A PROJECTION APPROACH TO ADAPTIVE IFIR FILTERING

Sander Wahls and Holger Boche

TU München, Lehrstuhl für Theoretische Informationstechnik, Theresienstr. 90, 80333 München, Germany
{sander.wahls,boche}@tum.de

ABSTRACT

Current adaptive interpolated finite impulse response (IFIR) filtering algorithms update the interpolation filter and the model filter separately while in each case the other filter is fixed. In contrast, we modify the standard least mean squares (LMS) algorithm such that after each iteration the adapted filter is projected back into the set of IFIR filters. This can be considered a joint update of the interpolation filter and the model filter. We also propose a simplified version of our new algorithm. With K and D denoting the lengths of the model and the interpolation filter, respectively, the complexity of our new simplified projected LMS algorithm is only $\mathcal{O}(K+D)$ flops per iteration. Normalized variants of the new algorithms are derived as well. The performance of the new simplified normalized algorithm is compared in two numerical examples with a current state-of-the-art adaptive IFIR filtering algorithm. We find that our new algorithm converges faster. At the same time, our algorithm does not require experimental tuning of the step size.

Index Terms— Adaptive filters, Interpolated finite impulse response filter, Least mean square algorithms, Projection algorithms

I. INTRODUCTION

Interpolated finite impulse response (IFIR) filters are filters whose impulse responses $\mathbf{h} \in \mathbb{R}^{L(K-1)+D}$ can be written as

$$\mathbf{h} = \mathbf{T_{Sg}} \mathbf{f} = \mathbf{T_f} \mathbf{Sg},$$

where $\mathbf{g} \in \mathbb{R}^K$ and $\mathbf{f} \in \mathbb{R}^D$. Furthermore, with $\mathbf{e_k}$ denoting the kth column of the $K \times K$ identity matrix $\mathbf{I_K}$,

$$\mathbf{S} = \left[\begin{array}{ccccccc} \mathbf{e_1} & \mathbf{0_{K\times(L-1)}} & \cdots & \mathbf{e_{K-1}} & \mathbf{0_{K\times(L-1)}} & \mathbf{e_K} \end{array} \right]^T$$

is a factor L upsampling matrix, and

$$\begin{aligned} \mathbf{T_{Sg}} &= [\mathbf{Sg}]_{i-j+1} \in \mathbb{R}^{(L(K-1)+D)\times D} \text{ and} \\ \mathbf{T_f} &= [\mathbf{f}]_{i-j+1} \in \mathbb{R}^{(L(K-1)+D)\times(L(K-1)+1)} \end{aligned}$$

are the convolution (Toeplitz) matrices associated with the impulse responses $\mathbf{Sg}$ and $\mathbf{f}$.[1] We only consider real filters for simplicity. Any IFIR filter $\mathbf{h} = \mathbf{T_f Sg}$ can be implemented as the convolution with the model filter $\mathbf{g}$, followed by an upsampling by a factor of L, and the convolution with the interpolation filter $\mathbf{f}$. Hence, convolution with an IFIR filter of length $L(K-1)+D$ can be implemented with a complexity of only $\mathcal{O}(K+D)$ floating point operations (flops) per time slot, instead of $\mathcal{O}(LK+D)$ for a standard finite impulse response (FIR) filter of length $L(K-1)+D$ [1]. IFIR filters are popular for the cheap implementation of filters with long but smooth impulse responses because then the approximation loss that results from the restricted structure is usually low. Typical application areas are acoustic echo cancellation [2] and channel equalization [3], [4]. The design of IFIR filters however is more complicated than in the standard unconstrained FIR case. In this paper, we discuss adaptive IFIR filter design where the IFIR filter is iteratively adapted to match some error criterion. Recent works on this topic include [5], [6], [7]. In all these works, implicitly some kind of alternating optimization is performed. The model and interpolation filters are updated separately, where the model filter is updated under the assumption that the interpolation filter is fixed and vice versa. In contrast, we propose a new projection approach. Instead of separate updates for the model and the interpolation filter, we first perform a step of the standard least mean squares (LMS) algorithm without the IFIR constraint and then project the result back into the set of IFIR filters. This can be considered a joint update of the model and interpolation filters.

This work has been supported by the DFG under grant BO 1734/5-2.

[1] We denote the element in ith row and jth column of any matrix $\mathbf{A}$ by $[\mathbf{A}]_{i,j}$. For any $\mathbf{v} \in \mathbb{R}^M$, we set $[\mathbf{v}]_k := [\mathbf{v}]_{k,1} := 0$ if $k \notin \{1,\ldots,M\}$.

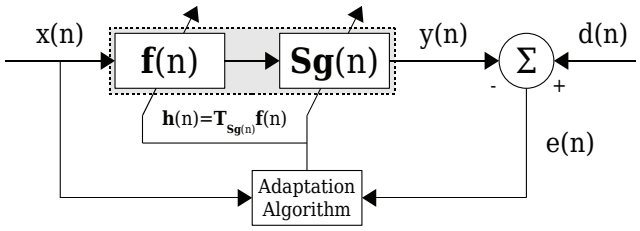

Fig. 1. Adaptive IFIR filtering / $x(n)$: input; $y(n)$: actual output; $d(n)$: desired output; $\mathbf{f}(n), \mathbf{g}(n)$: FIR filters ; $\mathbf{S}$: upsampling matrix

We proceed as follows. First, projection onto the set of IFIR filters is discussed (Sec. II). Then, we use these results in order to derive a new projected LMS algorithm (Sec. III). A computationally cheaper simplified variant of the algorithm is also established and the new algorithms are compared with current approaches in terms of complexity. Then, we investigate the performance of our new algorithms (Sec. IV). Finally, the paper is concluded (Sec. V).

II. PROJECTION ONTO THE SET OF IFIR FILTERS

Let $\bar{\mathbf{h}} \in \mathbb{R}^{L(K-1)+D}$ contain the impulse response of an FIR filter. Then, we are interested in finding the best IFIR approximation of $\bar{\mathbf{h}}$ in the least squares sense. That is, find $\mathbf{P_{LS}}(\bar{\mathbf{h}}) \in \operatorname{argmin}_{\mathbf{h}=\mathbf{T_f Sg}} \|\bar{\mathbf{h}} - \mathbf{h}\|^2$. Lin and Vaidyanathan have proposed the following iterative approach to this problem [8]. Choose some initial non-zero $\mathbf{f}(0) \in \mathbb{R}^D$ (e.g., randomly), and iterate

$$\left\{ \begin{array}{lcl} \mathbf{g}(n+1) &=& (\mathbf{T}_{\mathbf{f}(n)}\mathbf{S})^\dagger \bar{\mathbf{h}} \\ \mathbf{f}(n+1) &=& \mathbf{T}^\dagger_{\mathbf{Sg}(n+1)} \bar{\mathbf{h}} \\ \mathbf{h}(n+1) &=& \mathbf{T}_{\mathbf{Sg}(n+1)} \mathbf{f}(n+1) \end{array} \right. . \quad (1)$$

Here, $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse. We note that it is currently unclear whether the iteration really converges towards some projection $\mathbf{P_{LS}}(\bar{\mathbf{h}})$, or whether the approach is suboptimal.

III. ADAPTIVE IFIR FILTERING

The adaptive IFIR filtering problem is depicted in Fig. 1. Let

$$\mathbf{x}(n) = \left[\begin{array}{cccc} x(n) & \dots & x(n-L(K-1)-D+1) \end{array}\right]^T$$

denote the vector containing the past $L(K-1)+D$ inputs. Then, the current output of the IFIR filter $\mathbf{h}(n) = \mathbf{T}_{\mathbf{f}(n)}\mathbf{S}\mathbf{g}(n)$ is $y(n) = \mathbf{h}(n)^T\mathbf{x}(n)$. The current error becomes

$$e(n) = d(n) - y(n) = d(n) - \mathbf{h}(n)^T\mathbf{x}(n). \tag{2}$$

Our goal is to adapt the model filter $\mathbf{g}(n)$ and the interpolation filter $\mathbf{f}(n)$ such that this error becomes small.

III-A. Review of the Standard LMS Algorithm

Let us shortly discuss the standard least mean squares (LMS) algorithm without IFIR constraint. This corresponds to Fig. 1 if we assume that the gray box can be any FIR filter $\mathbf{h}$. The LMS algorithm tries to minimize the expected square error $J^{(n)}_{\text{LMS}}(\mathbf{h}) := \mathbb{E}[|d(n) - \mathbf{h}^T\mathbf{x}(n)|^2]$ under the assumption that the $d(n)$ are zero-mean random variables and the $\mathbf{x}(n)$ are zero-mean random vectors with covariance $\mathbf{R}_{\mathbf{x}}(n) := \mathbb{E}[\mathbf{x}(n)\mathbf{x}(n)^T]$. The basic idea is to perform stochastic gradient descent. That is, we iterate

$$\begin{aligned} \mathbf{h}(n+1) &= \mathbf{h}(n) - \frac{\mu}{2}\nabla J^{(n)}_{\text{LMS}}[\mathbf{h}(n)] \\ &= \mathbf{h}(n) + \mu[\mathbf{R}_{d\mathbf{x}^T}(n) - \mathbf{R}_{\mathbf{x}}(n)\mathbf{h}(n)], \end{aligned}$$

where $\mu > 0$ is the step size and $\mathbf{R}_{d\mathbf{x}^T}(n) := \mathbb{E}[d(n)\mathbf{x}(n)]$. Using the instantaneous approximations $\mathbf{R}_{\mathbf{x}}(n) \approx \mathbf{x}(n)\mathbf{x}(n)^T$ and $\mathbf{R}_{d\mathbf{x}^T}(n) \approx d(n)\mathbf{x}(n)$, we obtain the classical LMS algorithm:

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu e(n)\mathbf{x}(n). \tag{3}$$

The initial guess can be chosen arbitrarily, but the usual choice is $\mathbf{h}(0) = \mathbf{0}$. We refer to [9, Ch. 10] for details on the LMS algorithm.

Finally, a word on the step size. Usually, a small fixed value is chosen as step size. It has to be small enough to ensure stability of the algorithm but it should be as large as possible in order to ensure fast convergence. Let us define the a posteriori error

$$r(n) := d(n) - \mathbf{h}(n+1)^T\mathbf{x}(n).$$

Then, the LMS algorithm is stable in the sense that the absolute a priori error $|e(n)|$ is not larger than the absolute a posteriori error $|r(n)|$ if and only if $\mu \le 2/\|\mathbf{x}(n)\|^2 = 2/\mathbf{x}(n)^T\mathbf{x}(n)$ for all n. (Check that $|r(n)| = |1 - \mu\|\mathbf{x}(n)\|^2||e(n)|$ in order to see this [9, Ch. 10.4].) Motivated by this observation, one may replace the constant step size μ in the LMS algorithm with a variable step size of the form $\mu(n) = \bar{\mu}/(\epsilon^2 + \|\mathbf{x}(n)\|^2)$, where $0 < \bar{\mu} \le 2$ and $\epsilon > 0$ is a small constant that stabilizes the iteration. The resulting algorithm is known as the *normalized LMS (NLMS)* algorithm.

III-B. Projected LMS Algorithm

The impulse responses found by the LMS algorithm are not IFIR, in general. One approach to overcome this problem is to project the filters found by the LMS algorithm onto the set of IFIR filters. Hence, we integrate an iteration of the approximate projection from Sec. II. The *projected LMS (P-LMS)* algorithm reads:

$$\boxed{\left\{\begin{array}{lcl} \bar{\mathbf{h}}(n+1) &=& \mathbf{h}(n) + \mu e(n)\mathbf{x}(n) \\ \mathbf{g}(n+1) &=& (\mathbf{T}_{\mathbf{f}(n)}\mathbf{S})^\dagger\bar{\mathbf{h}}(n+1) \\ \mathbf{f}(n+1) &=& \mathbf{T}^\dagger_{\mathbf{S}\mathbf{g}(n+1)}\bar{\mathbf{h}}(n+1) \\ \mathbf{h}(n+1) &=& \mathbf{T}_{\mathbf{S}\mathbf{g}(n+1)}\mathbf{f}(n+1) \end{array}\right.} \tag{4}$$

Note that we have to initialize $\mathbf{h}(0)$ and $\mathbf{f}(0)$. We may choose $\mathbf{h}(0) = \mathbf{0}$, but $\mathbf{f}(0)$ should be non-zero in order to avoid a break down of the algorithm. For example, choose $\mathbf{f}(0)$ at random. As before, we obtain a *normalized P-LMS (P-NLMS)* algorithm if we replace the constant step size μ in (4) with the variable step size $\mu(n) = \bar{\mu}/(\epsilon^2 + \|\mathbf{x}(n)\|^2)$. We note that this scaling means no significant additional complexity if we use the update rule

$$\|\mathbf{x}(n+1)\|^2 = \|\mathbf{x}(n)\|^2 + x(n+1)^2 - x(n-KD)^2. \tag{5}$$

III-C. Simplified Projected LMS Algorithm

Next, we derive a cheaper variant of the P-LMS algorithm (4).

Approximation of the Pseudoinverses: We start with the expensive pseudoinverses in (4). The $\mathbf{g}(n+1)$ given in (4) can be approximated as the solution of the regularized normal equation

$$\underbrace{\left(\delta^2\mathbf{I} + \mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{T}_{\mathbf{f}(n)}\mathbf{S}\right)}_{=:\mathbf{\Lambda}(n)}\mathbf{g}(n+1) = \mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\bar{\mathbf{h}}(n+1),$$

where δ is a small regularization constant [10, Sec. 5.1.1]. Thus,

$$\begin{aligned} \mathbf{g}(n+1) &= \mathbf{\Lambda}(n)^{-1}\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\bar{\mathbf{h}}(n+1) \\ &= \mathbf{\Lambda}(n)^{-1}\left(\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{h}(n) + \mu e(n)\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n)\right) \\ &= \mathbf{\Lambda}(n)^{-1}\Big(\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{T}_{\mathbf{f}(n)}\mathbf{S}\mathbf{g}(n) \\ &\qquad + \mu e(n)\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n)\Big) \\ &\approx \mathbf{g}(n) + \mu e(n)\mathbf{\Lambda}(n)^{-1}\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n). \end{aligned}$$

Here, we have used that $\mathbf{h}(n) = \mathbf{T}_{\mathbf{f}(n)}\mathbf{S}\mathbf{g}(n)$. We approximate

$$\begin{aligned} \mathbf{\Lambda}(n)^{-1} &\approx \left[\mathbf{diag}\left(\delta^2\mathbf{I} + \mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{T}_{\mathbf{f}(n)}\mathbf{S}\right)\right]^{-1} \\ &= (\delta^2 + \|\mathbf{f}(n)\|^2)^{-1}\mathbf{I}, \end{aligned}$$

and finally obtain

$$\mathbf{g}(n+1) \approx \mathbf{g}(n) + \frac{\mu}{\delta^2 + \|\mathbf{f}(n)\|^2}e(n)\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n). \tag{6}$$

Similar reasoning gives the following approximation for $\mathbf{f}(n+1)$ if we additionally can assume that $\mathbf{g}(n+1)$ is very close to $\mathbf{g}(n)$:

$$\mathbf{f}(n+1) \approx \mathbf{f}(n) + \frac{\mu}{\delta^2 + \|\mathbf{g}(n)\|^2}e(n)\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n). \tag{7}$$

Approximation of the Matrix-Vector Products: Next, we will use an improved version of the approximation technique described in [7, (40)] in order to get rid of the matrix-vector products $\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n)$ and $\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n)$.[2] With $\mathbf{w}(-1) := \mathbf{0_{L\times 1}}$ and $\mathbf{v}(-1) := \mathbf{0_{LK\times 1}}$, we can introduce the following two sequences:

$$\mathbf{w}(n) := \left[\begin{array}{c} \frac{1}{\|\mathbf{g}(n)\|}\mathbf{g}(n)^T\mathbf{S}^T\mathbf{x}(n) \\ [\mathbf{w}(n-1)]_1 \\ \vdots \\ [\mathbf{w}(n-1)]_{L-1} \end{array}\right], \tag{8}$$

[2]The same technique is also used implicitly in [5], [6], and, according to [7], also [11]. (The authors do not have access to [11].) Our variant is improved by an additional normalization that neutralizes variations of the norms in the $\mathbf{g}(n)$ and $\mathbf{f}(n)$ as n increases. Compare Sec. III-D.

$$\mathbf{v}(n) \quad := \quad \begin{bmatrix} \frac{1}{\|\mathbf{f}(n)\|} \begin{bmatrix} \mathbf{f}(n)^T & \mathbf{0} \end{bmatrix} \mathbf{x}(n) \\ [\mathbf{v}(n-1)]_1 \\ \vdots \\ [\mathbf{v}(n-1)]_{LK-1} \end{bmatrix}. \tag{9}$$

If the filters $\mathbf{g}(n)$ and $\mathbf{f}(n)$ are slowly varying in time, we have

$$\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n) = \begin{bmatrix} \mathbf{g}(n)^T\mathbf{S}^T\mathbf{x}(n) \\ \mathbf{g}(n)^T\mathbf{S}^T\mathbf{x}(n-1) \\ \vdots \\ \mathbf{g}(n)^T\mathbf{S}^T\mathbf{x}(n-D+1) \end{bmatrix} \approx \|\mathbf{g}(n)\|\mathbf{w}(n), \tag{10}$$

and, similarly,

$$\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n) \approx \|\mathbf{f}(n)\|\mathbf{S}^T\mathbf{v}(n). \tag{11}$$

Application to the P-LMS Algorithm : With the previous results, we see that the error $e(n)$ in (2) is approximately equal to

$$\tilde{e}(n) := d(n) - \|\mathbf{g}(n)\|\mathbf{w}(n)^T\mathbf{f}(n). \tag{12}$$

The *simplified P-LMS (SP-LMS)* algorithm results if we approximate the terms $\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n)$, $\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n)$, and $e(n)$ in (6) and (7) as described in (10), (11), and (12), and consider the result exact:

$$\boxed{\begin{cases} \mathbf{g}(n+1) &= \mathbf{g}(n) + \frac{\|\mathbf{f}(n)\|}{\delta^2+\|\mathbf{f}(n)\|^2}\mu\tilde{e}(n)\mathbf{S}^T\mathbf{v}(n) \\ \mathbf{f}(n+1) &= \mathbf{f}(n) + \frac{\|\mathbf{g}(n)\|}{\delta^2+\|\mathbf{g}(n)\|^2}\mu\tilde{e}(n)\mathbf{w}(n) \end{cases}.} \tag{13}$$

The initial values $\mathbf{g}(0)$ and $\mathbf{f}(0)$ have to be non-zero. They can be chosen at random. Finally, we obtain the *simplified P-NLMS (SP-NLMS) algorithm* if we apply the same approximations that were used in the derivation of SP-LMS to the NLMS algorithm:

$$\boxed{\begin{cases} \mathbf{g}(n+1) &= \mathbf{g}(n) + \frac{\|\mathbf{f}(n)\|}{\delta^2+\|\mathbf{f}(n)\|^2}\frac{\bar{\mu}\tilde{e}(n)}{\epsilon^2+\|\mathbf{x}(n)\|^2}\mathbf{S}^T\mathbf{v}(n) \\ \mathbf{f}(n+1) &= \mathbf{f}(n) + \frac{\|\mathbf{g}(n)\|}{\delta^2+\|\mathbf{g}(n)\|^2}\frac{\bar{\mu}\tilde{e}(n)}{\epsilon^2+\|\mathbf{x}(n)\|^2}\mathbf{w}(n) \end{cases}} \tag{14}$$

III-D. Prior Approaches

Various authors have proposed alternative adaptive IFIR filtering algorithms [11], [5], [6], [7]. We follow the exposition in [7]. We can run two separate LMS-like algorithms that minimize the two expressions $e(n) = d(n) - \mathbf{g}(n)^T\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n)$ and $e(n) = d(n) - \mathbf{f}(n)^T\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n)$ for the error independently:

$$\begin{cases} \mathbf{g}(n+1) &= \mathbf{g}(n) + \mu_1 e(n)\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n) \\ \mathbf{f}(n+1) &= \mathbf{f}(n) + \mu_2 e(n)\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n) \end{cases}. \tag{15}$$

This is the so-called doubly adaptive IFIR filtering (DAIFIR) algorithm given in [7, Eqns. (37)+(38)]. The initial guesses $\mathbf{g}(0)$ and $\mathbf{f}(0)$ have to be non-zero. The matrix-vector products in (15) make the algorithm computationally expensive compared to the standard LMS algorithm (3). Therefore, one may try to approximate the products. The approximation technique used in the previous subsection is an improved version of [7, (40)]. Instead of the vectors $\mathbf{w}(n)$ and $\mathbf{v}(n)$, they consider the unnormalized variants $\hat{\mathbf{w}}(n)$ and $\hat{\mathbf{v}}(n)$ where the divisions by $\|\mathbf{g}(n)\|^2$ and $\|\mathbf{f}(n)\|^2$ in (8) and (9) are omitted. Then, the matrix-vector products are approximated as $\mathbf{T}^T_{\mathbf{S}\mathbf{g}(n)}\mathbf{x}(n) \approx \hat{\mathbf{w}}(n)$ and $\mathbf{S}^T\mathbf{T}^T_{\mathbf{f}(n)}\mathbf{x}(n) \approx \hat{\mathbf{v}}(n)$, and the error becomes $e(n) \approx d(n) - \hat{\mathbf{w}}(n)^T\mathbf{f}(n) =: \hat{e}(n)$. Insertion of these approximations into the DAIFIR algorithm (15) results in a cheaper simplified DAIFIR (S-DAIFIR) algorithm [7, Eqns. (42)+(43)]:

$$\begin{cases} \mathbf{g}(n+1) &= \mathbf{g}(n) + \mu_1\hat{e}(n)\mathbf{S}^T\hat{\mathbf{v}}(n) \\ \mathbf{f}(n+1) &= \mathbf{f}(n) + \mu_2\hat{e}(n)\hat{\mathbf{w}}(n) \end{cases}. \tag{16}$$

The initial values $\mathbf{g}(0)$ and $\mathbf{f}(0)$ have to be non-zero. The S-DAIFIR algorithm (16) has initially been proposed by various authors for constant step sizes μ_1 and μ_2 as well as for normalized step sizes $\mu_1(n) = \bar{\mu}_1/(\epsilon^2 + \|\mathbf{S}^T\hat{\mathbf{v}}(n)\|^2)$, $\mu_2(n) = \bar{\mu}_2/(\epsilon^2 + \|\hat{\mathbf{w}}(n)\|^2)$ [11], [5], [6], [7]. The normalized variant is sometimes called the normalized S-DAIFIR (SN-DAIFIR) algorithm. A difficult question is how the step sizes should be selected, even for SN-DAIFIR. Numerical experiments have shown that asymmetric step sizes ($\bar{\mu}_1 \neq \bar{\mu}_2$) can give superior performance [6, Sec. IV]. However, experimental optimization of asymmetric step sizes is a very tedious process. Analytical expressions for the optimal step sizes of SN-DAIFIR similar to the ones for NLMS (see, e.g., [9, Ch. 25.1]) seem to be unknown. Consequently, Batista et al. have proposed to use symmetric step sizes ($\bar{\mu}_1 = \bar{\mu}_2$) so that experimental step size selection becomes manageable [7, Sec. 3.3]. We note that as the gradient vectors in (14) and (16) are almost the same, our new SN-PLMS algorithm can also be interpreted as an analytical method to select asymmetric step sizes.

III-E. Comparison of Computational Complexities

We only compare the LMS with the simplified algorithms, as they are the only $\mathcal{O}(K+D)$ algorithms. The approximated total complexities (in flops per iteration) of the various algorithms are:[3]

LMS (3), NLMS	$4LK + 4(D-L)$
SP-LMS (13), SP-NLMS (14)	$6K + 8D$
S-DAIFIR (16), SN-DAIFIR	$4K + 6D$

The SP-LMS algorithm requires about $3/(2L)$ times the flops per iteration of the standard LMS algorithm, but $3/2$ times the flops per iteration of the S-DAIFIR algorithm (assuming $K \gg D \geq L$). The normalized versions of the algorithms have approximately the same costs because the values of $\|\mathbf{x}(n)\|^2$, $\|\hat{\mathbf{w}}(n)\|^2$, and $\|\mathbf{S}^T\hat{\mathbf{v}}(n)\|^2$ can be updated in $\mathcal{O}(1)$ using (5) or similar formulas, respectively.

IV. NUMERICAL EXAMPLES

First Example: We consider a system identification problem. The impulse responses $\bar{\mathbf{h}}$ of the to be identified systems are random combinations of a few low-frequency cosines combined with an exponential power loss and a normalization.[4] A couple of typical impulse responses is depicted in Fig. 2 (bottom). The configuration was $K = 46$, $L = 3$, and $D = 4$. The inputs $x(1), \ldots, x(8000)$ of the adaptive filter were chosen as normally distributed random variables. The desired outputs of the adaptive filter are a noisy version of the systems outputs that correspond to these inputs. That is, $d(n) = \bar{\mathbf{h}}^T\mathbf{x}(n) + 10^{-1}\eta(n)$ for all $n = 1, \ldots, 8000$, where the $\eta(1), \ldots, \eta(8000)$ again denote normally distributed random variables (the noise). Our optimality

[3] We assess M flops for the addition of two $M \times 1$ vectors and a cost of $2M$ flops for the corresponding inner product. The multiplication of a $N \times M$ matrix and an $M \times 1$ vector is assessed with $2MN$ flops. The product of a scalar with an $M \times 1$ vector is assessed with M flops. Purely scalar and other operations are ignored. The structure of $\mathbf{S}$ is accounted for.

[4] Consult the source code of the examples for details: http://goo.gl/573z9.

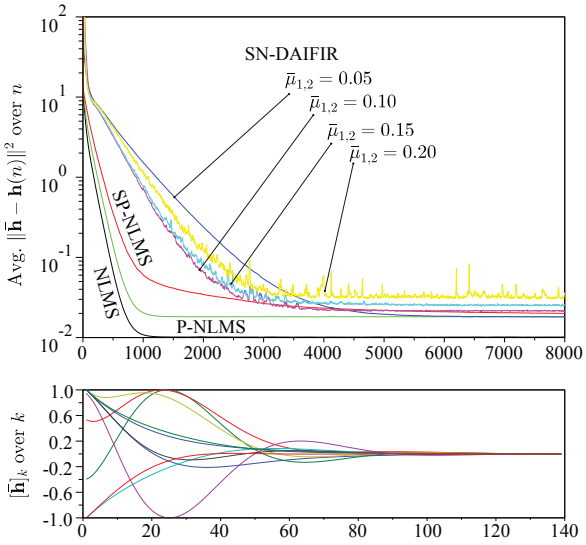

Fig. 2. First Example: Smooth Random Impulse Responses

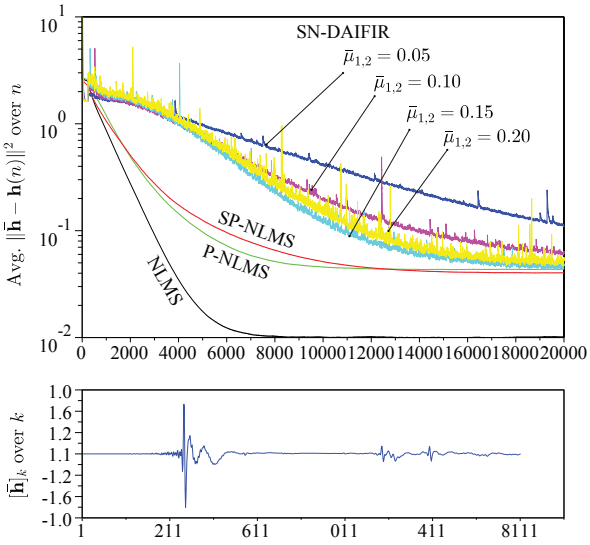

Fig. 3. Second Example: Real-World Room Impulse Response

criterion is the system identification error $\|\bar{\mathbf{h}}-\mathbf{h}(n)\|^2$, where $\mathbf{h}(n)$ denotes the approximation found by the respective adaptive filtering algorithm in the nth iteration. We evaluated the evolution of the identification error for the NLMS algorithm ($\bar{\mu} = 1$), our new P-NLMS and SP-NLMS algorithms (both $\bar{\mu} = 1$), and the SN-DAIFIR algorithm of [6] (we try various $\bar{\mu}_1 = \bar{\mu}_2$; cf. Sec. III-D). Our focus however lies on the two cheap algorithms SP-NLMS and SN-DAIFIR [both $\mathcal{O}(K + D)$]. The regularization constants were $\delta = \epsilon = 10^{-3}$, and we have averaged over 10000 runs. The results are shown in Fig. 2 (top). Our new SP-NLMS algorithm reduces the error much faster than the SN-DAIFIR algorithm for any of the tested step sizes $\bar{\mu}_1 = \bar{\mu}_2$. Step size tuning was not necessary for SP-NLMS. We simply used the same step size as for NLMS.

Second Example: We want to identify a part of a real-world room impulse response from the measurements described in [12]. Specifically, we consider the samples 800 to 1800 of the impulse response 30x10y.wav measured in the classroom (omnidirectional). See Fig. 3 (bottom). The general setup was the same as in the first example, with a configuration of $K = 500$, $L = 2$, and $D = 3$. We considered 20000 time slots, and averaged over 100 runs. The simulation results are shown in Fig. 3 (top). The same observations as in the previous example can be made. Our new filter reduces the error faster, and step size tuning has not been necessary.

V. CONCLUSION

We have presented new adaptive IFIR filtering algorithms based on an approximate projection into the set of IFIR filters. In particular, we presented two new low complexity algorithms, SP-LMS and SP-NLMS, with an complexity of only $\mathcal{O}(K + D)$ flops per iteration. We have evaluated the performance of our new SP-NLMS algorithm in numerical examples, where it has performed significantly better than SN-DAIFIR with symmetric step sizes.

VI. REFERENCES

[1] Y. Neuvo, C.-Y. Dong, and S. Mitra, "Interpolated finite impulse response filters," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, no. 3, pp. 563–570, 1984.

[2] A. Abousaada, T. Aboulnasr, and W. Steenaart, "An echo tail canceller based on adaptive interpolated FIR filtering," *IEEE Trans. Circuit Syst. II*, vol. 39, no. 7, pp. 409–416, 1992.

[3] R. C. de Lamare and R. Sampaio-Neto, "Adaptive interference suppression for DS-CDMA systems based on interpolated FIR filters with adaptive interpolators in multipath channels," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 2457–2474, 2007.

[4] C. Meng and J. Tuqan, "High-performance low-cost DFE using IFIR filters," in *Proc. Asilomar Conf. Sig. Syst. Comp.*, Pacific Grove, CA, Nov. 2007.

[5] R. C. Bilcu, P. Kuosmanen, and K. Egiazarian, "On adaptive interpolated FIR filters," in *Proc. IEEE ICASSP*, Montreal, Canada, May 2004.

[6] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank MMSE filtering with interpolated FIR filters and adaptive interpolators," *IEEE Signal Process. Lett.*, vol. 12, no. 3, pp. 177–180, 2005.

[7] E. L. O. Batista, O. J. Tobias, and R. Seara, "New insights in adaptive cascaded FIR structure: Application to fully adaptive interpolated FIR structures," in *Proc. Eur. Signal Process. Conf. (EUSIPCO)*, Poznan, Poland, Sept. 2007.

[8] Y.-P. Lin and P. P. Vaidyanathan, "An iterative approach to the design of IFIR matched filters," in *Proc. IEEE ISCAS*, Hong Kong, 1997.

[9] A. H. Sayed, *Adaptive Filters*, Wiley, Hoboken, NJ, 2008.

[10] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects*, SIAM, Philadelphia, PA, 1998.

[11] M. D. Grosen, *New FIR Structures for fixed and adaptive digital filters*, PhD thesis, UC Santa Barbara, CA, 1987.

[12] R. Stewart and M. Sandler, "Database of omnidirectional and B-format room impulse responses," in *Proc. IEEE ICASSP*, Dallas, TX, Mar. 2010.