm (with the forgetting factors $\lambda_i = 1 - 1/(KL_i)$, $i = 1, 2, 3$, where $K = 20$): (**a**) NPM $\left[\mathbf{h}_1, \widehat{\mathbf{h}}_1(n)\right]$, (**b**) NPM $\left[\mathbf{h}_2, \widehat{\mathbf{h}}_2(n)\right]$, and (**c**) NPM $\left[\mathbf{h}_3, \widehat{\mathbf{h}}_3(n)\right]$.

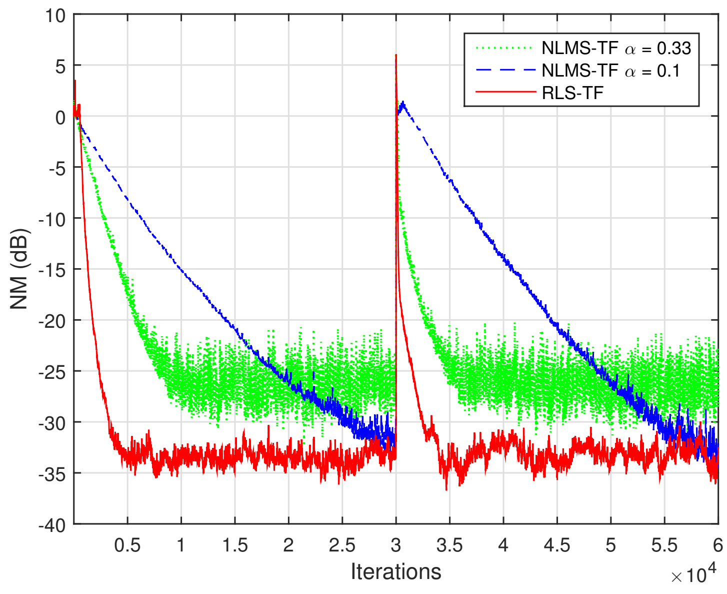

Figure 6. Normalized misalignment (NM) evaluated based on (58), in dB, for the identification of the global impulse response $\mathbf{h}$ (of length $L = 2048$) from Figure 2d, using the NLMS-TF algorithm [17] (with different normalized step-sizes $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$) and the RLS-TF algorithm (with the forgetting factors $\lambda_i = 1 - 1/(KL_i)$, $i = 1, 2, 3$, where $K = 20$).

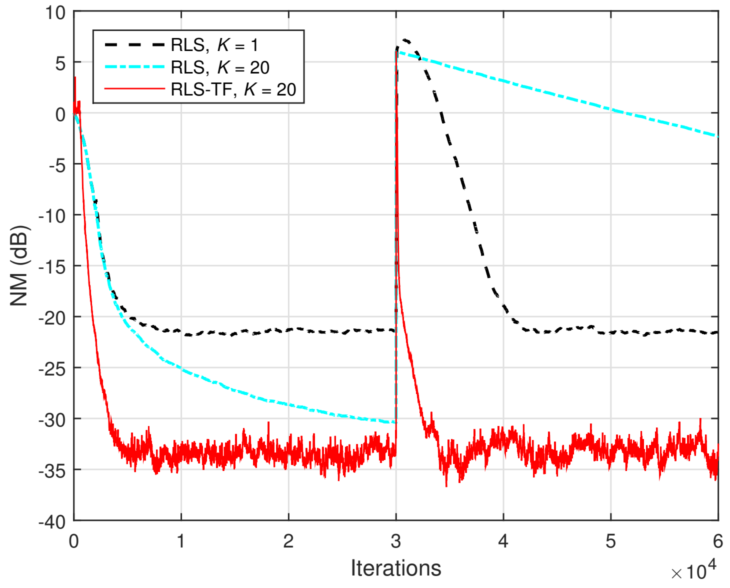

Figure 7. Normalized misalignment (NM) evaluated based on (58), in dB, for the identification of the global impulse response **h** (of length $L = 2048$) from Figure 2d, using the conventional RLS algorithm (with different values of the forgetting factor $\lambda = 1 - 1/(KL)$, varying the value of K) and the RLS-TF algorithm (with the forgetting factors $\lambda_i = 1 - 1/(KL_i)$, $i = 1, 2, 3$, where $K = 20$).

5. Conclusions and Future Works

In this paper, we explored the identification of trilinear forms using the RLS algorithm. The approach was developed in the framework of an MISO system identification problem, based on the decomposition and modelling of third-order tensors. The resulting RLS-TF algorithm was tailored for the identification of such trilinear forms in a more efficient way as compared to the conventional RLS algorithm. Moreover, the proposed RLS-TF algorithm outperformed its previously developed NLMS-TF counterpart in terms of the main performance criteria, providing a faster convergence rate and tracking, together with low misalignment.

The ideas presented in this paper could be further exploited in an extended framework, aiming to identify more general forms of global impulse responses, which cannot be decomposed as rank-1 tensors. Several preliminary results can be found in [30–32], but they are applicable only in the bilinear context (i.e., second-order tensors). The extension to the trilinear case represents a very challenging problem, since finding (and approximating) the rank of a higher order tensor is a much more sensitive task. Furthermore, it would be interesting to extend other versions of the NLMS and RLS algorithms (e.g., based on variable step-sizes [33] and variable forgetting factors [34], respectively) to the trilinear forms.

Finally, it would be useful to evaluate how the proposed algorithm could benefit (in terms of implementation) from the current technology of the tensor processing unit (TPU) and the TensorFlow software [35]. This aspect could bring additional advantages, especially in the framework of specific applications related to machine learning/vision, neural networks, and artificial intelligence.

Author Contributions: Conceptualization, C.E.-I.; methodology, L.-M.D.; investigation, C.P.; validation, J.B.; software, A.-A.E.; formal analysis, S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by UEFISCDI grant number PN-III-P1-1.1-PD-2019-0340.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Comon, P. Tensors: A brief introduction. *IEEE Signal Process. Mag.* **2014**, *31*, 44–53. [CrossRef]
2. Cichocki, A.; Mandic, D.; De Lathauwer, L.; Zhou, G.; Zhao, Q.; Caiafa, C.; Phan, H.A. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Process. Mag.* **2015**, *32*, 145–163. [CrossRef]
3. Vervliet, N.; Debals, O.; Sorber, L.; De Lathauwer, L. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Process. Mag.* **2014**, *31*, 71–79. [CrossRef]
4. Boussé, M.; Debals, O.; De Lathauwer, L. A tensor-based method for large-scale blind source separation using segmentation. *IEEE Trans. Signal Process.* **2017**, *65*, 346–358. [CrossRef]
5. Sidiropoulos, N.; De Lathauwer, L.; Fu, X.; Huang, K.; Papalexakis, E.; Faloutsos, C. Tensor decomposition for signal processing and machine learning. *IEEE Trans. Signal Process.* **2017**, *65*, 3551–3582. [CrossRef]
6. da Costa, M.N.; Favier, G.; Romano, J.M.T. Tensor modelling of MIMO communication systems with performance analysis and Kronecker receivers. *Signal Process.* **2018**, *145*, 304–316. [CrossRef]
7. Ribeiro, L.N.; de Almeida, A.L.; Mota, J.C.M. Separable linearly constrained minimum variance beamformers. *Signal Process.* **2019**, *158*, 15–25. [CrossRef]
8. De Lathauwer, L. Signal Processing Based on Multilinear Algebra. Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.
9. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455–500. [CrossRef]
10. Benesty, J.; Paleologu, C.; Ciochină, S. On the identification of bilinear forms with the Wiener filter. *IEEE Signal Process. Lett.* **2017**, *24*, 653–657. [CrossRef]
11. Paleologu, C.; Benesty, J.; Ciochină, S. Adaptive filtering for the identification of bilinear forms. *Digit. Signal Process.* **2018**, *75*, 153–167. [CrossRef]
12. Elisei-Iliescu, C.; Stanciu, C.; Paleologu,C.; Benesty, J.; Anghel, C.; Ciochină, S. Efficient recursive least-squares algorithms for the identification of bilinear forms. *Digit. Signal Process.* **2018**, *83*, 280–296. [CrossRef]
13. Dogariu, L.-M.; Ciochină, S.; Paleologu, C.; Benesty, J. A connection between the Kalman filter and an optimized LMS algorithm for bilinear forms. *Algorithms* **2018**, *11*, 211. [CrossRef]
14. Ribeiro, L.N.; de Almeida, A.L.F.; Mota, J.C.M. Identification of separable systems using trilinear filtering. In Proceedings of the 2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Cancun, Mexico, 13–16 December 2015; pp. 189–192.
15. Rupp, M.; Schwarz, S. A tensor LMS algorithm. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, Australia, 19–24 April 2015; pp. 3347–3351.
16. Dogariu, L.-M.; Ciochină, S.; Benesty, J.; Paleologu, C. An iterative Wiener filter for the identification of trilinear forms. In Proceedings of the 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 1–3 July 2019; pp. 88–93.
17. Dogariu, L.-M.; Ciochină, S.; Benesty, J.; Paleologu, C. System identification based on tensor decompositions: A trilinear approach. *Symmetry* **2019**, *11*, 556. [CrossRef]
18. Haykin, S. *Adaptive Filter Theory*, 4th ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2002.
19. Parathai, P.; Tengtrairat, N.; Woo, W.L.; Gao, B. Single-channel signal separation using spectral basis correlation with sparse nonnegative tensor factorization. *Circuits Syst. Signal Process.* **2019**, *38*, 5786–5816. [CrossRef]
20. Woo, W.L.; Dlay, S.S.; Al-Tmeme, A.; Gao, B. Reverberant signal separation using optimized complex sparse nonnegative tensor deconvolution on spectral covariance matrix. *Digit. Signal Process.* **2018**, *83*, 9–23. [CrossRef]
21. Gao, B.; Lu, P.; Woo, W.L.; Tian, G.Y.; Zhu, Y.; Johnston, M. Variational Bayes sub-group adaptive sparse component extraction for diagnostic imaging system. *IEEE Trans. Ind. Electron.* **2018**, *65*, 8142–8152. [CrossRef]
22. Kiers, H.A.L. Towards a standardized notation and terminology in multiway analysis. *J. Chemom.* **2000**, *14*, 105–122. [CrossRef]
23. Kroonenberg, P. *Applied Multiway Data Analysis*; Wiley: Hoboken, NJ, USA, 2008.

24. Van Loan, C.F. The ubiquitous Kronecker product. *J. Comput. Appl. Math.* **2000**, *123*, 85–100. [CrossRef]
25. Bertsekas, D.P. *Nonlinear Programming*, 2nd ed.; Athena Scientific: Belmont, MA, USA, 1999.
26. *Digital Network Echo Cancellers*; ITU-T Recommendations G.168; ITU: Geneva, Switzerland, 2002.
27. Gay, S.L.; Benesty, J. (Eds.) *Acoustic Signal Processing for Telecommunication*; Kluwer Academic Publisher: Boston, MA, USA, 2000.
28. Morgan, D.R.; Benesty, J.; Sondhi, M.M. On the evaluation of estimated impulse responses. *IEEE Signal Process. Lett.* **1998**, *5*, 174–176. [CrossRef]
29. Ciochină, S.; Paleologu, C.; Benesty, J.; Enescu, A. A. On the influence of the forgetting factor of the RLS adaptive filter in system identification. In Proceedings of the 2009 International Symposium on Signals, Circuits and Systems, Iasi, Romania, 9–10 July 2009; pp. 205–208.
30. Paleologu, C.; Benesty, J.; Ciochină, S. Linear system identification based on a Kronecker product decomposition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2018**, *26*, 1793–1808. [CrossRef]
31. Elisei-Iliescu, C.; Paleologu, C.; Benesty, J.; Ciochină, S. A recursive least-squares algorithm based on the nearest Kronecker product decomposition. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 4843–4847.
32. Elisei-Iliescu, C.; Paleologu, C.; Benesty, J.; Stanciu, C.; Anghel, C.; Ciochină, S. Recursive least-squares algorithms for the identification of low-rank systems. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2019**, *27*, 903–918. [CrossRef]
33. Benesty, J.; Rey, H.; Rey Vega, L.; Tressens, S. A non-parametric VSS NLMS algorithm. *IEEE Signal Process. Lett.* **2006**, *13*, 581–584. [CrossRef]
34. Paleologu, C.; Benesty, J.; Ciochină, S. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Process. Lett.* **2008**, *15*, 597–600. [CrossRef]
35. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture, Toronto, ON, Canada, 24–28 June 2017; pp. 1–12.