electronics

MDPI

Article

Cascaded RLS Adaptive Filters Based on a Kronecker Product Decomposition

Alexandru-George Rusu [1,2], Silviu Ciochină [1], Constantin Paleologu [2,*] and Jacob Benesty [3]

[1] Department of Telecommunications, University Politehnica of Bucharest, 061071 Bucharest, Romania; alexandru.rusu@rohde-schwarz.com (A.-G.R.); silviu@comm.pub.ro (S.C.)
[2] Department of Research and Development, Rohde & Schwarz Topex, 020335 Bucharest, Romania
[3] INRS-EMT, University of Quebec, Montreal, QC H5A 1K6, Canada; Jacob.Benesty@inrs.ca
* Correspondence: pale@comm.pub.ro

Abstract: The multilinear system framework allows for the exploitation of the system identification problem from different perspectives in the context of various applications, such as nonlinear acoustic echo cancellation, multi-party audio conferencing, and video conferencing, in which the system could be modeled through parallel or cascaded filters. In this paper, we introduce different memoryless and memory structures that are described from a bilinear perspective. Following the memory structures, we develop the multilinear recursive least-squares algorithm by considering the Kronecker product decomposition concept. We have performed a set of simulations in the context of echo cancellation, aiming both long length impulse responses and the reverberation effect.

Keywords: recursive least-squares (RLS) algorithm; adaptive filters; Kronecker product decomposition; system identification; echo cancellation

Citation: Rusu, A.-G.; Ciochină, S.; Paleologu, C.; Benesty, J. Cascaded RLS Adaptive Filters Based on a Kronecker Product Decomposition. *Electronics* **2022**, *11*, 409. https://doi.org/10.3390/electronics11030409

Academic Editor: Manohar Das

Received: 10 December 2021
Accepted: 27 January 2022
Published: 29 January 2022

1. Introduction

In the field of system identification, many applications involve adaptive filtering algorithms [1,2]. One of them is the echo cancellation problem, which has raised many challenges over the years [3,4]. Based on the input-output relation, a dynamic system should be determined (i.e., the echo path), considering various parameters and external factors that must be estimated. These dynamic systems are modeled linearly through an adaptive filter with a finite-impulse-response (FIR) structure [5,6]. The main performance bottlenecks, in terms of computational complexity, tracking, and convergence rate, arise when the length of the impulse response reaches hundreds/thousands of coefficients. The literature presents many approaches to improve the overall performance, also taking into account the fact that the echo paths are sparse in nature [7–13]. Recently, in our previous work [14], we introduced a new approach of splitting a long length impulse response into several impulse responses of shorter lengths, aiming to reduce the computational complexity by maintaining the overall performance. Another challenge arises when the echo path produces multiple reflections, and this effect is called reverberation. From a mathematical point of view, this effect could be described (to some extent) by using the Kronecker product decomposition of the impulse response [15,16].

In this paper, we extend our study on cascaded adaptive filters, aiming to reduce the computational complexity considering both long length impulse responses and the reverberation effect. Our approach is based on multilinear structures and the Kronecker product decomposition. The main goal is to outline the features of this development and its potential.

The rest of the paper is organized as follows. Section 2 presents the background for different bilinear structures without memory, while Section 3 introduces bilinear structures with memory. In Section 4, the new development is combined with the recursive least-

squares (RLS) algorithm, thus resulting in a practical solution based on adaptive filtering. We perform an experimental study in Section 5 and conclude the paper in Section 6.

2. Bilinear Structures without Memory

In order to introduce the bilinear structures with memory and the development based on the Kronecker product decomposition, let us start by presenting the bilinear structure without memory [14,17,18], defined as

$$y(n) = \mathbf{h}_1^T \mathbf{X}(n) \mathbf{h}_2 = \sum_{l_1=1}^{L_1} \sum_{l_2=1}^{L_2} x_{l_1 l_2}(n) h_{1,l_1} h_{2,l_2}, \tag{1}$$

where $\mathbf{X}(n)$ is the multiple input data matrix of size $L_1 \times L_2$, with

$$\mathbf{X}(n) = \left[\begin{array}{cccccc} \mathbf{x}_1(n) & \cdots & \mathbf{x}_{l_2}(n) & \cdots & \mathbf{x}_{L_2}(n) \end{array} \right], \tag{2}$$

and

$$\mathbf{x}_{l_2}(n) = \left[\begin{array}{cccccc} x_{l_2,1}(n) & \cdots & x_{l_2,l_1}(n) & \cdots & x_{l_2,L_1}(n) \end{array} \right]^T, \; l_2 = 1, 2, \ldots, L_2 \tag{3}$$

is an input signal vector containing the L_1 most recent data at the discrete-time index n, while the superscript ^{T} is the transpose operator. The two impulse responses $\mathbf{h}_1$ and $\mathbf{h}_2$ have L_1 and L_2 coefficients, respectively. In other words, the input-output equation in (1) describes a system with $L_1 L_2$ inputs and a single output. In order to facilitate the graphical representation, let us rewrite (1) as

$$\begin{aligned} y(n) &= \sum_{l_1=1}^{L_1} h_{1,l_1} \sum_{l_2=1}^{L_2} h_{2,l_2} x_{l_1 l_2}(n) = \sum_{l_2=1}^{L_2} h_{2,l_2} \sum_{l_1=1}^{L_1} h_{1,l_1} x_{l_1 l_2}(n) \\ &= \sum_{l_2=1}^{L_2} h_{2,l_2} s_{1,l_2}(n), \end{aligned} \tag{4}$$

where

$$s_{1,l_2}(n) = \sum_{l_1=1}^{L_1} h_{1,l_1} x_{l_1 l_2}(n) \tag{5}$$

is the output of a memoryless weighted adder with L_1 inputs at the discrete-time index n. We can transpose (5) in a graphical representation as shown in Figure 1.

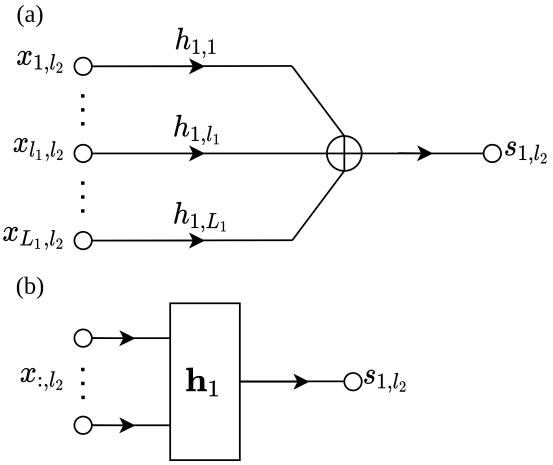

Figure 1. (**a**) The structure of $s_{1,l_2}(n)$ [see (5)] and (**b**) the symbolic representation of the $s_{1,l_2}(n)$ memoryless weighted adder.

Based on (4) and Figure 1, we can introduce the graphical respresentation of the multiple-input single-output (MISO) system as shown in Figure 2. Overall, this structure consists of two levels of combiners.

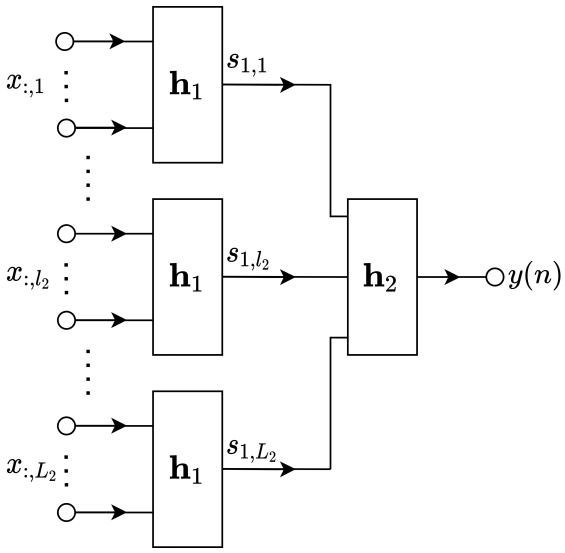

Figure 2. The two combiner levels structure (i.e., a MISO system).

3. Bilinear Structures with Memory

By introducing a delay line, the $s_{1,l_2}(n)$ structure described by L_1 inputs and a single output can be transformed in a single-input single-output (SISO) structure. Therefore, the following input signal vector results

$$\mathbf{x}_{l_2}(n) = \left[\begin{array}{cccccc} x_{l_2}(n) & \cdots & x_{l_2}(n-l_1+1) & \cdots & x_{l_2}(n-L_1+1) \end{array} \right]^T, \; l_2 = 1, 2, \ldots, L_2. \quad (6)$$

Thus, the input data matrix has the following structure:

$$\mathbf{X}(n) = \begin{bmatrix} x_1(n) & \cdots & x_{l_2}(n) & \cdots & x_{L_2}(n) \\ x_1(n-1) & \cdots & x_{l_2}(n-1) & \cdots & x_{L_2}(n-1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1(n-L_1+1) & \cdots & x_{l_2}(n-L_1+1) & \cdots & x_{L_2}(n-L_1+1) \end{bmatrix}.$$

Hence,

$$s_{1,l_2}(n) = \sum_{l_1=1}^{L_1} h_{1,l_1} x_{l_2}(n-l_1+1) \quad (7)$$

is a structure associated to a transversal filter, with the weighted function $\mathbf{h}_1$, having as input the vector $\mathbf{x}_{l_2}(n)$. The graph representation of the new $s_{1,l_2}(n)$ structure is shown in Figure 3. Also, Figure 4 outlines the two combiner level structures based on the transversal filters.

In terms of the z-transform, (7) is defined as

$$\begin{aligned} S_{1,l_2}(z) &= \sum_{n=-\infty}^{\infty} s_{1,l_2}(n) z^{-n} = \sum_{l_1=1}^{L_1} h_{1,l_1} \sum_{n=-\infty}^{\infty} x_{l_2}(n-l_1+1) z^{-n} \\ &= X_{l_2}(z) \sum_{l_1=1}^{L_1} h_{1,l_1} z^{-l_1+1} \\ &= X_{l_2}(z) H_1(z), \end{aligned} \quad (8)$$

where

$$H_1(z) = \sum_{l_1=1}^{L_1} h_{1,l_1} z^{-l_1+1}. \tag{9}$$

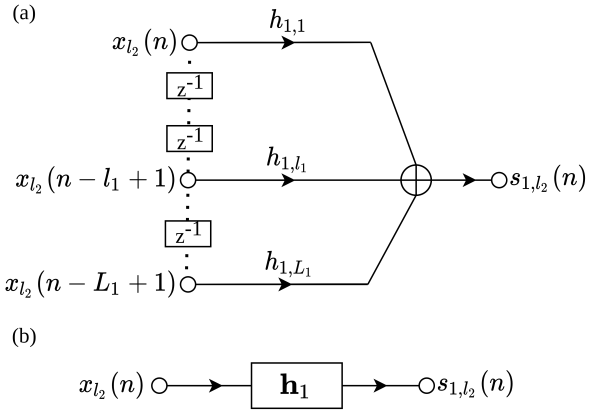

Figure 3. (**a**) The structure of $s_{1,l_2}(n)$ with delay line and (**b**) its symbolic representation.

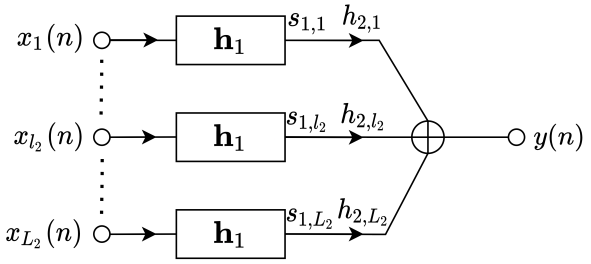

Figure 4. The two combiner levels structure based on transversal filters.

A more efficient form in terms of correlation between the columns of the input matrix $\mathbf{X}(n)$ can be obtained if we consider successive data related to the columns, and is defined as

$$\mathbf{X}(n) = \begin{bmatrix} x_1(n) & \cdots & x_1(n-(l_2-1)L_1) & \cdots & x_1(n-(L_2-1)L_1) \\ x_1(n-1) & \cdots & x_1(n-(l_2-1)L_1-1) & \cdots & x_1(n-(L_2-1)L_1-1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1(n-L_1+1) & \cdots & x_1(n-(l_2-1)L_1-L_1+1) & \cdots & x_1(n-(L_2-1)L_1-L_1+1) \end{bmatrix}.$$

The input signal vector becomes a sequence of L_1L_2 successive data applied to a FIR filter of length L_1L_2, so that

$$\mathbf{x}(n) = \text{vec}[\mathbf{X}(n)], \tag{10}$$

where $\text{vec}(\cdot)$ denotes the vectorization operation and

$$s_{1,l_2}(n) = \sum_{l_1=1}^{L_1} h_{1,l_1} x_1(n-l_1L_2), \tag{11}$$

with $x_1(n) = x_1(n-L_1), \ldots, x_{l_2}(n) = x_1(n-l_1L_2), \ldots, x_{L_2}(n) = x_1(n-L_1L_2)$. In terms of the z-transform, we can write (11) as

$$\begin{aligned} S_{1,l_2}(z) &= \sum_{l_1=1}^{L_1} X_1(z) z^{-l_1 L_2} h_{1,l_1} \\ &= X_1(z) \sum_{l_1=1}^{L_1} z^{-l_1 L_2} h_{1,l_1} = X_1(z) H_1\left(z^{L_2}\right). \end{aligned} \tag{12}$$

Forwards, the output of the global system in the z-transform domain is

$$Y(z) = S_{1,l_2}(z) H_2(z) = X_1(z) H_1\left(z^{L_2}\right) H_2(z) = X_1(z) H(z), \tag{13}$$

where

$$H(z) = H_1\left(z^{L_2}\right) H_2(z). \tag{14}$$

Equation (12) describes a SISO structure of two cascaded filters as shown in Figure 5. The first filter $H_1\left(z^{L_2}\right)$ is obtained through interpolation with zeroes by L_2 factor of the $H_1(z)$ function and its length is $L_2(L_1-1)+1$. Indeed, it has only L_1 non-zero coefficients, from the total of $L_2(L_1-1)+1$, (hence a certain degree of sparsity), according to its impulse response

$$\mathbf{h}_1^{'}(n) = \begin{cases} \mathbf{h}_1(\frac{n}{L_1}), & \text{for } n \bmod L_1 = 0, n = 1, 2, \ldots, L_2(L_1-1)+1 \\ 0, & \text{otherwise} \end{cases}, \tag{15}$$

where $\mathbf{h}_1^{'}(n)$ is the impulse response of $H_1\left(z^{L_2}\right)$ and mod denotes the modulo operation. The second filter, $H_2(z)$, is of length L_2. Afterwards, the total length of the $H(z)$ filter is $L_2(L_1-1)+1+L_2-1 = L_1 L_2$.

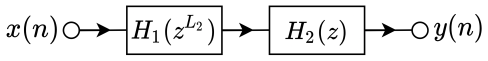

Figure 5. SISO system in cascaded configuration.

Based on this configuration, let us consider the two vectors:

$$\mathbf{h}_1 = \begin{bmatrix} h_{1,1} \\ h_{1,2} \\ \vdots \\ h_{1,l_1} \\ \vdots \\ h_{1,L_1} \end{bmatrix}, \ \mathbf{h}_2 = \begin{bmatrix} h_{2,1} \\ h_{2,2} \\ \vdots \\ h_{2,l_2} \\ \vdots \\ h_{2,L_2} \end{bmatrix},$$

and the Kronecker product:

$$\mathbf{h} = \mathbf{h}_1 \otimes \mathbf{h}_2 = \begin{bmatrix} h_{1,1}\mathbf{h}_2 \\ h_{1,2}\mathbf{h}_2 \\ \vdots \\ h_{1,l_1}\mathbf{h}_2 \\ \vdots \\ h_{1,L_1}\mathbf{h}_2 \end{bmatrix}.$$

Having as coefficients the elements of this vector, the polynomial form is developed as

$$H(z) = \sum_{r=0}^{L_1 L_2 - 1} h_r z^{-r} \tag{16}$$

and can be factored in the form described in (14). While the position of an element for the l_1, l_2 indexes is $r = (l_1 - 1)L_2 + l_2 - 1$, we have

$$\begin{aligned} H_1\left(z^{L_2}\right) H_2(z) &= \sum_{l_1=1}^{L_1} h_{1,l_1} z^{-(l_1-1)L_2} \sum_{l_2=1}^{L_2} h_{2,l_2} z^{-(l_2-1)} \\ &= \sum_{l_1=1}^{L_1} \sum_{l_2=1}^{L_2} h_{1,l_1} h_{2,l_2} z^{-(l_1-1)L_2-(l_2-1)} \\ &= \sum_{r=0}^{L_1 L_2 - 1} h_r z^{-r}. \end{aligned} \tag{17}$$

4. Cascaded Multilinear RLS Algorithm Using Kronecker Product Decomposition

Based on the development from Section 3, we introduce the set of equations for the RLS algorithm in a multilinear manner, following the Kronecker product decomposition [14,19]. Our approach is determined considering the system identification framework. In this context, the output of the MISO system is

$$y(n) = \boldsymbol{\mathcal{X}}(n) \times_1 \mathbf{h}_1^T \times_2 \mathbf{h}_2^T \times_i \cdots \times_N \mathbf{h}_N^T, \tag{18}$$

where N denotes the multilinear degree and $\times_i$ represents the multiplication operation by the dimension $i = 1, 2, \ldots, N$. The input data are described in a N degree tensorial form as $[\boldsymbol{\mathcal{X}}(n)]_{l_1 l_2 \ldots l_N}$ with the real-values $x_{l_1 l_2 \ldots l_N}$, $l_i = 1, 2, \ldots, L_i$, $i = 1, 2, \ldots, N$. The vector $\mathbf{h}_i$ of length L_i, stores the impulse response for the i cascaded filter, $i = 1, 2, \ldots, N$. Based on the $\mathbf{h}_i$ ($i = 1, 2, \ldots, N$) impulse responses of the MISO system, the rank-1 tensor of dimension $L_1 \times L_2 \times \cdots \times L_N$ is

$$\boldsymbol{\mathcal{H}} = \mathbf{h}_1 \circ \mathbf{h}_2 \circ \cdots \circ \mathbf{h}_N, \tag{19}$$

where $\circ$ denotes the outer product. Usually, in the context of system identification, the desired signal results from the output signal corrupted by an additive noise, $w(n)$, which in our development is a zero-mean Gaussian signal, so that

$$d(n) = y(n) + w(n). \tag{20}$$

Consequently, the output signal described by (18) results in

$$y(n) = \text{vec}^T[\boldsymbol{\mathcal{X}}(n)]\text{vec}(\boldsymbol{\mathcal{H}}), \tag{21}$$

where

$$\text{vec}[\boldsymbol{\mathcal{X}}(n)] = \begin{bmatrix} \text{vec}[\mathbf{X}_{::\cdots:L_1}(n)] \\ \text{vec}[\mathbf{X}_{::\cdots:L_2}(n)] \\ \vdots \\ \text{vec}[\mathbf{X}_{::\cdots:L_N}(n)] \end{bmatrix} \triangleq \widetilde{\mathbf{x}}(n)$$

and

$$\text{vec}(\boldsymbol{\mathcal{H}}) = \begin{bmatrix} \text{vec}(\mathbf{H}_{::\cdots:L_1}) \\ \text{vec}(\mathbf{H}_{::\cdots:L_2}) \\ \vdots \\ \text{vec}(\mathbf{H}_{::\cdots:L_N}) \end{bmatrix} = \mathbf{h}_N \otimes \cdots \otimes \mathbf{h}_2 \otimes \mathbf{h}_1 \triangleq \mathbf{h},$$

with $\mathbf{X}_{::\cdots:l_i}(n) \in \mathbb{R}^{L_1 \times L_2 \times \cdots \times L_{N-1}}$ and $\mathbf{H}_{::\cdots:l_i}(n) \in \mathbb{R}^{L_1 \times L_2 \times \cdots \times L_{N-1}}$ representing the frontal slices of $\boldsymbol{\mathcal{X}}(n)$ and $\boldsymbol{\mathcal{H}}(n)$, respectively. The two new vectors $\widetilde{\mathbf{x}}(n)$ and $\mathbf{h}$ consist of $L_1 L_2 \cdots L_N$ elements. Also, the output of the system can be rewritten as

$$y(n) = \widetilde{\mathbf{x}}^T(n)\mathbf{h}. \tag{22}$$

Then, the a priori error signal is computed as

$$e(n) = d(n) - \hat{y}(n), \tag{23}$$

where $\hat{y}(n)$ represents an estimate of the output signal. Following the least-squares (LS) error criterion, we can introduce the cost functions:

$$\begin{aligned}
\mathcal{J}[\hat{\mathbf{h}}_1(n)] &= \sum_{k=1}^{n} \lambda_1^{n-k}[d(n) - \hat{\mathbf{h}}_1^T(n)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)]^2, \\
\mathcal{J}[\hat{\mathbf{h}}_2(n)] &= \sum_{k=1}^{n} \lambda_2^{n-k}[d(n) - \hat{\mathbf{h}}_2^T(n)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)]^2, \\
&\vdots \\
\mathcal{J}[\hat{\mathbf{h}}_N(n)] &= \sum_{k=1}^{n} \lambda_N^{n-k}[d(n) - \hat{\mathbf{h}}_N^T(n)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)]^2,
\end{aligned} \tag{24}$$

where $0 < \lambda_i \leq 1$, $i = 1, 2, \ldots, N$, represent the forgetting factors and

$$\begin{aligned}
\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= [\hat{\mathbf{h}}_N(n-1) \otimes \cdots \otimes \hat{\mathbf{h}}_2(n-1) \otimes \mathbf{I}_{L_1}]^T\widetilde{\mathbf{x}}(n), \\
\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= [\hat{\mathbf{h}}_N(n-1) \otimes \cdots \otimes \mathbf{I}_{L_2} \otimes \hat{\mathbf{h}}_1(n-1)]^T\widetilde{\mathbf{x}}(n), \\
&\vdots \\
\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n) &= [\mathbf{I}_{L_N} \otimes \cdots \otimes \hat{\mathbf{h}}_2(n-1) \otimes \hat{\mathbf{h}}_1(n-1)]^T\widetilde{\mathbf{x}}(n),
\end{aligned} \tag{25}$$

with $\mathbf{I}_{L_i}$ denoting the identity matrix of size $L_i \times L_i$, $i = 1, 2, \ldots, N$. Following the minimization of the cost functions $\mathcal{J}[\hat{\mathbf{h}}_1(n)]$, $\mathcal{J}[\hat{\mathbf{h}}_2(n)]$,..., $\mathcal{J}[\hat{\mathbf{h}}_N(n)]$, the update equations of the RLS algorithm in the multilinear approach result:

$$\begin{aligned}
\hat{\mathbf{h}}_1(n) &= \hat{\mathbf{h}}_1(n-1) + \mathbf{s}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)e_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n), \\
\hat{\mathbf{h}}_2(n) &= \hat{\mathbf{h}}_2(n-1) + \mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)e_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n), \\
&\vdots \\
\hat{\mathbf{h}}_N(n) &= \hat{\mathbf{h}}_N(n-1) + \mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)e_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n),
\end{aligned} \tag{26}$$

where the a priori errors are defined as

$$\begin{aligned}
e_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= d(n) - \hat{\mathbf{h}}_1^T(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n), \\
e_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= d(n) - \hat{\mathbf{h}}_2^T(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n), \\
&\vdots \\
e_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n) &= d(n) - \hat{\mathbf{h}}_N^T(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n),
\end{aligned} \tag{27}$$

with

$$\begin{aligned}
\mathbf{s}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= \mathbf{P}_{\hat{\mathbf{h}}_1}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)[\lambda_1 + \widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\mathbf{P}_{\hat{\mathbf{h}}_1}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)]^{-1},\\
\mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n) &= \mathbf{P}_{\hat{\mathbf{h}}_2}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)[\lambda_2 + \widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\mathbf{P}_{\hat{\mathbf{h}}_2}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)]^{-1},\\
&\vdots\\
\mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n) &= \mathbf{P}_{\hat{\mathbf{h}}_N}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)\\
&\quad\times[\lambda_N + \widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)\mathbf{P}_{\hat{\mathbf{h}}_N}(n-1)\widetilde{\mathbf{x}}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)]^{-1}
\end{aligned} \tag{28}$$

and

$$\begin{aligned}
\mathbf{P}_{\hat{\mathbf{h}}_1}(n) &= \lambda_1^{-1}\mathbf{P}_{\hat{\mathbf{h}}_1}(n-1) - \lambda_1^{-1}\mathbf{s}_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_2\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\mathbf{P}_{\hat{\mathbf{h}}_1}(n-1),\\
\mathbf{P}_{\hat{\mathbf{h}}_2}(n) &= \lambda_2^{-1}\mathbf{P}_{\hat{\mathbf{h}}_2}(n-1) - \lambda_2^{-1}\mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_3\cdots\hat{\mathbf{h}}_N}(n)\mathbf{P}_{\hat{\mathbf{h}}_2}(n-1),\\
&\vdots\\
\mathbf{P}_{\hat{\mathbf{h}}_N}(n) &= \lambda_N^{-1}\mathbf{P}_{\hat{\mathbf{h}}_N}(n-1) - \lambda_N^{-1}\mathbf{s}_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)\widetilde{\mathbf{x}}^T_{\hat{\mathbf{h}}_1\hat{\mathbf{h}}_2\cdots\hat{\mathbf{h}}_{N-1}}(n)\mathbf{P}_{\hat{\mathbf{h}}_N}(n-1).
\end{aligned} \tag{29}$$

In fact, the equations from (26) represent a multilinear optimization strategy, where $N-1$ impulse responses are considered fixed during the optimization of the remaining one [20]. In other words, in each of the cost functions from Equation (24), for the optimization of $\hat{\mathbf{h}}_i(n)$, we consider that the other $\hat{\mathbf{h}}_j(n)$, with $i \neq j$, are fixed. The initialization of the RLS-based algorithms is influenced by the initialization of the matrix $\mathbf{P}(n)$, which represents a recursive estimate of the inverse of the covariance matrix of the input signal [21]. In fact, this is the initialization factor that controls the initial convergence of the algorithm. Usually, this initialization is $\mathbf{P}(0) = \delta^{-1}\mathbf{I}_L$, where δ is the so-called regularization parameter and I_L is the identity matrix of size $L \times L$. This regularization parameter depends on the length of the filter and the power of the input signal. In the case of the RLS-CKD algorithm, the matrices from (29) should be initialized in a similar manner. However, even if the initialization of the conventional RLS and RLS-CKD algorithms could be different from this point of view, the regularization parameters do not bias the overall performance, since their influence (for n large enough) is negligible due to the forgetting factors (i.e., λ for the conventional RLS and λ_i for the RLS-CKD algorithm), which are positive constants smaller than 1. Finally, the cascaded multilinear RLS algorithm based on the Kronecker product decomposition (RLS-CKD) is defined by Equations (26)–(29). While the classical RLS algorithm involves matrices of size $L \times L$, the RLS-CKD algorithm solves the system identification problem by splitting the long length impulse response in shorter length impulse responses, so that it implies matrices of sizes $L_i \times L_i$, $i = 1, 2, \ldots, N$, where $L = L_1 L_2 \cdots L_N$. The classical RLS algorithm involves a computational complexity of $O(L^2)$. In the case of the RLS-CKD algorithm, the computation complexity results as a sum of $O(L_i^2)$. Following the presented approach, the computational complexity of the RLS-CKD is reduced to $O(L_1^2) + O(L_2^2) + \cdots + O(L_N^2) + O(NL)$, with $N \ll L$. The extra $O(NL)$ computational amount is due to the Kronecker product operations. At this point, we can observe a drastic reduction in computational complexity for the RLS-CKD algorithm as compared to that of the classical RLS algorithm, especially for impulse responses of long length (as in echo cancellation).

5. Simulation Results

In order to simulate the RLS-CKD algorithm, we have chosen two different multilinear degrees, $N = 2$ (bilinear) and $N = 3$ (trilinear), considering the echo cancellation framework. As input signals, we have used white Gaussian noise (i.e., a random process with standard normal distribution, zero mean, and unit variance), an AR(1) process produced by filtering a white Gaussian noise through a first-order system $1/(1 - 0.9z^{-1})$, and a speech sequence, at a sample rate of 8 kHz. For the purpose of these simulations, we have considered that the output of the target system (i.e., the echo signal) is corrupted by white Gaussian noise [i.e., $w(n)$], considering an echo-to-noise ratio (ENR) of 20 dB when the

input signal is a white Gaussian noise or an AR(1) process, and 30 dB when the input signal is a speech sequence. In order to measure the performance, we have used the normalized misalignment in dB, defined as

$$\mathrm{NM}[\mathbf{h}, \hat{\mathbf{h}}(n)] = 20\log_{10}\left[\frac{||\mathbf{h} - \hat{\mathbf{h}}(n)||_2}{||\mathbf{h}||_2}\right], \tag{30}$$

where $||\cdot||_2$ denotes the Euclidean norm and

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}_N(n) \otimes \cdots \otimes \hat{\mathbf{h}}_2(n) \otimes \hat{\mathbf{h}}_1(n). \tag{31}$$

As initialization we have used $\hat{\mathbf{h}}_1(0) = [1\ \mathbf{0}_{L_1-1}^T]^T$ (i.e., the first coefficient is equal to one, which is followed by $L_1 - 1$ zeros), while the other impulse responses $\hat{\mathbf{h}}_j(0)$, with $j = 2, 3, \ldots, N$ are initialized as

$$\hat{\mathbf{h}}_j(0) = \frac{1}{L_j}\mathbf{1}_{L_j}, \tag{32}$$

where $\mathbf{1}_{L_j}$ denotes a column vector with all its L_j elements equal to one. The conventional all-zeros initialization specific to most of the adaptive filtering algorithms cannot be used in the case of tensor-based algorithms, due to connection between the individual filters, as shown in Equation (25). In this case, the initialization $\hat{\mathbf{h}}_i(0) = \mathbf{0}_{L_i}$ ($i = 1, 2, \ldots, N$) would stall the algorithm.

For the first set of simulations that implies the bilinear approach, we have considered the impulse responses depicted in Figure 6. In the first plot, Figure 6a, the first impulse response $\mathbf{h}_1$ from the G168 Recommendation [22] is represented (i.e., a 64 coefficients cluster). Next, Figure 6b depicts the second impulse response $\mathbf{h}_2$, evaluated as $h_{2l_2} = 0.5^{l2-1}$, with $l_2 = 1, 2, \ldots, L_2$, where $L_2 = 8$. The third impulse response is the target that must be determined and is obtained as the Kronecker product between the first two impulse responses, i.e., $\mathbf{h} = \mathbf{h}_2 \otimes \mathbf{h}_1$. This impulse response is similar to the echo produced by an acoustic environment characterized by a reverberation effect and its length is $L = L_1 L_2 = 512$ coefficients. Here, we consider the case of a linearly separable system, which is the benchmark of our approach, and show how it can be efficiently exploited in the framework of system identification problems. The impulse response from Figure 6c could correspond to a channel with echoes. This repetitive (but not periodic) structure could also result if a certain impulse response is followed by its reflections, e.g., as in wireless transmissions. The method allows temporal localization and magnitude estimation of the reflections, considering a temporal grid, without any restrictions of periodicity. However, the tensor-based adaptive algorithms can efficiently model the separable part of the system. The forgetting factor used for the RLS algorithm is computed as $\lambda = 1 - 1/(KL)$, with $K = 10$ in the bilinear context and $K = 1$ in the trilinear context, while for the RLS-CKD algorithm is computed as $\lambda_i = 1 - 1/(MKL_i)$, $i = 1, 2$, with $K = 10$ and $M = 1, 3, 5$.

In the first simulation represented in Figure 7, we analyze the performance of the RLS-CKD algorithm with that of the classical RLS algorithm. The echo path changes after 4 s of simulation by changing the impulse response $\mathbf{h}_2$ with a random impulse response of the same length, with samples between 0 and 0.5. In the first part of the plot, we can remark that the RLS-CKD algorithm achieves a convergence rate similar to that of the classical RLS algorithm. Regarding the tracking capability, when the echo path changes, the RLS-CKD algorithm outperforms the RLS algorithm. The RLS-CKD achieves a normalized misalignment of -30 dB in less than 200 ms. We can remark that the constant value M only affects the normalized misalignment level when the echo path changes.

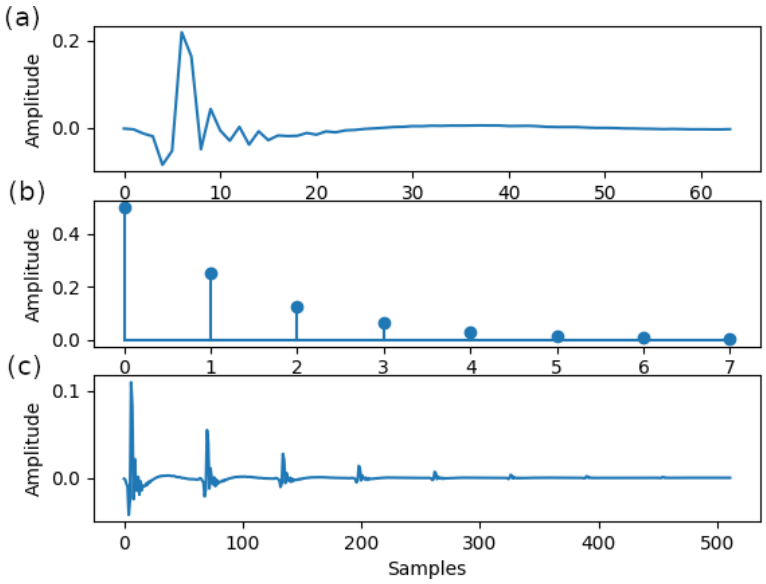

Figure 6. Impulse responses for the bilinear setup: (**a**) $\mathbf{h}_1$, first impulse from the G168 Recommendation [22]; (**b**) $\mathbf{h}_2$, exponential generated impulse response; and (**c**) impulse response of the target system, $\mathbf{h} = \mathbf{h}_2 \otimes \mathbf{h}_1$.

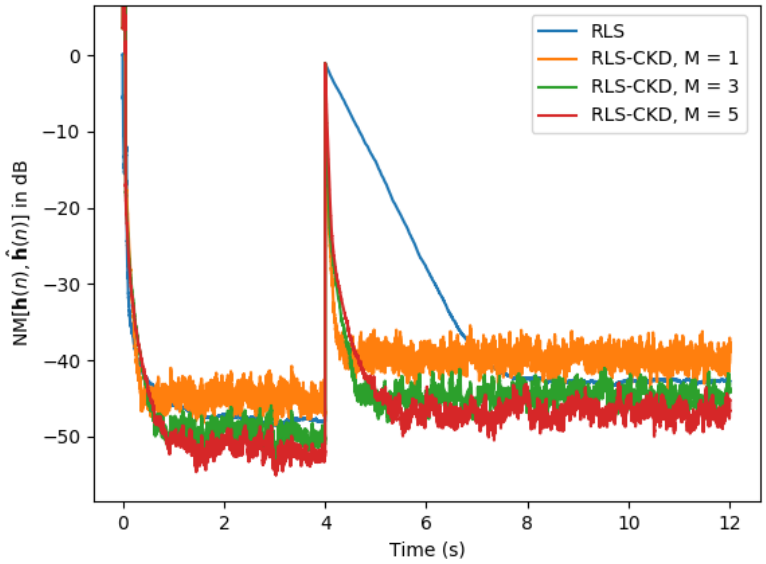

Figure 7. Normalized misalignment of the classical RLS ($L = 512$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$) algorithms. The input signal is white Gaussian noise and the impulse response changes after 4 s of simulation.

Next, in Figure 8, we analyze the behavior of the RLS-CKD algorithm in a scenario where the input signal is an AR(1) process. The echo path changes in the same manner as in the previous scenario. In this case, the RLS-CKD algorithm achieves an even lower normalized misalignment of almost 10 dB (e.g., when $M = 5$) compared to the RLS algorithm. When the echo path changes, the values of M do not impact the RLC-CKD algorithm too much. This time, the RLS-CKD achieves a normalized misalignment of -40 dB in less than 500 ms, while the RLS algorithm requires at least 3 s to achieve a comparable level.

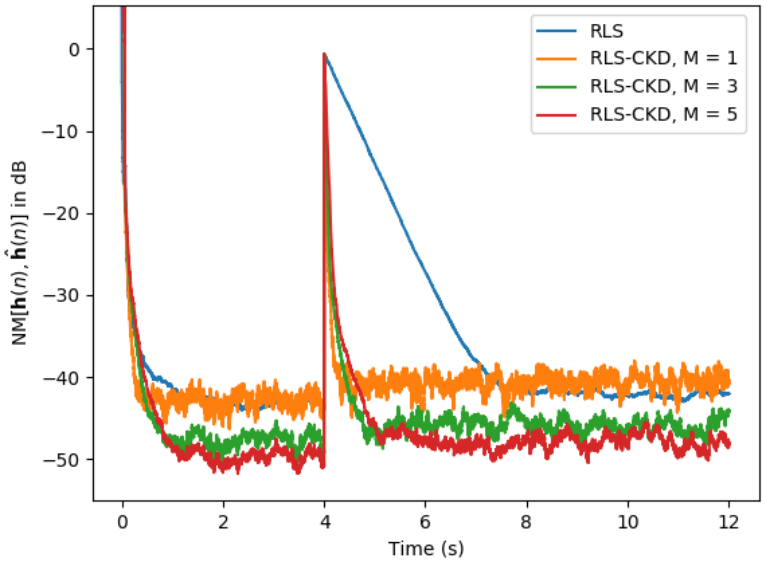

Figure 8. Normalized misalignment of the classical RLS ($L = 512$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$) algorithms. The input signal is an AR(1) process and the impulse response changes after 4 s of simulation.

We conclude the first set of simulations with the scenario depicted in Figure 9, where the input signal is a speech sequence and the echo path changes in the middle of the simulation in the same manner.

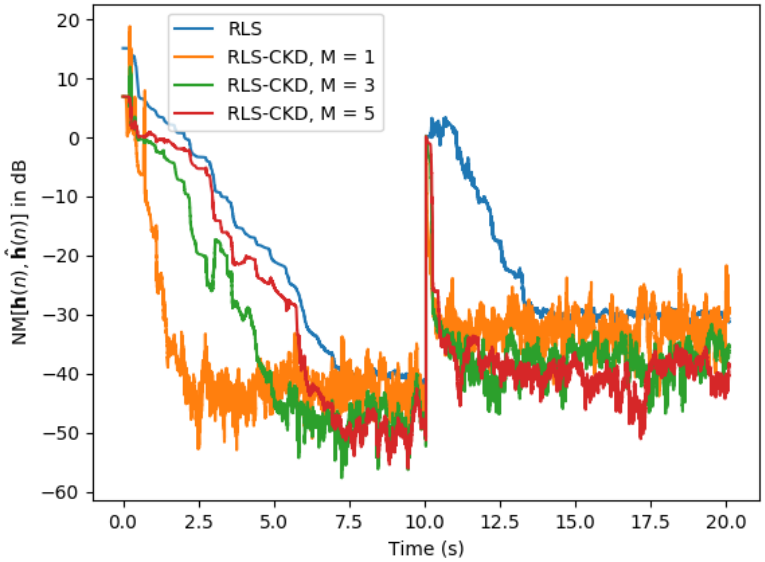

Figure 9. Normalized misalignment of the classical RLS ($L = 512$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$) algorithms. The input signal is a speech sequence and the impulse response changes in the middle of the simulation.

As we can notice in Figure 9, the steady-state misalignment of the conventional RLS algorithm (the blue curve) is similar to the misalignment of the RLS-CKD algorithm using $M = 1$, while the initial convergence rate and tracking of the proposed algorithm are much better. A larger value of M influences only the initial convergence rate of the RLS-CKD algorithm, but keeps the same fast tracking reaction. On the other hand, the steady-state misalignment of the RLS-CKD is improving for a larger value of M (i.e., for larger values of the forgetting factors λ_i, closer to 1).

Furthermore, we continue the simulations with the trilinear approach, based on the impulse responses from Figure 10. In this case, we have considered an even longer echo

path of thousands of coefficients. The echo path of the system that must be identified is obtained as $\mathbf{h} = \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1$, of size $L = L_1 L_2 L_3 = 2048$, with $\mathbf{h}_1$ ($L_1 = 64$) from Figure 6a, $\mathbf{h}_2$ ($L_2 = 8$) from Figure 10a, and $\mathbf{h}_3$ ($L_3 = 4$) from Figure 10b. The second impulse response (i.e., $\mathbf{h}_2$) is randomly generated, with samples between 0 and 0.5, while the third impulse response (i.e., $\mathbf{h}_3$) is obtained as $h_{3l_3} = 0.5^{l_3 - 1}$, with $l_3 = 1, 2, \ldots, L_3$, where $L_3 = 4$.

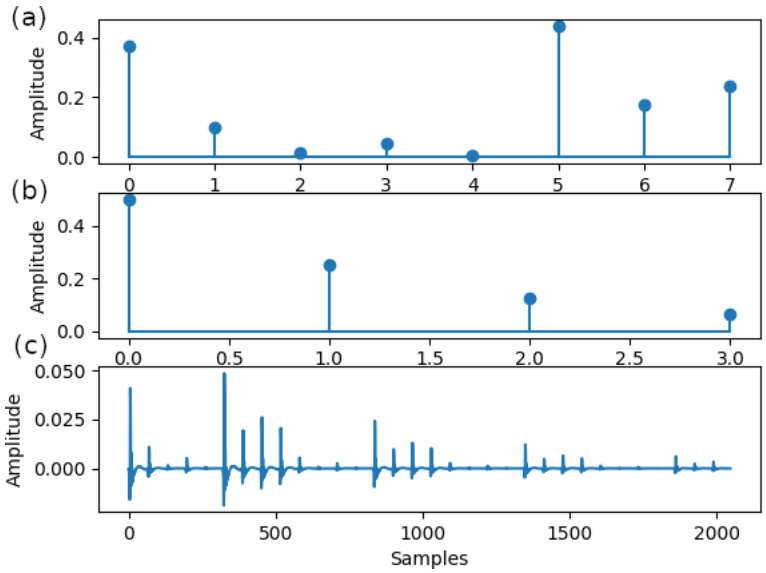

Figure 10. Impulse responses for the trilinear setup: (**a**) $\mathbf{h}_2$, random generated impulse response; (**b**) $\mathbf{h}_3$, exponential generated impulse response; and (**c**) Impulse response of the target system, $\mathbf{h} = \mathbf{h}_3 \otimes \mathbf{h}_2 \otimes \mathbf{h}_1$.

In Figure 11, the first simulation in the trilinear scenario is represented. The input signal is a white Gaussian noise and the echo path changes by generating $\mathbf{h}_3$ as a random impulse response after 4 s, so this impacts the whole system. It is worth noting that the RLS-CKD algorithm presents a slightly faster converge rate compared to that of the RLS algorithm and a lower normalized misalignment for $M = 5$ of at least 10 dB. In terms of tracking, the RLS-CKD algorithm succeeds in re-estimating the new echo path and we can see that the smaller the forgetting factor is (i.e., $M = 1$), the faster the tracking.

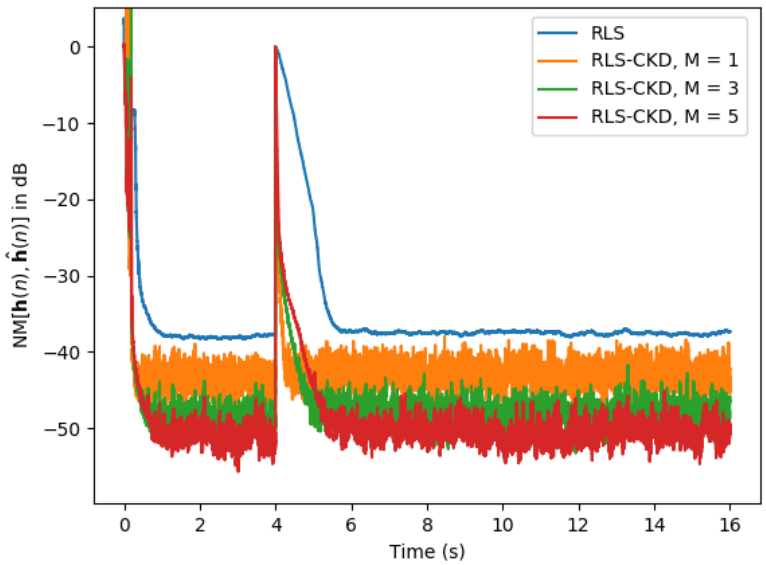

Figure 11. Normalized misalignment of the classical RLS ($L = 2048$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$, $L_3 = 4$) algorithms. The input signal is white Gaussian noise and the impulse response changes after 4 s.

In the scenario represented in Figure 12, the input signal is an AR(1) process and the echo path changes by regenerating $\mathbf{h}_3$ after 4 s. The RLS-CKD algorithm outperforms the classical RLS algorithm in terms of convergence rate, normalized misalignment, and tracking capability, with a much lower computational complexity. Finally, in Figure 13, we conclude the set of simulations with a scenario where the input signal is a speech sequence. Again, the echo path changes by regenerating $\mathbf{h}_3$ after 6 s of simulation. While the classical RLS algorithm requires more than 3 s to achieve a reasonable normalized misalignment level, the RLS-CKD algorithm succeeds at estimating the target system, presenting a good tracking capability even when the echo path changes. However, for a faster convergence rate, the RLS-CKD algorithm requires a much lower forgetting factor (e.g., $M = 1$). Also, the steady-state misalignment of the conventional RLS algorithm (after the change of the system) is similar to the misalignment of the RLS-CKD algorithm using $M = 1$.

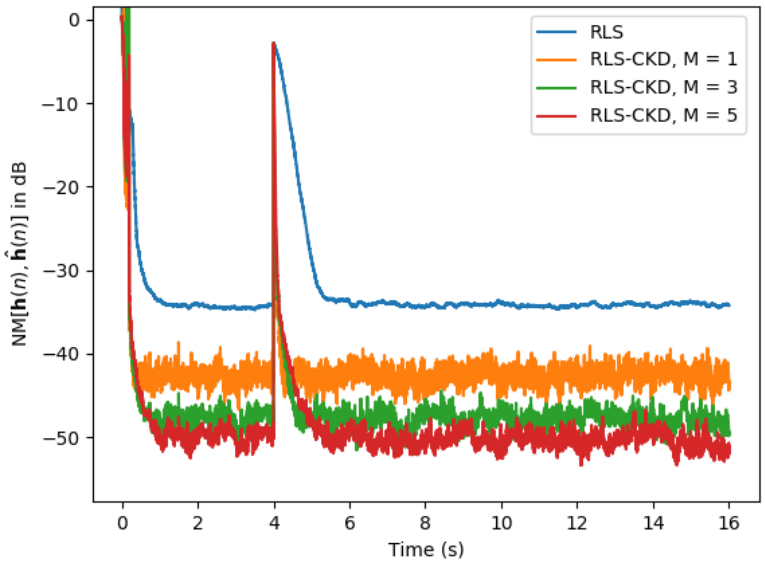

Figure 12. Normalized misalignment of the classical RLS ($L = 2048$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$, $L_3 = 4$) algorithms. The input signal is an AR(1) process and the impulse response changes after 4 s.

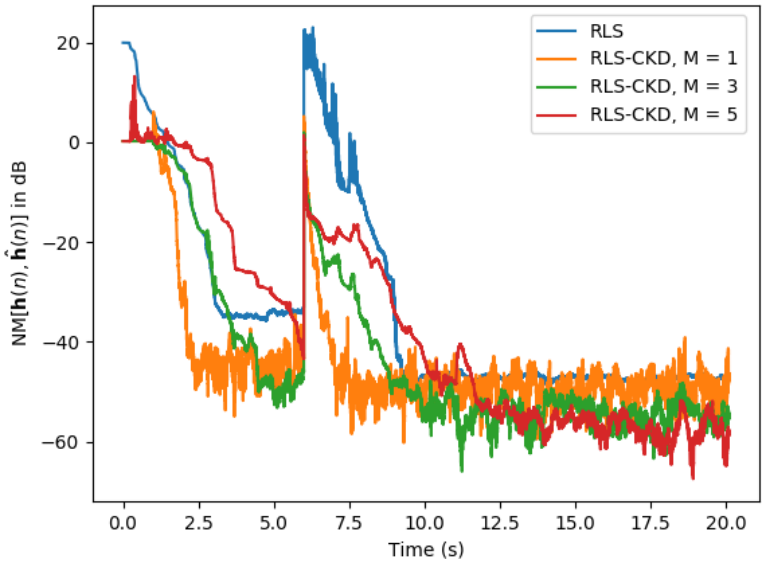

Figure 13. Normalized misalignment of the classical RLS ($L = 2048$) and RLS-CKD ($L_1 = 64$, $L_2 = 8$, $L_3 = 4$) algorithms. The input signal is a speech sequence and the impulse response changes after 6 s.

6. Conclusions

In this paper, we have introduced various memoryless and memory structures described by a bilinear input-output relation. Based on this approach, we have obtained a SISO system from a MISO system, which is a cascade of shorter length filters. We then developed the multilinear RLS algorithm considering the Kronecker product decomposition and outlining the reduction in terms of computational complexity. Finally, we have presented a set of simulations as a comparison between the newly developed RLS-CKD algorithm and the classical RLS algorithm. Simulations proved that the RLS-CKD algorithm outperforms the classical RLS algorithm in terms of convergence rate, normalized misalignment, and tracking capability. We can conclude that the RLS-CKD algorithm is a good candidate for real-time applications, which implies long length impulse responses and systems characterized by reverberation.

Author Contributions: Conceptualization, A.-G.R.; Formal analysis, S.C.; Software, C.P.; Methodology, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a grant of the Romanian Ministry of Education and Research, CNCS-UEFISCDI, project number: PN-III-P1-1.1-TE-2019-0420, within PNCDI III.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Haykin, S. *Adaptive Filter Theory*, 4th ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2002.
2. Benesty, J.; Huang, Y. (Eds.) *Adaptive Signal Processing–Applications to Real-World Problems*; Springer: Berlin/Heidelberg, Germany, 2003.
3. Gay, S.L.; Benesty, J. (Eds.) *Acoustic Signal Processing for Telecommunication*; Kluwer Academic Publisher: Boston, MA, USA, 2000.
4. Benesty, J.; Gaensler, T.; Morgan, D.R.; Sondhi, M.M.; Gay, S.L. *Advances in Network and Acoustic Echo Cancellation*; Springer: Berlin/Heidelberg, Germany, 2001.
5. Tsoulos, I.G.; Stavrou, V.; Mastorakis, N.E.; Tsalikakis, D. GenConstraint: A programming tool for constraint optimization problems. *SoftwareX* **2019**, *10*. [CrossRef]
6. Stavrou, V.N.; Tsoulos, I.G.; Mastorakis, N.E. Transformations for FIR and IIR Filters' Design. *Symmetry* **2021**, *13*, 533. [CrossRef]
7. Duttweiler, D.L. Proportionate normalized least-mean-squares adaptation in echo cancelers. *IEEE Trans. Speech Audio Process.* **2000**, *8*, 508–518. [CrossRef]
8. Benesty, J.; Gay, S.L. An improved PNLMS algorithm. In Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, Orlando, FL, USA, 13–17 May 2002; pp. II-1881–II-1884.
9. Deng, H.; Doroslovački, M. Proportionate adaptive algorithms for network echo cancellation. *IEEE Trans. Signal Process.* **2006**, *54*, 1794–1803. [CrossRef]
10. Loganathan, P.; Khong, A.W.; Naylor, P. A class of sparseness-controlled algorithms for echo cancellation. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *17*, 1591–1601. [CrossRef]
11. Paleologu, C.; Benesty, J.; Ciochină, S. *Sparse Adaptive Filters for Echo Cancellation*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2010.
12. Yang, Z.; Zheng, Y.R.; Grant, S.L. Proportionate affine projection sign algorithms for network echo cancellation. *IEEE Trans. Audio Speech Lang. Process.* **2011**, *19*, 2273–2284. [CrossRef]
13. Liu, J.; Grant, S.L. Proportionate adaptive filtering for block-sparse system identification. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 623–630. [CrossRef]
14. Rusu, A.-G.; Ciochină, S. Cascaded adaptive filters in a bilinear approach for system identification. In Proceedings of the 2020 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 5–6 November 2020; pp. 1–4.
15. Loan, C.F.V. The ubiquitous Kronecker product. *J. Comput. Appl. Math.* **2000**, *123*, 85–100. [CrossRef]
16. Benesty, J.; Cohen, I.; Chen, J. *Array Processing–Kronecker Product Beamforming*; Springer: Cham, Switzerland, 2019.
17. 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]
18. Paleologu, C.; Benesty, J.; Ciochină, S. Adaptive filtering for the identification of bilinear forms. *Digital Signal Process.* **2018**, *75*, 153–167. [CrossRef]
19. Dogariu, L.-M.; Stanciu, C.L.; Elisei-Iliescu, C.; Paleologu, C.; Benesty, J.; Ciochină, S. Tensor-based adaptive filtering algorithms. *Symmetry* **2021**, *13*, 481. [CrossRef]
20. Bertsekas, D.P. *Nonlinear Programming*, 2nd ed.; Athena Scientific: Belmont, MA, USA, 1999.

21. Benesty, J.; Paleologu, C.; Ciochină, S. Regularization of the RLS algorithm. *IEICE Trans. Fundam.* **2011**, *E94-A*, 1628–1629. [CrossRef]
22. *Digital Network Echo Cancellers*; ITU-T Recommendation G.168; ITU: Geneva, Switzerland, 2002.