Low-Complexity Implementation of the Improved Multiband-Structured Subband Adaptive Filter Algorithm

Feiran Yang, *Member, IEEE*, Ming Wu, *Member, IEEE*, Peifeng Ji, *Member, IEEE*, and Jun Yang, *Senior Member, IEEE*

***Abstract*—Previously, we proposed an improved multiband-structured subband adaptive filter (IMSAF) algorithm to accelerate the convergence rate of the MSAF algorithm. When the projection order and/or the number of subbands is increased, the convergence rate of the IMSAF algorithm improves at the cost of increased complexity. Thus, this paper proposes several approaches to reduce the complexity of the IMSAF algorithm, both in error vector calculation and matrix inversion operation. Specifically, three approaches are developed to efficiently calculate error vector. The first approach gives an approximate filtering, whereas the other two approaches can provide a fast exact filtering with or without update of the weight vector explicitly based on a recursive scheme. The decorrelation property of IMSAF is determined, and two simplified variants are developed to reduce the complexity as by-products, i.e., the simplified IMSAF (SIMSAF) and pseudo IMSAF algorithms. Then, we discuss the problem of solving a linear system of equations. The performance advantages, limitations, and preferable applications of various methods are analyzed and discussed. Computer simulations are conducted in the context of system identification to determine the principle and efficiency of the proposed fast algorithms.**

***Index Terms*—Adaptive filtering, affine projection, decorrelation, linear system of equations, low complexity.**

I. Introduction

ADAPTIVE filters [1] are widely used in communication, active noise control (ANC), and acoustic echo cancellation (AEC), among others. In particular, subband adaptive filters (SAF) [2] are commonly employed to improve the convergence behavior. Unfortunately, the convergence rate of the traditional SAF is reduced due to aliasing and band-edge effects [2]. To address this problem, a multiband weight-control mechanism was developed in [3]–[6] where the fullband weight vector is updated by subband signals. We also proposed an improved multiband-structured subband adaptive filter (IMSAF) algorithm recently [7] to better handle the colored signal and long impulse response. Interestingly, this algorithm can be regarded as a generalized form of the normalized least-mean-square (NLMS), affine projection (AP) [8], and multiband-structured subband adaptive filter (MSAF) algorithms [6].

Numerous fast adaptive algorithms have been reported in the literature. The recursive least squares (RLS) algorithm displays good convergence performance; however, the fast implementation of this algorithm remains too complex for long adaptive filters. For example, the stabilized fast transversal filter (FTF) algorithm [9] has a complexity of $O(8M)$ (M being the filter length), but this algorithm suffers from numerical instability. The lattice filters have improved numerical behavior and stability but they generally require $O(20M)$ operations per sample [1]. The fast AP (FAP) algorithm [10], [11] requires $2M + 20P$ operations per sample (P being projection order), and many variants of this algorithm were presented [12]–[20]. The complexity of FAP is considerably lower than that of RLS-type algorithms; however, its convergence is still slow for the colored input signals. The complexity of the fast IMSAF algorithms proposed in this manuscript is only slightly higher than that of FAP but is significantly lower than that of RLS-type algorithms. Furthermore, the IMSAF algorithm displays a faster convergence rate than the FAP and NLMS algorithms do, as per the current study findings. The fast IMSAF algorithms also exhibit good numerical stability. Thus, the IMSAF algorithm is appealing for application in various fields given its favorable properties.

The computational complexity of the IMSAF algorithm mainly comes from three operations: i) error vector calculation, ii) weight vector update, and iii) matrix inversion operation. We attempt to present complete solutions to these problems through this work. The main contributions of this study are organized as follows:

1) Three fast filtering approaches are proposed to compute error vector for different applications. The first one gives an approximate filtering, whereas the second and third approaches can calculate the error vector precisely. These ideas can also be generalized to a family of adaptive filters with the similar filtering and update structures.
2) We present the decorrelation property of the IMSAF algorithm. This property provides new insight into the IMSAF algorithm, i.e., the IMSAF employs two prewhitening techniques to accelerate the convergence.

Manuscript received July 05, 2014; revised February 16, 2015 and June 02, 2015; accepted June 10, 2015. Date of publication June 25, 2015; date of current version August 27, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. John McAllister. This work was supported by Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06040501 and also partially supported by National Natural Science Fund of China under Grants 11174317 and 11304349.

The authors are with the State Key Laboratory of Acoustics and the Key Laboratory of Noise and Vibration Research, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China (e-mail: feirany.ioa@gmail.com; mingwu@mail.ioa.ac.cn; jipeifeng@mail.ioa.ac.cn; jyang@mail.ioa.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2015.2450198

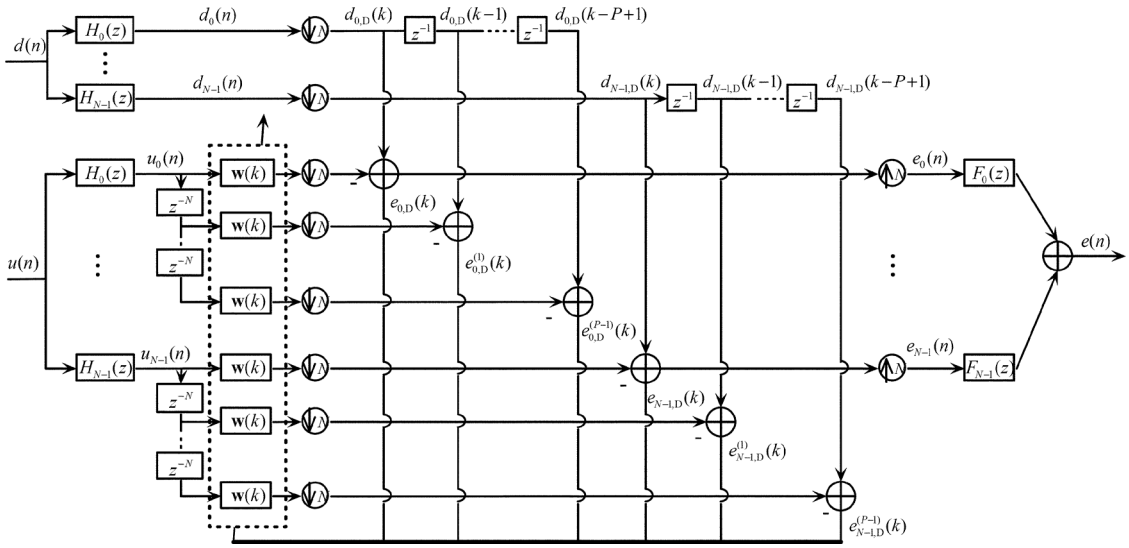


Fig. 1. Block diagram of the IMSAF algorithm.

3) Two IMSAF variants, namely, the simplified IMSAF (SIMSAF) and pseudo IMSAF algorithms, are presented as by-products of the decorrelation property. The SIMSAF algorithm converts the operation of a single matrix inversion with size $NP \times NP$ into the calculation of N inversions of the matrices with size $P \times P$ (N and P being the number of subbands and projection order, respectively). This procedure reduces the complexity considerably. Meanwhile, the pseudo IMSAF algorithm resembles a projection type gradient algorithm like NLMS with the complexity $O(2M)$.

Notations: Throughout this paper, we use uppercase and lowercase bold fonts to denote matrices and vectors, respectively, e.g., $\mathbf{R}$ and $\mathbf{r}$. The elements of the matrix and vector are denoted as $[\mathbf{R}]_{i,j}$ and $[\mathbf{r}]_i$. A pth column of $\mathbf{R}$ is denoted as $\mathbf{R}^{(p)}$. Superscript T denotes the transpose operator. $\mathbf{I}$ and $\mathbf{0}$ are the identity and null matrices of appropriate size that can be deduced easily from the text.

II. Review of the IMSAF Algorithm

We consider data $d(n)$ arising from the linear model

$$d(n) = \mathbf{w}_{\mathrm{o}}^T \mathbf{u}(n) + v(n) \tag{1}$$

where $\mathbf{w}_{\mathrm{o}} = [w_0, w_1, \ldots, w_{M-1}]^T$ is the weight vector of an unknown system with length M, $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$ denotes the input signal vector, and $v(n)$ is the system noise. The derivations in this manuscript are based on the time-shift structure in the input regression vectors. Nevertheless, we will highlight which relations are structure independent and which ones are not, which can be used to develop the fast algorithm for arbitrary regression vectors [21]–[24].

Fig. 1 shows the block diagram of the IMSAF algorithm. The desired signal $d(n)$ and input signal $u(n)$ are partitioned into N subband signals, $d_i(n)$ and $u_i(n)$, by means of analysis filters $H_i(z) = \sum_{j=0}^{L-1} h_i(j) z^{-j}$. The subband input signals $u_i(n)$ are filtered by the adaptive filter to generate the subband output signals $y_i(n)$. The subband signals $d_i(n)$ and $y_i(n)$ are then decimated by a factor N to generate $d_{i,\mathrm{D}}(k)$ and $y_{i,\mathrm{D}}(k)$. Notations $\downarrow N$ and $\uparrow N$ represent N-fold decimation and interpolation. The fullband error signal $e(n)$ is finally obtained by interpolating and recombining all the subband error signals $e_{i,\mathrm{D}}(k)$ using a synthesis filter bank $F_i(z) = \sum_{j=0}^{L-1} f_i(j) z^{-j}$. Variables n and k are used to indicate the original and decimated sequences. $\mathbf{u}_i(k) = [u_i(kN), u_i(kN-1), \ldots, u_i(kN-M+1)]^T$ denotes the ith subband reference signal vector, and $\mathbf{w}(k) = [w_0(k), w_1(k), \ldots, w_{M-1}(k)]^T$ is the fullband weight vector of the adaptive filter.

To describe the IMSAF algorithm, we define the input signal matrix $\mathbf{U}(\mathbf{k})$, the desired signal vector $\mathbf{d}_{\mathrm{D}}(k)$, the estimated signal vector $\mathbf{y}_{\mathrm{D}}(k)$, the *a priori* error signal vector $\mathbf{e}_{\mathrm{D}}(k)$, and the *a posteriori* error signal vector $\boldsymbol{\xi}_{\mathrm{D}}(k)$ as follows

$$\begin{aligned}
\mathbf{U}(k) &= [\mathbf{U}_0(k), \mathbf{U}_1(k), \ldots, \mathbf{U}_{N-1}(k)], && (2)\\
\mathbf{d}_{\mathrm{D}}(k) &= [\mathbf{d}_{0,\mathrm{D}}^T(k), \mathbf{d}_{1,\mathrm{D}}^T(k), \ldots, \mathbf{d}_{N-1,\mathrm{D}}^T(k)]^T, && (3)\\
\mathbf{y}_{\mathrm{D}}(k) &= [\mathbf{y}_{0,\mathrm{D}}^T(k), \mathbf{y}_{1,\mathrm{D}}^T(k), \ldots, \mathbf{y}_{N-1,\mathrm{D}}^T(k)]^T \\
&= \mathbf{U}^T(k)\mathbf{w}(k), && (4)\\
\mathbf{e}_{\mathrm{D}}(k) &= [\mathbf{e}_{0,\mathrm{D}}^T(k), \mathbf{e}_{1,\mathrm{D}}^T(k), \ldots, \mathbf{e}_{N-1,\mathrm{D}}^T(k)]^T \\
&= \mathbf{d}_{\mathrm{D}}(k) - \mathbf{y}_{\mathrm{D}}(k), && (5)\\
\boldsymbol{\xi}_{\mathrm{D}}(k) &= [\boldsymbol{\xi}_{0,\mathrm{D}}^T(k), \boldsymbol{\xi}_{1,\mathrm{D}}^T(k), \ldots, \boldsymbol{\xi}_{N-1,\mathrm{D}}^T(k)]^T \\
&= \mathbf{d}_{\mathrm{D}}(k) - \mathbf{U}^T(k)\mathbf{w}(k+1) && (6)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{U}_i(k) &= [\mathbf{u}_i(k), \mathbf{u}_i(k-1), \ldots, \mathbf{u}_i(k-P+1)], && (7)\\
\mathbf{d}_{i,\mathrm{D}}(k) &= [d_{i,\mathrm{D}}(k), d_{i,\mathrm{D}}(k-1), \ldots, d_{i,\mathrm{D}}(k-P+1)]^T, && (8)\\
\mathbf{y}_{i,\mathrm{D}}(k) &= [y_{i,\mathrm{D}}^{(0)}(k), y_{i,\mathrm{D}}^{(1)}(k), \ldots, y_{i,\mathrm{D}}^{(P-1)}(k)]^T \\
&= \mathbf{U}_i^T(k)\mathbf{w}(k), && (9)\\
\mathbf{e}_{i,\mathrm{D}}(k) &= [e_{i,\mathrm{D}}(k), e_{i,\mathrm{D}}^{(1)}(k), \ldots, e_{i,\mathrm{D}}^{(P-1)}(k)]^T \\
&= \mathbf{d}_{i,\mathrm{D}}(k) - \mathbf{y}_{i,\mathrm{D}}(k), && (10)\\
\boldsymbol{\xi}_{i,\mathrm{D}}(k) &= \mathbf{d}_{i,\mathrm{D}}(k) - \mathbf{U}_i^T(k)\mathbf{w}(k+1). && (11)
\end{aligned}$$

TABLE I
IMSAF ALGORITHM

Equation	$\times$	$+$
$\mathbf{y}_{\mathrm{D}}(k)=\mathbf{U}^T(k)\mathbf{w}(k)$	PM	$P(M-1)$
$\mathbf{e}_{\mathrm{D}}(k)=\mathbf{d}_{\mathrm{D}}(k)-\mathbf{y}_{\mathrm{D}}(k)$	$-$	P
$\mathbf{R}(k)=\mathbf{U}^T(k)\mathbf{U}(k)$	$2N^2P$	$2N^2P$
$\boldsymbol{\varepsilon}_{\mathrm{D}}(k)=\mu[\mathbf{R}(k)+\delta\mathbf{I}]^{-1}\mathbf{e}_{\mathrm{D}}(k)$	P_{m1}	P_{a1}
$\mathbf{w}(k+1)=\mathbf{w}(k)+\mathbf{U}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k)$	PM	PM
Band partitioning and synthesizing		
$u_i(n)=\mathbf{h}_i^T\mathbf{u}_L(n)$	NL	$N(L-1)$
$d_{i,\mathrm{D}}(k)=\mathbf{h}_i^T\mathbf{d}(k)$	L	$L-1$
$e(n)=\sum_{i=0}^{N-1}\mathbf{f}_i^T\mathbf{e}_i(n)$	L	$L-1$
Variables $\mathbf{u}_L(n)=[u(n),u(n-1),...,u(n-L+1)]^T$ $\mathbf{d}(k)=[d(kN),d(kN-1),...,d(kN-L+1)]^T$ $\mathbf{e}_i(n)=[e_i(n),e_i(n-1),...,e_i(n-L+1)]^T$ $\mathbf{h}_i=[h_i(0),h_i(1),...,h_i(L-1)]^T$ - analysis filter $\mathbf{f}_i=[f_i(0),f_i(1),...,f_i(L-1)]^T$ - synthesis filter		
Parameters M - length of adaptive filter N - number of subbands P - projection order L - length of analysis and synthesis filters μ - step size δ - regularization parameter		
Total Complexity: $2PM+2N^2P+P_{m1}+(N+2)L$ multiplications $2PM+2N^2P+P_{a1}+(N+2)(L-1)$ additions		

We now seek $\mathbf{w}(k+1)$ by solving the constrained optimization criterion:

$$\begin{aligned}&\min_{\mathbf{w}(k+1)}\|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2\\&\text{s.t.}\quad\boldsymbol{\xi}_{\mathrm{D}}(k)=\left[\mathbf{I}-\mu\mathbf{U}^T(k)\mathbf{U}(k)\left(\mathbf{U}^T(k)\mathbf{U}(k)+\delta\mathbf{I}\right)^{-1}\right]\\&\qquad\qquad\times\mathbf{e}_{\mathrm{D}}(k).\end{aligned}\tag{12}$$

The solution of this optimization problem is [7]

$$\mathbf{w}(k+1)=\mathbf{w}(k)+\mu\mathbf{U}(k)[\mathbf{U}^T(k)\mathbf{U}(k)+\delta\mathbf{I}]^{-1}\mathbf{e}_{\mathrm{D}}(k)\tag{13}$$

where μ is the step size and δ is the regularization parameter. The weight vector update is performed for each N input samples.

Table I describes the computational complexity of the IMSAF algorithm in terms of the number of multiplications and additions per sample. The parts related to signal decomposition and synthesis are presented in Table I but are omitted in the subsequent tables for simplicity. We assume that solving the linear system of equations $[\mathbf{R}(k)+\delta\mathbf{I}]\boldsymbol{\varepsilon}_{\mathrm{D}}(k)=\mu\mathbf{e}_{\mathrm{D}}(k)$ requires P_{m1} multiplications and P_{a1} additions per sample, where $\mathbf{R}(k)=\mathbf{U}^T(k)\mathbf{U}(k)$.

As per Table I, the complexity of the IMSAF algorithm involves three operations: i) calculating the filtering output signal vector $\mathbf{y}_{\mathrm{D}}(k)$, ii) updating the weight vector obtained with (13), and iii) solving the linear system of equations. We present efficient solutions to these three problems in the following sections.

III. Efficient Computation of Weight Vector

Direct update of the weight vector according to (13) requires PM operations per sample. Given the special structure of (13), the weight vector can be expressed using an auxiliary coefficient vector.

The vector $\boldsymbol{\varepsilon}_{\mathrm{D}}(k)$ can be divided into $\boldsymbol{\varepsilon}_{\mathrm{D}}(k)=[\boldsymbol{\varepsilon}_{0,\mathrm{D}}^T(k),\boldsymbol{\varepsilon}_{1,\mathrm{D}}^T(k),\ldots,\boldsymbol{\varepsilon}_{N-1,\mathrm{D}}^T(k)]^T$, where $\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k)$ is a vector of size $P\times 1$. Expanding the matrix/vector multiplications in (13), $\mathbf{w}(k+1)$ can be rewritten as

$$\begin{aligned}\mathbf{w}(k+1)&=\mathbf{w}(k)+\mathbf{U}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k)\\&=\mathbf{w}(k)+\sum_{i=0}^{N-1}\mathbf{U}_i(k)\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k).\end{aligned}\tag{14}$$

Continuing to recursively expand (14) yields

$$\begin{aligned}&\mathbf{w}(k+1)\\&\quad=\mathbf{w}(0)+\sum_{i=0}^{N-1}\sum_{j=0}^{k}\mathbf{U}_i(k-j)\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-j)\\&\quad=\mathbf{w}(0)+\sum_{i=0}^{N-1}\sum_{j=0}^{k}\sum_{m=0}^{P-1}\mathbf{u}_i(k-j-m)[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-j)]_m.\end{aligned}\tag{15}$$

Assuming that $\mathbf{u}_i(k)=\mathbf{0}$ for $k<0$, dividing the summation pair in (15) into two groups yields

$$\begin{aligned}\mathbf{w}(k+1)&=\mathbf{w}(0)\\&\quad+\sum_{i=0}^{N-1}\sum_{j=0}^{P-1}\mathbf{u}_i(k-j)\sum_{m=0}^{j}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-j+m)]_m\\&\quad+\sum_{i=0}^{N-1}\sum_{j=P}^{k}\mathbf{u}_i(k-j)\sum_{m=0}^{P-1}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-j+m)]_m.\end{aligned}\tag{16}$$

Using (16), $\mathbf{w}(k+1)$ can be alternatively expressed as

$$\mathbf{w}(k+1)=\hat{\mathbf{w}}(k)+\sum_{i=0}^{N-1}\mathbf{U}_i(k)\boldsymbol{\varphi}_i(k)\tag{17}$$

where

$$\hat{\mathbf{w}}(k)=\mathbf{w}(0)+\sum_{i=0}^{N-1}\sum_{j=P}^{k}\mathbf{u}_i(k-j)\sum_{m=0}^{P-1}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-j+m)]_m\tag{18}$$

$$\boldsymbol{\varphi}_i(k)=\begin{bmatrix}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k)]_0\\\sum_{m=0}^{1}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-1+m)]_m\\\cdots\\\sum_{m=0}^{P-1}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-P+1+m)]_m\end{bmatrix}.\tag{19}$$

Interestingly, both $\hat{\mathbf{w}}(k)$ and $\boldsymbol{\varphi}_i(k)$ can be updated recursively. From (18), it is noted that $\hat{\mathbf{w}}(k)$ can be computed in a recursive way:

$$\begin{aligned}\hat{\mathbf{w}}(k+1)&=\hat{\mathbf{w}}(k)+\sum_{i=0}^{N-1}\mathbf{u}_i(k-P+1)\\&\quad\times\sum_{m=0}^{P-1}[\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k-P+1+m)]_m\\&=\hat{\mathbf{w}}(k)+\sum_{i=0}^{N-1}\mathbf{u}_i(k-P+1)[\boldsymbol{\varphi}_i(k)]_{P-1}.\end{aligned}\tag{20}$$

It is seen from (19) that $\boldsymbol{\varphi}_i(k)$ can also be calculated recursively

$$\boldsymbol{\varphi}_i(k) = \boldsymbol{\varepsilon}_{i,\mathrm{D}}(k) + \begin{bmatrix} 0 \\ \bar{\boldsymbol{\varphi}}_i(k-1) \end{bmatrix} \tag{21}$$

where $\bar{\boldsymbol{\varphi}}_i(k-1)$ is a vector that consists the uppermost $P-1$ elements of $\boldsymbol{\varphi}_i(k)$.

The derivation of fast adaptation strategy is not limited to tapped-delay-line data structure, i.e., the arguments in this section hold for arbitrary regression vectors.

IV. Fast Filtering Approaches

In this section, we present three schemes to significantly reduce the complexity of the filtering step.

A. Fast Approximate Filtering

Since the structure of the IMSAF algorithm is similar to that of the AP algorithm, our first attempt towards the complexity reduction of the IMSAF algorithm is to apply the idea in FAP [10], [11] to the IMSAF algorithm and derive an efficient method for the error vector calculation. This fast approximate filtering approach is named as "FAF."

Taking (10) and (11) into account, one has

$$\mathbf{e}_{i,\mathrm{D}}(k) = \begin{bmatrix} d_{i,\mathrm{D}}(k) - \mathbf{u}_i^T(k)\mathbf{w}(k) \\ \bar{\boldsymbol{\xi}}_{i,\mathrm{D}}(k-1) \end{bmatrix} \tag{22}$$

where $\bar{\boldsymbol{\xi}}_{i,\mathrm{D}}(k-1)$ consists of the $P-1$ upper elements of $\boldsymbol{\xi}_{i,\mathrm{D}}(k-1)$. Using (6), one has

$$\boldsymbol{\xi}_{\mathrm{D}}(k-1) = \mathbf{d}_{\mathrm{D}}(k-1) - \mathbf{U}^T(k-1)\mathbf{w}(k). \tag{23}$$

Using (4) and (5), $\mathbf{d}_{\mathrm{D}}(k-1)$ can be rewritten as

$$\mathbf{d}_{\mathrm{D}}(k-1) = \mathbf{e}_{\mathrm{D}}(k-1) + \mathbf{U}^T(k-1)\mathbf{w}(k-1). \tag{24}$$

By substituting (24) into (23) and using (13), we get[1]

$$\begin{aligned} \boldsymbol{\xi}_{\mathrm{D}}(k-1) &= \mathbf{e}_{\mathrm{D}}(k-1) - \mathbf{U}^T(k-1)[\mathbf{w}(k) - \mathbf{w}(k-1)] \\ &= \mathbf{I} - \mu\mathbf{U}^T(k-1)\mathbf{U}(k-1) \\ &\quad \times [\mathbf{U}^T(k-1)\mathbf{U}(k-1) + \delta\mathbf{I}]^{-1}\mathbf{e}_{\mathrm{D}}(k-1). \end{aligned} \tag{25}$$

As long as δ is significantly smaller than the eigenvalues of $\mathbf{U}_i^T(k-1)\mathbf{U}_i(k-1)$, the following approximation can be derived:

$$\mathbf{U}^T(k-1)\mathbf{U}(k-1)[\mathbf{U}^T(k-1)\mathbf{U}(k-1) + \delta\mathbf{I}]^{-1} \approx \mathbf{I}. \tag{26}$$

Substituting (26) into (25), $\mathbf{e}_{\mathrm{D}}(k-1)$ and $\boldsymbol{\xi}_{\mathrm{D}}(k-1)$ can be related as follows

$$\boldsymbol{\xi}_{\mathrm{D}}(k-1) = (1-\mu)\mathbf{e}_{\mathrm{D}}(k-1). \tag{27}$$

Substituting (27) into (22) yields

$$\mathbf{e}_{i,\mathrm{D}}(k) = \begin{bmatrix} d_{i,\mathrm{D}}(k) - \mathbf{u}_i^T(k)\mathbf{w}(k) \\ (1-\mu)\bar{\mathbf{e}}_{i,\mathrm{D}}(k-1) \end{bmatrix} \tag{28}$$

where $\bar{\mathbf{e}}_{i,\mathrm{D}}(k-1)$ consists of the $P-1$ upper elements of $\mathbf{e}_{i,\mathrm{D}}(k-1)$.

[1]It can also be directly obtained from the constraint condition in (12).

The first element of $\mathbf{e}_{i,\mathrm{D}}(k)$ can be computed by applying the auxiliary coefficient vector strategy discussed in Section III. Using (17), we obtain

$$\begin{aligned} e_{i,\mathrm{D}}(k) &= d_{i,\mathrm{D}}(k) - \mathbf{u}_i^T(k)\hat{\mathbf{w}}(k-1) \\ &\quad - \sum_{j=0}^{N-1} \boldsymbol{\chi}_{i,j}^T(k)\boldsymbol{\varphi}_j(k-1) \end{aligned} \tag{29}$$

where $\boldsymbol{\chi}_{i,j}(k) = \mathbf{U}_j^T(k-1)\mathbf{u}_i(k)$. We define the correlation element

$$\rho_{i,j}^{(m)}(k) = \mathbf{u}_i^T(k)\mathbf{u}_j(k-m). \tag{30}$$

By employing (30), we can write $\boldsymbol{\chi}_{i,j}(k)$ as

$$\begin{aligned} \boldsymbol{\chi}_{i,j}(k) &= [\mathbf{u}_i^T(k)\mathbf{u}_j(k-1), \mathbf{u}_i^T(k)\mathbf{u}_j(k-2), \\ &\quad \ldots, \mathbf{u}_i^T(k)\mathbf{u}_j(k-P)]^T \\ &= [\rho_{i,j}^{(1)}(k), \rho_{i,j}^{(2)}(k), \ldots, \rho_{i,j}^{(P)}(k)]^T. \end{aligned} \tag{31}$$

The matrix $\mathbf{R}(k)$ can be written in the following block matrix form

$$\begin{aligned} \mathbf{R}(k) &= \mathbf{U}^T(k)\mathbf{U}(k) \\ &= \begin{bmatrix} \mathbf{R}_{0,0}(k) & \mathbf{R}_{0,1}(k) & \ldots & \mathbf{R}_{0,N-1}(k) \\ \mathbf{R}_{1,0}(k) & \mathbf{R}_{1,1}(k) & \ldots & \mathbf{R}_{1,N-1}(k) \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{R}_{N-1,0}(k) & \mathbf{R}_{N-1,1}(k) & \ldots & \mathbf{R}_{N-1,N-1}(k) \end{bmatrix} \end{aligned} \tag{32}$$

where $\mathbf{R}_{i,j}(k) = \mathbf{U}_i^T(k)\mathbf{U}_j(k)$. Using (30), we can express $\mathbf{R}_{i,j}(k)$ as

$$\begin{aligned} &\mathbf{R}_{i,j}(k) \\ &= \begin{bmatrix} \rho_{i,j}^{(0)}(k) & \rho_{i,j}^{(1)}(k) & \ldots & \rho_{i,j}^{(P-1)}(k) \\ \rho_{j,i}^{(1)}(k) & \rho_{i,j}^{(0)}(k-1) & \ldots & \rho_{i,j}^{(P-2)}(k-1) \\ \ldots & \ldots & \ldots & \ldots \\ \rho_{j,i}^{(P-1)}(k) & \rho_{j,i}^{(P-2)}(k-1) & \ldots & \rho_{i,j}^{(0)}(k-P+1) \end{bmatrix}. \end{aligned} \tag{33}$$

Equation (33) indicates the lower $(P-1)\times(P-1)$ block of $\mathbf{R}_{i,j}(k)$ can be determined by copying the upper $(P-1)\times(P-1)$ block of $\mathbf{R}_{i,j}(k-1)$. Using (31) and (33), the updating of $\mathbf{R}_{i,j}(k)$ and $\boldsymbol{\chi}_{i,j}(k)$ requires only the calculation of $\rho_{i,j}^{(m)}(k), 0 \le m \le P$. The correlation element $\rho_{i,j}^{(m)}(k)$ can be recursively computed as

$$\begin{aligned} \rho_{i,j}^{(m)}(k) &= \rho_{i,j}^{(m)}(k-1) + \sum_{l=0}^{N-1} u_i(kN-l)u_j[(k-m)N-l] \\ &\quad - \sum_{l=0}^{N-1} u_i(kN-M-l)u_j[(k-m)N-M-l]. \end{aligned} \tag{34}$$

Consequently, updating of $\mathbf{R}_{i,j}(k)$ and $\boldsymbol{\chi}_{i,j}(k)$ requires $2N(P+1)$ multiplications and $2N(P+1)$ additions in total.

The FAF approach is presented in Table II. The complexity of this scheme is reduced significantly because filtering and updating require only $O(2M)$ operations, unlike the $O(2PM)$ operations needed for direct implementation. In the special case wherein $N=1$, the proposed approach is reduced to the FAP algorithm [10], [11].

TABLE II
FAST APPROXIMATE FILTERING

Equation	×	+
$\mathbf{R}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k)$ $\boldsymbol{\chi}_{i,j}(k)=\mathbf{U}_j^T(k-1)\mathbf{u}_i(k)$	$2(P+1)N^2$	$2(P+1)N^2$
$e_{i,\mathrm{D}}(k)=d_{i,\mathrm{D}}(k)-\mathbf{u}_i^T(k)\hat{\mathbf{w}}(k-1)$ $-\sum_{j=0}^{N-1}\boldsymbol{\chi}_{i,j}^T(k)\boldsymbol{\varphi}_j(k-1)$	$M+PN$	$M+PN$
$\mathbf{e}_{i,\mathrm{D}}(k)=\begin{bmatrix}e_{i,\mathrm{D}}(k)\\(1-\mu)\bar{\mathbf{e}}_{i,\mathrm{D}}(k-1)\end{bmatrix}$	$P-1$	$-$
$\boldsymbol{\varphi}_i(k)=\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k)+\begin{bmatrix}0\\\bar{\boldsymbol{\varphi}}_i(k-1)\end{bmatrix}$	$-$	P
$\hat{\mathbf{w}}(k+1)=\hat{\mathbf{w}}(k)$ $+\sum_{i=0}^{N-1}\mathbf{u}_i(k-P+1)[\boldsymbol{\varphi}_i(k)]_{P-1}$	M	M
Total $2M+(2P+2)N^2+P(N+1)$ multiplications $2M+(2P+2)N^2+P(N+1)$ additions		

Note that (28) is only an approximate implementation of (10) under the condition that δ is significantly small. If the eigenvalues of $\mathbf{R}(k)$ are "large", one usually does not need regularization parameter. This fast filtering scheme can be useful in this case. In some applications, however, regularization is necessary. For example, the regularization parameter is $10^3\sigma_u^2$ (σ_u^2 is the variance of the input signal $u(n)$) when the signal-to-noise ratio (SNR) is 0 dB and $M=512$ [25]. Thus, the approximation in (26) does not hold again. If (28) is used to calculate the error vector, then the performance does not exactly match that of the original IMSAF algorithm [12]. Moreover, Appendix A shows that given the same regularization parameter, the FAF approach can achieves lower steady-state misalignment. Simulation results indicate that by making adjustment δ, the FAF approach and the standard IMSAF can exhibit similar convergence performance. Nevertheless, it is desirable to derive a fast, yet exact filtering algorithm. We will do that in the next two subsections.

B. Fast Exact Filtering With Weight Vector Calculation

In the first approach, only the first component of each subband error vector is exactly calculated, and the others are approximated. Motivated by the idea presented in [19], we propose a new fast filtering approach in which all the error vector components can be calculated exactly.

Substituting (13) into (4), $\mathbf{y}_{\mathrm{D}}(k)$ can be rewritten as

$$\begin{aligned}\mathbf{y}_{\mathrm{D}}(k)&=\mathbf{U}^T(k)[\mathbf{w}(k-1)+\mathbf{U}(k-1)\boldsymbol{\varepsilon}_{\mathrm{D}}(k-1)]\\&=\mathbf{z}(k)+\mathbf{G}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k-1)\end{aligned}\tag{35}$$

where $\mathbf{z}(k)=\mathbf{U}^T(k)\mathbf{w}(k-1)$ and $\mathbf{G}(k)=\mathbf{U}^T(k)\mathbf{U}(k-1)$. Using (2), $\mathbf{z}(k)$ can be rewritten as

$$\mathbf{z}(k)=[\mathbf{z}_0^T(k),\mathbf{z}_1^T(k),\ldots,\mathbf{z}_{N-1}^T(k)]^T\tag{36}$$

where

$$\mathbf{z}_i(k)=\mathbf{U}_i^T(k)\mathbf{w}(k-1).\tag{37}$$

When (9) is considered, $\mathbf{z}_i(k)$ can be expressed as

$$\begin{aligned}\mathbf{z}_i(k)&=[\mathbf{u}_i^T(k)\mathbf{w}(k-1),\mathbf{u}_i^T(k-1)\mathbf{w}(k-1),\\&\quad\ldots,\mathbf{u}_i^T(k-P+1)\mathbf{w}(k-1)]^T\\&=[z_i(k),y_{i,\mathrm{D}}^{(0)}(k-1),..,y_{i,\mathrm{D}}^{(P-2)}(k-1)]^T\end{aligned}\tag{38}$$

TABLE III
FAST EXACT FILTERING WITH WEIGHT VECTOR CALCULATION

Equation	×	+
$z_i(k)=\mathbf{u}_i^T(k)\mathbf{w}(k-1)$	M	$M-1$
$\mathbf{z}_i(k)=[z_i(k),y_{i,\mathrm{D}}^{(0:P-2)}(k-1)]^T$	$-$	$-$
$\mathbf{G}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k-1)$ $\mathbf{R}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k)$	$2(P+1)N^2$	$2(P+1)N^2$
$\mathbf{y}_{\mathrm{D}}(k)=\mathbf{z}(k)+\mathbf{G}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k-1)$	NP^2	NP^2
$\mathbf{e}_{\mathrm{D}}(k)=\mathbf{d}_{\mathrm{D}}(k)-\mathbf{y}_{\mathrm{D}}(k)$	$-$	P
$\mathbf{w}(k+1)=\mathbf{w}(k)+\mathbf{U}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k)$	PM	PM
Total $(P+1)M+(2P+2)N^2+NP^2$ multiplications $(P+1)M+(2P+2)N^2+NP^2+P$ additions		

where

$$z_i(k)=\mathbf{u}_i^T(k)\mathbf{w}(k-1).\tag{39}$$

The last $P-1$ elements of $\mathbf{z}_i(k)$ can be obtained directly from $\mathbf{y}_{i,\mathrm{D}}(k-1)$. To update $\mathbf{z}_i(k)$, we need only compute $z_i(k)$ that requires M multiplications and $M-1$ additions.

In (35), updating $\mathbf{y}_{\mathrm{D}}(k)$ necessitates the calculation of $\mathbf{G}(k)$, which can also be expressed in the following block matrix form:

$$\begin{aligned}\mathbf{G}(k)&=\mathbf{U}^T(k)\mathbf{U}(k-1)\\&=\begin{bmatrix}\mathbf{G}_{0,0}(k)&\mathbf{G}_{0,1}(k)&\ldots&\mathbf{G}_{0,N-1}(k)\\\mathbf{G}_{1,0}(k)&\mathbf{G}_{1,1}(k)&\ldots&\mathbf{G}_{1,N-1}(k)\\\ldots&\ldots&\ldots&\ldots\\\mathbf{G}_{N-1,0}(k)&\mathbf{G}_{N-1,1}(k)&\ldots&\mathbf{G}_{N-1,N-1}(k)\end{bmatrix}\end{aligned}\tag{40}$$

where $\mathbf{G}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k-1)$, $i=0,\ldots,N-1;j=0,\ldots,N-1$. Using (30), we can express $\mathbf{G}_{i,j}(k)$ as

$$\begin{aligned}&\mathbf{G}_{i,j}(k)\\&=\begin{bmatrix}\rho_{i,j}^{(1)}(k)&\rho_{i,j}^{(2)}(k)&\ldots&\rho_{i,j}^{(P)}(k)\\\rho_{j,i}^{(0)}(k-1)&\rho_{i,j}^{(1)}(k-1)&\ldots&\rho_{i,j}^{(P-1)}(k-1)\\\ldots&\ldots&\ldots&\ldots\\\rho_{j,i}^{(P-2)}(k-1)&\rho_{j,i}^{(P-3)}(k-2)&\ldots&\rho_{i,j}^{(1)}(k-P+1)\end{bmatrix}.\end{aligned}\tag{41}$$

In addition, we must update $\mathbf{R}(k)$. By means of (41) and (33), most elements of $\mathbf{R}_{i,j}(k)$ and $\mathbf{G}_{i,j}(k)$ can be taken from $\mathbf{R}_{i,j}(k-1)$. Updating $\mathbf{R}_{i,j}(k)$ and $\mathbf{G}_{i,j}(k)$ requires only the computation of $\rho_{i,j}^{(m)}(k), 0\le m\le P$. This process requires a total of $2N(P+1)$ multiplications and $2N(P+1)$ additions.

This approach provides an exact implementation of (4) and prevents any numerical instability. The proposed fast filtering technique reduces the filtering complexity from $O(PM)$ operations, as in the original IMSAF algorithm, to $O(M)$ operations per sample. This approach is summarized in Table III. For convenience, the approach is known as "FEF1". The fast exact AP algorithm [19] is a special case of the proposed approach at $N=1$.

C. Fast Exact Filtering Without Weight Vector Calculation

The second approach is appealing because it facilitates exact filtering. However, the calculation of $z_i(k)$ requires the explicit update of $\mathbf{w}(k)$ that needs $O(PM)$ operations per sample,

TABLE IV
FAST EXACT FILTERING WITHOUT WEIGHT VECTOR CALCULATION

Equation	$\times$	$+$
$\mathbf{G}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k-1)$ $\mathbf{R}_{i,j}(k)=\mathbf{U}_i^T(k)\mathbf{U}_j(k)$ $\boldsymbol{\psi}_{i,j}(k)=\mathbf{U}_j^T(k-2)\mathbf{u}_i(k)$	$2(P+2)N^2$	$2(P+2)N^2$
$z_i(k)=\mathbf{u}_i^T(k)\hat{\mathbf{w}}(k-2)$ $+\sum_{j=0}^{N-1}\boldsymbol{\psi}_{i,j}^T(k)\boldsymbol{\varphi}_j(k-2)$	$M+NP$	$M+NP$
$\mathbf{z}_i(k)=[z_i(k),y_{i,\mathrm{D}}^{(0:P-2)}(k-1)]^T$	$-$	$-$
$\mathbf{y}_{\mathrm{D}}(k)=\mathbf{z}(k)+\mathbf{G}(k)\boldsymbol{\varepsilon}_{\mathrm{D}}(k-1)$	NP^2	NP^2
$\mathbf{e}_{\mathrm{D}}(k)=\mathbf{d}_{\mathrm{D}}(k)-\mathbf{y}_{\mathrm{D}}(k)$	$-$	P
$\boldsymbol{\varphi}_i(k)=\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k)+\begin{bmatrix}0\\ \bar{\boldsymbol{\varphi}}_i(k-1)\end{bmatrix}$	$-$	P
$\hat{\mathbf{w}}(k+1)=\hat{\mathbf{w}}(k)$ $+\sum_{i=0}^{N-1}\mathbf{u}_i(k-P+1)[\boldsymbol{\varphi}_i(k)]_{P-1}$	M	M
Total $2M+(2P+4)N^2+N(P^2+P)$ multiplications $2M+(2P+4)N^2+N(P^2+P)+2P$ additions		

which contributes the most to the algorithm complexity. However, the weight vector calculation is not a primary concern in many applications. By exploiting the special structure of the matrix $\mathbf{U}(k)$, it is sufficient to calculate the error vector exactly even if $\mathbf{w}(k)$ is not available in every sampling period.

Given the special expression of $\mathbf{w}(k+1)$ in (17), $z_i(k)$ can be calculated using the auxiliary coefficient vector $\hat{\mathbf{w}}(k)$ rather than $\mathbf{w}(k)$. Substituting (17) into (39) yields

$$\begin{aligned} z_i(k)&=\mathbf{u}_i^T(k)\left[\hat{\mathbf{w}}(k-2)+\sum_{j=0}^{N-1}\mathbf{U}_j(k-2)\boldsymbol{\varphi}_j(k-2)\right]\\ &=\mathbf{u}_i^T(k)\hat{\mathbf{w}}(k-2)+\sum_{j=0}^{N-1}\boldsymbol{\psi}_{i,j}^T(k)\boldsymbol{\varphi}_j(k-2) \end{aligned} \tag{42}$$

where

$$\begin{aligned} \boldsymbol{\psi}_{i,j}(k)&=\mathbf{U}_j^T(k-2)\mathbf{u}_i(k)\\ &=[\mathbf{u}_i^T(k)\mathbf{u}_j(k-2),\mathbf{u}_i^T(k)\mathbf{u}_j(k-3),\\ &\quad\ldots,\mathbf{u}_i^T(k)\mathbf{u}_j(k-P-1)]^T\\ &=[\rho_{i,j}^{(2)}(k),\rho_{i,j}^{(3)}(k),\ldots,\rho_{i,j}^{(P+1)}(k)]^T. \end{aligned} \tag{43}$$

The first $P-1$ elements of $\boldsymbol{\psi}_{i,j}(k)$ are calculated while updating matrix $\mathbf{G}_{i,j}(k)$. Consequently, $\boldsymbol{\psi}_{i,j}(k)$ calculation requires only the computation of $\rho_{i,j}^{(P+1)}(k)$ by using (34).

This fast filtering approach uses the auxiliary coefficient vector $\hat{\mathbf{w}}(k)$ in place of $\mathbf{w}(k)$ to calculate the filtering out vector $\mathbf{y}_{\mathrm{D}}(k)$. Update of $\mathbf{w}(k)$ needs PM operations while calculation of $\hat{\mathbf{w}}(k)$ requires only M operations per sample. This fast exact filtering algorithm, named "FEF2," is summarized in Table IV.

This fast filtering approach is only marginally more complex than the first one; however, the former calculates the error signal vector exactly. In fact, the third fast exact filtering approach can be applied as a general framework to many adaptive filtering algorithms that have the similar filtering and update structures in (4) and (13); see [26] and [27] for more details.

We should stress that almost all of the relations in this section hold irrespective of structure, with the exception of the calculation of correlation element in (30) based on the time-shift structure. This finding indicates that it is possible to derive fast filtering approaches for situations in which successive regressors are not time-shifted versions of one another.

V. DECORRELATION PROPERTY OF IMSAF

A. Design of the SIMSAF Algorithm

The computational savings attributed to fast filtering approaches can be increased by calculating matrix inversion efficiently. In this section, we analyze the structure of the correlation matrix $\mathbf{R}(k)$ and propose a simplified version of the IMSAF algorithm.

The cross-correlation function $\gamma_{ij}(l)$ of two arbitrary subband signals $u_i(n)$ and $u_j(n)$ can be formulated as [2]

$$\begin{aligned} \gamma_{ij}(l)&=E[u_i(n)u_j(n-l)]\\ &=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|H_i(e^{j\omega})\right|\left|H_j(e^{j\omega})\right|\Gamma_{uu}(e^{j\omega})e^{j[\phi_{ij}(\omega)+\omega l]}d\omega \end{aligned} \tag{44}$$

where $|H_i(e^{j\omega})|$ and $|H_j(e^{j\omega})|$ are the magnitude responses of the analysis filters, $\phi_{ij}(\omega)=\phi_i(\omega)-\phi_j(\omega)$ is their phase difference, and $\Gamma_{uu}(e^{j\omega})$ is the power spectrum of the input signal. Assuming ergodicity, $\gamma_{ij}(lN)$ can be approximated with the time average $\hat{\gamma}_{ij}(lN)=\mathbf{u}_i^T(k-m)\mathbf{u}_j(k-m-l)/M$. As per (44), if the magnitude responses of the analysis filters do not overlap significantly, then the cross-correlation $\gamma_{ij}(lN)$ is negligible compared to the autocorrelation $\gamma_{ii}(0)$. In other words, the elements $\mathbf{u}_i^T(k-m)\mathbf{u}_j(k-m-l)=M\hat{\gamma}_{ij}(lN)$ of $\mathbf{R}_{i,j}(k)$ are much smaller than its diagonal elements $\mathbf{u}_i^T(k)\mathbf{u}_i(k)=M\hat{\gamma}_{ii}(0)$. Therefore, $\mathbf{R}_{i,j}(k)$ can be neglected in comparison with $\mathbf{R}_{i,i}(k)$. However, the subband signal still displays high autocorrelation for highly colored signals. Therefore, the off-diagonal elements of $\mathbf{R}_{i,i}(k)$ cannot be neglected in comparison with its diagonal elements [7].

At this point, we present numerical results to understand the nature of $\mathbf{R}(k)$. The input signals are an AR(1) process with coefficients $(1,-0.9)$ and an AR(10) process with coefficients (5.3217, -9.2948, 7.0933, -2.8152, 2.5805, -2.4230, 0.3747, 2.2628, -0.3028, -1.7444, 1.1053) [2]. The latter is a speech-like signal and exhibits a large spectral dynamic range. The power spectra of these signals are plotted in Figs. 2(a) and 2(b), respectively. Cosine modulated filter banks [28] are used for the subband structure. Figs. 2(c) and 2(d) provide the pictorial representations of $\mathbf{R}(k)$ for AR(1) and AR(10) signals, where we choose $N=4$, $P=4, L=64$ and each element $[\mathbf{R}(k)]_{i,j}$ is normalized by the corresponding leading diagonal element $[\mathbf{R}(k)]_{i,i}$. The elements of $\mathbf{R}_{i,j}(k)$ are considerably smaller than those of $\mathbf{R}_{i,i}(k)$; thus, the former can be neglected.

Thus, $\mathbf{R}(k)$ can be simplified to

$$\begin{aligned} \mathbf{R}(k)&=\mathbf{U}^T(k)\mathbf{U}(k)\\ &\approx\begin{bmatrix}\mathbf{R}_{0,0}(k) & \mathbf{0} & \ldots & \mathbf{0}\\ \mathbf{0} & \mathbf{R}_{1,1}(k) & \ldots & \mathbf{0}\\ \ldots & \ldots & \ldots & \ldots\\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{R}_{N-1,N-1}(k)\end{bmatrix}. \end{aligned} \tag{45}$$

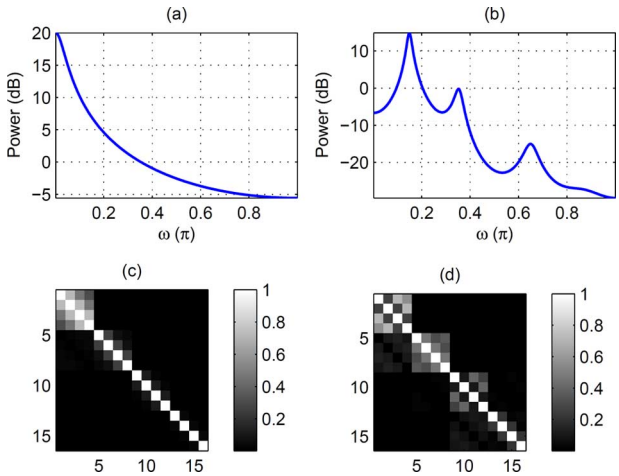


Fig. 2. (a) Power spectral of the AR(1) process. (b) Power spectral of the AR(10) process. Pictorial representation of the matrix $\mathbf{R}(k)$ for (c) AR(1) input and (d) AR(10) input.

Using (45) and the fact that the inverse of a block diagonal matrix is composed of the inverse of each block, (13) can be rewritten as

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) + \mu\mathbf{U}(k)[\mathbf{U}^T(k)\mathbf{U}(k) + \delta\mathbf{I}]^{-1}\mathbf{e}_{\mathrm{D}}(k) \\ &= \mathbf{w}(k) + \mu\sum_{i=0}^{N-1}\mathbf{U}_i(k)[\mathbf{U}_i^T(k)\mathbf{U}_i(k) + \delta\mathbf{I}]^{-1}\mathbf{e}_{i,\mathrm{D}}(k). \quad (46)\end{aligned}$$

One advantage of the simplification above lies in the complexity reduction. The IMSAF algorithm in (13) must implement a matrix inversion with size $NP \times NP$. By contrast, the SIMSAF algorithm in (46) involves N inversions of the matrices with size $P \times P$. For a direct implementation, the complexity of the former is $O(N^3P^3)$, whereas that of the latter is $O(NP^3)$. We assume that calculation of $\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k) = \mu[\mathbf{R}_{i,i}(k) + \delta\mathbf{I}]^{-1}\mathbf{e}_{i,\mathrm{D}}(k), i = 0, 1, \ldots, N-1$ needs P_{m2} multiplications and P_{a2} additions per sample. The sole difference between the IMSAF and SIMSAF algorithms lies in the calculation of $\boldsymbol{\varepsilon}_{\mathrm{D}}(k)$. Thus, the fast filtering approaches presented in Section IV can also be applied to the SIMSAF algorithm without any modification.

B. Decorrelation Property of the IMSAF Algorithm

Moreover, we can obtain an intuitive interpretation of the IMSAF algorithm according to (46) [29]. In the special case wherein $\mu = 1.0$ and $\delta = 0$, Appendix B proves that the weight update (46) can be written as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \sum_{i=0}^{N-1}\frac{\boldsymbol{\Phi}_i(k)}{\boldsymbol{\Phi}_i^T(k)\boldsymbol{\Phi}_i(k)}e_{i,\mathrm{D}}(k) \quad (47)$$

where

$$\boldsymbol{\Phi}_i(k) = \mathbf{u}_i(k) - \mathbf{D}_i(k)\mathbf{a}_i(k), \quad (48)$$

$$\mathbf{D}_i(k) = [\mathbf{u}_i(k-1), \ldots, \mathbf{u}_i(k-P+1)], \quad (49)$$

$$\mathbf{a}_i(k) = [\mathbf{D}_i^T(k)\mathbf{D}_i(k)]^{-1}\mathbf{D}_i^T(k)\mathbf{u}_i(k). \quad (50)$$

Interestingly, (47) provides new insight into the IMSAF algorithm. The IMSAF algorithm with step size $\mu = 1.0$ runs with a decorrelated direction subband vector $\boldsymbol{\Phi}_i(k)$ rather than with the original input vector $\mathbf{u}(n)$. As a consequence, the IMSAF algorithm uses two prewhitening techniques to accelerate the convergence. First, the input signal $u(n)$ is filtered by the analysis filters to obtain the subband signals $u_i(n)$ that have flatter spectrum [2]. Second, each of the subband signal $u_i(n)$ is prewhitened by a decorrelation filter $[1, -\mathbf{a}_i^T(k)]$ before this signal is incorporated into the adaptive filtering algorithm. If the number of subbands N and projection order P are properly chosen, then $\boldsymbol{\Phi}_i(k)$ is a vector whose elements are estimates of a white random process. When an AR(P) process is decomposed into N subbands, the ith subband signal can be modeled by an AR process with less order [30]. Thus, the projection order needed by the IMSAF algorithm can be less than that required by the AP algorithm to achieve the same convergence rate.

C. Pseudo IMSAF Algorithm

The IMSAF variant in (47) is interesting in that this form resembles a projection-type gradient algorithm, such as NLMS. Introducing a step size $0 < \mu < 1$ can enhance flexibility and facilitate a tradeoff between the convergence speed and steady-state misalignment. Given μ, the weight vector update (47) becomes

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\sum_{i=0}^{N-1}\frac{\boldsymbol{\Phi}_i(k)}{\boldsymbol{\Phi}_i^T(k)\boldsymbol{\Phi}_i(k) + \delta}e_{i,\mathrm{D}}(k) \quad (51)$$

where a regularization factor δ is also used to assist in handling the numerical instability. When the step size $\mu \neq 1$, the IMSAF property is not satisfied again in (51). Thus, the result is known as the pseudo IMSAF algorithm. As per (89) in Appendix B, we note that $\boldsymbol{\Phi}_i^T(k)\boldsymbol{\Phi}_i(k) = \mathbf{u}_i^T(k)\boldsymbol{\Phi}_i(k)$ and thus (51) can be reformulated into

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\sum_{i=0}^{N-1}\frac{\boldsymbol{\Phi}_i(k)}{\boldsymbol{\Phi}_i^T(k)\mathbf{u}_i(k) + \delta}e_{i,\mathrm{D}}(k). \quad (52)$$

The update equation of the pseudo AP (PAP) algorithm in [31] is thus a special case of (51), whereas the update equation of another PAP algorithm in [32] is a special case of (52). The update (51) and (52) are indeed equal; therefore, the two variants of PAP algorithm in [31] and [32] are also mathematically equivalent. This phenomenon has not been reported previously.

The pseudo IMSAF algorithm in either (51) or (52) remains too complex for implementation. The major contributor to this is (48) because calculation of $\boldsymbol{\Phi}_i(k)$ requires $(P-1)M$ multiplications. In practice, the estimates of the linear predictor coefficient $\mathbf{a}_i(k)$ can be treated as time-invariant values in a short period of time (the speech signal is stationary for the duration for approximately 10 ms). We can then approximate $\boldsymbol{\Phi}_i(k)$ through a tapped-delay line, i.e., $\phi_{i,m}(k) = \phi_{i,m-N}(k-1), m = N, \ldots, M-1$. The top N elements of $\boldsymbol{\Phi}_i(k)$ can be computed as:

$$\phi_{i,m}(k) = u_i(kN-m) - \boldsymbol{\Theta}_{i,m}^T(k)\mathbf{a}_i(k), m = 0, \ldots, N-1 \quad (53)$$

where $\boldsymbol{\Theta}_{i,m}(k) = [u_i(kN-N-m), u_i(kN-2N-m), \ldots, u_i(kN-(P-1)N-m)]^T$. Using this approximation, update of $\boldsymbol{\Phi}_i(k)$ requires $P-1$ multiplications. Thus, the total complexity is significantly reduced. The pseudo IMSAF

TABLE V
PSEUDO IMSAF ALGORITHM

Equation	×	+
$e_{i,\mathrm{D}}(k) = d_{i,\mathrm{D}}(k) - \mathbf{u}_i^T(k)\mathbf{w}(k)$	M	$M-1$
$\boldsymbol{\tau}_i(k) = \mathbf{D}_i^T(k)\mathbf{u}_i(k)$ $\mathbf{J}_i(k) = \mathbf{D}_i^T(k)\mathbf{D}_i(k)$	$2NP$	$2NP$
$\mathbf{a}_i(k) = \mathbf{J}_i^{-1}(k)\boldsymbol{\tau}_i(k)$	P_{m3}	P_{a3}
$\phi_{i,m}(k) = \phi_{i,m-N}(k-1)$ $m = N, ..., M-1$	$-$	$-$
$\phi_{i,m}(k) = u_i(kN-m) - \boldsymbol{\Theta}_{i,m}^T(k)\mathbf{a}_i(k)$ $m = 0, ..., N-1$	$P-1$	$P-1$
$\mathbf{w}(k+1) = \mathbf{w}(k) + \sum_{i=0}^{N-1} \frac{\mu\boldsymbol{\Phi}_i(k)e_{i,\mathrm{D}}(k)}{\boldsymbol{\Phi}_i^T(k)\boldsymbol{\Phi}_i(k)+\delta}$	M	M
Total $2M + 2NP + P + P_{m3} + (N+2)L$ multiplications $2M + 2NP + P + P_{a3} + (N+2)(L-1)$ additions		

TABLE VI
CG ALGORITHM

Equation	×	+
$\mathbf{x} = \mathbf{0}, \mathbf{r} = \mathbf{b}, \theta_0 = \mathbf{r}^T\mathbf{r}, \mathbf{d} = \mathbf{r}$	$-$	$-$
for $l = 1, ..., N_u$	$-$	$-$
if $l > 1$	$-$	$-$
$\mathbf{d} = \mathbf{r} + (\theta_{l-1}/\theta_{l-2})\mathbf{d}$	P	P
endif	$-$	$-$
$\mathbf{v} = \mathbf{A}\mathbf{d}$	P^2	P^2-P
$\alpha = \theta_{l-1}/\mathbf{d}^T\mathbf{v}$	P	$P-1$
$\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$	P	P
$\mathbf{r} = \mathbf{r} - \alpha\mathbf{v}$	P	P
$\theta_l = \mathbf{r}^T\mathbf{r}$	P	$P-1$
endfor	$-$	$-$
Total $P(P+5)N_u$ multiplications $(P^2+4P-2)N_u$ additions		

algorithm is summarized in Table VI. The matrix $\mathbf{J}_i(k)$ and the vector $\boldsymbol{\tau}_i(k)$ can be updated by using (30). The linear system (50) is similar to the normal equation in the SIMSAF algorithm; therefore, both equations can be solved with the same method. For simplicity, an exact solution of (50) is used in the simulation.

The computational cost of the pseudo IMSAF algorithm is only slightly more complex than that of the MSAF algorithm. As per the simulation, however, the convergence speed of the former is higher than that of the latter. Hence, the pseudo IMSAF algorithm can serve as a feasible solution for consumer devices that typically possess limited resources.

VI. SOLUTION TO THE LINEAR SYSTEMS OF EQUATIONS

A. Problem Description

At this point, the remaining problem involves solving the linear system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{54}$$

where $\mathbf{A}$ is a symmetric positive definite matrix, and $\mathbf{x}$ is a vector. In the IMSAF algorithm, $\mathbf{A} = \mathbf{R}(k) + \delta\mathbf{I}$, $\mathbf{x} = \boldsymbol{\varepsilon}_{\mathrm{D}}(k)$, and $\mathbf{b} = \mu\mathbf{e}_{\mathrm{D}}(k)$. In the SIMSAF algorithm, $\mathbf{A} = \mathbf{R}_{i,i}(k) + \delta\mathbf{I}$, $\mathbf{x} = \boldsymbol{\varepsilon}_{i,\mathrm{D}}(k)$, and $\mathbf{b} = \mu\mathbf{e}_{i,\mathrm{D}}(k)$. Many methods have been proposed to solve this problem [33]. For simplicity, however, we focus only on the linear system of equations in the SIMSAF algorithm in the following sections. Those results can easily be extended to the IMSAF algorithm.

B. Solutions

1) Exact Solution: Since $\tilde{\mathbf{R}}_{i,i}(k) = \mathbf{R}_{i,i}(k) + \delta\mathbf{I}$ is symmetric, the matrix $\mathbf{LDL}^T$ factorization [17], [33] can be used to provide an exact solution of (54). The matrix $\tilde{\mathbf{R}}_{i,i}(k)$ can be uniquely factored into

$$\tilde{\mathbf{R}}_{i,i}(k) = \mathbf{LDL}^T \tag{55}$$

where $\mathbf{D}$ is a diagonal matrix, and $\mathbf{L}$ is a unit lower triangular matrix. We can then use the forward and backward substitutions to solve $\tilde{\mathbf{R}}_{i,i}(k)\boldsymbol{\varepsilon}_{i,\mathrm{D}}(k) = \mu\mathbf{e}_{i,\mathrm{D}}(k)$. Interested readers may refer to [17] for more details. This scheme needs $P^3/6 + 2P^2 - 7P/6$ multiplications/additions, as well as P divisions.

2) Levinson Recursion: For most applications, $M \gg P$, the following relation holds

$$\rho_{i,i}^{(m)}(k) \approx \rho_{i,i}^{(m)}(k-1) \approx \cdots \approx \rho_{i,i}^{(m)}(k-P+1). \tag{56}$$

Thus, the elements on each diagonal of $\mathbf{R}_{i,i}(k)$ can be approximated as equal, i.e.,

$$\mathbf{R}_{i,i}(k) \approx \begin{bmatrix} \rho_{i,i}^{(0)}(k) & \rho_{i,i}^{(1)}(k) & \cdots & \rho_{i,i}^{(P-1)}(k) \\ \rho_{i,i}^{(1)}(k) & \rho_{i,i}^{(0)}(k) & \cdots & \rho_{i,i}^{(P-2)}(k) \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{i,i}^{(P-1)}(k) & \rho_{i,i}^{(P-2)}(k) & \cdots & \rho_{i,i}^{(0)}(k) \end{bmatrix}. \tag{57}$$

Since $\mathbf{R}_{i,i}(k)$ is simplified as a Toeplitz matrix, the Levinson recursion [33] can be used to solve the linear system with $2P^2 + 4P$ multiplications, $2P^2 + P$ additions, and P divisions. This approach only gives an approximate solution because the actual matrix $\mathbf{R}_{i,i}(k)$ is not strictly Toeplitz.

Solving the normal (54) is equivalent to minimizing the quadratic function [33]:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}. \tag{58}$$

Several iterative algorithms have been proposed to solve the linear system of equations based on (58). Specifically, we are interested in low-complexity and robust line search methods, e.g., the dichotomous coordinate descent (DCD) and conjugate gradient (CG) algorithms.

3) CG Algorithm: The CG method [34], [35] is an iterative method that minimizes the quadratic cost function (58). This method has been adopted to solve the linear equations in the FAP and RLS algorithms [17], [36]. In this study, we would like to use the CG method to calculate $[\mathbf{U}^T(k)\mathbf{U}(k) + \delta\mathbf{I}]^{-1}\mathbf{e}_{\mathrm{D}}(k)$ and $[\mathbf{U}_i^T(k)\mathbf{U}_i(k) + \delta\mathbf{I}]^{-1}\mathbf{e}_{i,\mathrm{D}}(k)$ in (13) and (46), respectively. This algorithm converges quickly; however, its complexity is expensive especially for a large number of iterations N_u. Table VI describes the CG algorithm.

4) DCD Algorithm: A new, inexact line search method, i.e., the DCD algorithm [36]–[38], has also been proposed to solve the linear system of equations. In this algorithm, the step-size α can take on one of predefined M_b values corresponding to binary representation of elements of $\mathbf{x}$ within an amplitude range

TABLE VII
DCD ALGORITHM [36]

Equation	$\times$	$+$
$\mathbf{x}=\mathbf{0}, \mathbf{r}=\mathbf{b}, \alpha=H/2, m=1$	$-$	$-$
for $l=1,...,N_u$	$-$	$-$
$q=\arg\max_{i=0,\ldots,P-1}\{\lvert[\mathbf{r}]_i\rvert\}$	$-$	$P-1$
while $\lvert[\mathbf{r}]_q\rvert \le (\alpha/2)\,[\mathbf{A}]_{q,q} \& m \le M_b$ $m=m+1, \alpha=\alpha/2$ endwhile	$-$	1
if $m>M_b$ break endif	$-$	$-$
$[\mathbf{x}]_q=[\mathbf{x}]_q+\text{sign}([\mathbf{r}]_q)\alpha$	$-$	1
$\mathbf{r}=\mathbf{r}-\text{sign}([\mathbf{r}]_q)\alpha\mathbf{A}^{(q)}$	$-$	P
endfor	$-$	$-$
Total 0 multiplications $2N_uP+M_b$ additions		

$[-H, H]$. The algorithm starts the iterative search from the most significant bit (MSB) of elements in $\mathbf{x}$. As the MSB is updated, the algorithm proceeds to the next significant bit. Parameter N_u denotes the maximum number of "successful" iterations. The DCD algorithm requires no multiplication, no division, and no square root operations, which is preferable for implementations on hardware platforms, e.g., FPGA and ASIC. The DCD algorithm is outlined in Table VII.

In the following sections, we use SIMSAF-FEF2-DCD to denote that the FEF2 approach has been adopted to calculate error vector while the DCD method has been employed to solve the linear system of equations. Incorporating DCD iterations into the FEF1 and FEF2 approaches produces the SIMSAF-FEF1-DCD and SIMSAF-FEF2-DCD algorithms, respectively. The performance levels of both algorithms are identical; the sole difference between the two algorithms lies in structural complexity. In the SIMSAF-FEF1-DCD algorithm, weight updates are incorporated into the DCD iterations without explicit multiplication [19]. The number of multiplication operations required by the SIMSAF-FEF1-DCD algorithm is thus significantly reduced. However, the number of the addition procedures needed for weight vector adaptation remains large and increases with N_u. The SIMSAF-FEF2-DCD algorithm performs more M multiplication operations than the SIMSAF-FEF1-DCD algorithm but saves $(N_u-1)M$ addition procedures. Thus, the total operations of the SIMSAF-FEF2-DCD algorithm are considerably reduced. Each algorithm may have its own merit depending on the implementation platform used, such as DSP, FPGA, and ARM [37].

C. Computational Complexity

By combining different filtering schemes and matrix inversion methods, we can derive several implementations of the IMSAF algorithm. Table VIII compares the complexity of several fast versions of the IMSAF algorithm with respect to the number of additions and multiplications per sample. Table IX presents the complexity of various matrix inversion operations. The $\mathbf{LDL}^T$ approach is applied to in the original IMSAF and SIMSAF algorithms in order to provide a benchmark.

Table X shows an example of the computational load comparison of several variants. The parameters are $P=8, N=8, M=1024$. Compared to the original IMSAF algorithms, the SIMSAF-FE1-$\mathbf{LDL}^T$ and SIMSAF-FE2-$\mathbf{LDL}^T$ algorithms save approximately 52% and 81% of computation, respectively. The complexity reduction is attributed to both of the fast filtering approaches and simplified matrix inversion operations. The complexity of filtering and updating in the FEF1 and FEF2 approaches is only 63% and 22% of that in the original IMSAF algorithm, respectively. Note that solving a linear system of equations with size 64×64 using the $\mathbf{LDL}^T$ approach in the IMSAF algorithm needs 6476 operations per sample. By contrast, the SIMSAF algorithm needs to solve only eight linear systems of equations with size 8×8 that only requires 203 operations per sample. When the projection order P is small ($P<8$), the complexity reduction of the Levinson recursion is limited compared to the $\mathbf{LDL}^T$ approach. In applications such as AEC, the projection order is often set between 2 to 10 [17]; thus, the use of the $\mathbf{LDL}^T$ approach is encouraged for practical applications. The DCD approach does not need multiplication and division, which is preferable for hardware implementations. When the number of iterations N_u is large, the complexity of the CG method is quite expensive. Fortunately, a small number of iterations can give good result [36]. Moreover, the complexities of the FAF and FEF2 approaches are similar; however, the latter can provide an exact filtering. The pseudo IMSAF is the least complex of all fast versions of IMSAF.

VII. SIMULATION RESULTS

Computer simulations are conducted to evaluate the performance of the proposed fast algorithms in the context of system identification. The impulse response $\mathbf{w}_o$ is generated according to $w_i=e^{-\tau i}r(i), i=0,1,\ldots,M-1$, where $r(i)$ is a zero-mean white noise sequence and τ is the envelope decay rate. The sampling rate is 8 kHz. The AR(10) process depicted in Fig. 2 is adopted as input signal for Examples 1, 2, 4, 5, 6 and 7. A white noise is added to the echo signal, with different SNR levels. The convergence performance is evaluated in terms of the mean-square error (MSE) or normalized misalignment, defined as $10\log_{10}\left[\|\mathbf{w}_o-\mathbf{w}(k)\|^2/\|\mathbf{w}_o\|^2\right]$. The results are obtained by averaging over 100 Monte Carlo trials for AR process and white noise input but just once for other input signals.

In the following simulations, the exact filtering approaches are adopted to calculate the error vector in all examples except for Example 1. Since FEF1 and FEF2 both provide the exact filtering, we do not indicate which approach is used. For example, "IMSAF-DCD" indicates that the DCD algorithm is used to solve the linear system of equations while either FEF1 or FEF2 can be employed to compute the error vector.

A. Example 1: Effect of the Regularization Parameter on the Performance of the IMSAF-FAF

Fig. 3 compares the convergence performance of the IMSAF-FAF using different regularization parameters δ. An exact solution of the matrix inversion is applied in this example. In Fig. 3(a), the input signal is white noise with $\sigma_u^2=0.01$ and the SNR is 10 dB. When $\delta=0.5\,\sigma_u^2$, the performance of the IMSAF-FAF and original IMSAF algorithms are indistinguishable. However, when a large regularization parameter δ

TABLE VIII
COMPLEXITY OF THE FAST IMSAF ALGORITHMS

Algorithm	Number of multiplications per sample	Number of additions per sample
IMSAF	$2PM + 2N^2P + P_{m1} + (N+2)L$	$2PM + 2N^2P + P_{a1} + (N+2)(L-1)$
SIMSAF	$2PM + 2NP + P_{m2} + (N+2)L$	$2PM + 2NP + P_{a2} + (N+2)(L-1)$
Pseudo IMSAF	$2M + 2NP + P + P_{m3} + (N+2)L$	$2M + 2NP + P + P_{a3} + (N+2)(L-1)$
IMSAF-FAF	$2M + (2P+2)N^2 + P(N+1) + P_{m1} + (N+2)L$	$2M + (2P+2)N^2 + P(N+1) + P_{a1} + (N+2)(L-1)$
SIMSAF-FAF	$2M + (2P+2)N^2 + P(N+1) + P_{m2} + (N+2)L$	$2M + (2P+2)N^2 + P(N+1) + P_{a2} + (N+2)(L-1)$
IMSAF-FE1	$(P+1)M + (2P+2)N^2 + NP^2 + P_{m1} + (N+2)L$	$(P+1)M + (2P+2)N^2 + NP^2 + P_{a1} + (N+2)(L-1)$
SIMSAF-FE1	$(P+1)M + (2P+2)N^2 + NP^2 + P_{m2} + (N+2)L$	$(P+1)M + (2P+2)N^2 + NP^2 + P_{a2} + (N+2)(L-1)$
IMSAF-FE2	$2M + (2P+4)N^2 + N(P^2+P) + P_{m1} + (N+2)L$	$2M + (2P+4)N^2 + N(P^2+P) + P_{a1} + (N+2)(L-1)$
SIMSAF-FE2	$2M + (2P+4)N^2 + N(P^2+P) + P_{m2} + (N+2)L$	$2M + (2P+4)N^2 + N(P^2+P) + P_{a2} + (N+2)(L-1)$

TABLE IX
COMPLEXITY OF SEVERAL MATRIX INVERSION ALGORITHMS

Algorithm	P_{m1}	P_{a1}	P_{m2}	P_{a2}
$\mathbf{LDL}^T$	$\frac{N^2P^3}{6} + 2NP^2 - \frac{7P}{6}$	$\frac{N^2P^3}{6} + 2NP^2 - \frac{7P}{6}$	$\frac{P^3}{6} + 2P^2 - \frac{7P}{6}$	$\frac{P^3}{6} + 2P^2 - \frac{7P}{6}$
Levinson	$2NP^2 + 4P$	$2NP^2 + P$	$2P^2 + 4P$	$2P^2 + P$
CG	$(NP^2 + 5P)N_u$	$(NP^2 + 5P)N_u$	$P(P+5)N_u$	$(P^2 + 4P - 2)N_u$
DCD	0	$2N_uP$	0	$2N_uP + M_b$

TABLE X
COMPLEXITY COMPARISON OF SEVERAL FAST IMSAF ALGORITHMS ($P = 8, N = 8, M = 1024$)

Algorithm	Filtering and weight update		Matrix inversion		Total	
	×	+	×	+	×	+
IMSAF	17408	17408	6476	6476	24524	24514
SIMSAF	16512	16512	203	203	17356	17346
Pseudo IMSAF	2176	2176	147	147	2963	2953
SIMSAF-FAF-$\mathbf{LDL}^T$	3264	3264	203	203	4108	4098
SIMSAF-FAF-Levinson	3264	3264	160	136	4064	4030
SIMSAF-FAF-CG	3264	3264	832	752	4736	4646
SIMSAF-FAF-DCD	3264	3264	0	144	3904	4038
SIMSAF-FEF1-$\mathbf{LDL}^T$	10880	10888	203	203	11724	11722
SIMSAF-FEF1-Levinson	10880	10888	160	136	11680	11654
SIMSAF-FEF1-CG	10880	10888	832	752	12352	12270
SIMSAF-FEF1-DCD	2688	10888	0	144	3328	11662
SIMSAF-FEF2-$\mathbf{LDL}^T$	3904	3920	203	203	4748	4754
SIMSAF-FEF2-Levinson	3904	3920	160	136	4704	4686
SIMSAF-FEF2-CG	3904	3920	832	752	5376	5302
SIMSAF-FEF2-DCD	3904	3920	0	144	4544	4694

$= 200\ \sigma_u^2$ is employed, IMSAF-FAF performance deviates from that of the original IMSAF algorithm and the former achieves smaller misalignment. Because the input signal is white noise, the matrix $\mathbf{U}_i^T(k)\mathbf{U}_i(k)$ has four equal eigenvalues $\lambda = 0.64$. The regularization parameter $0.5\ \sigma_u^2$ (that is 0.005) is much smaller than $\lambda = 0.64$, but $\delta = 200\ \sigma_u^2$ (that is 2.0) is larger than $\lambda = 0.64$. The simulation results agree with the analysis in Appendix A. However, the IMSAF-FAF with $\delta = 80\ \sigma_u^2$ displays the same convergence performance as the standard IMSAF with $\delta = 200\ \sigma_u^2$. In Fig. 3(b), we use the AR(10) process as input and the SNR is 30 dB. The IMSAF-FAF with $\delta = 9.4\ \sigma_u^2$ exhibits almost the same convergence performance as the standard IMSAF with $\delta = 20\ \sigma_u^2$. Extensive simulation were carried out and it is found that by tuning the regularization parameter, the convergence performance of IMSAF-FAF and standard IMSAF algorithms can be similar. This finding indicates that the IMSAF-FAF algorithm can be used in a real-time system, although this algorithm is not an exact implementation of the original IMSAF. Furthermore,

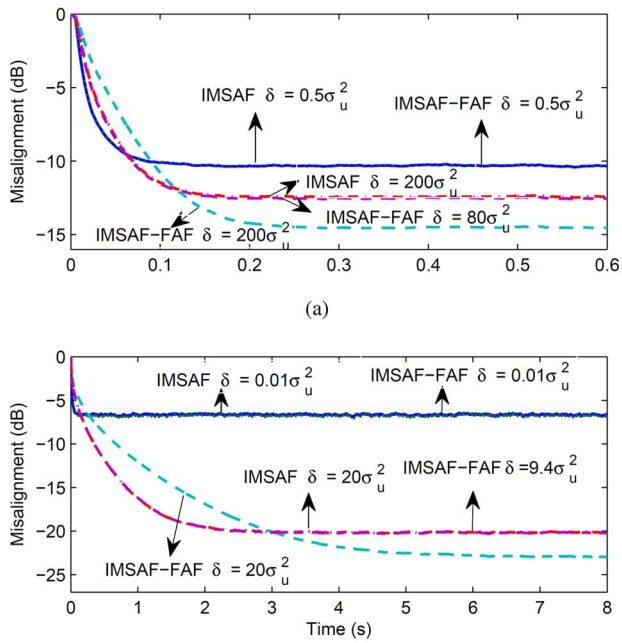


Fig. 3. Learning curves for the IMSAF-FAF and original IMSAF algorithms; $M = 256, \mu = 0.5, N = 4, P = 4$. (a) White noise as input, SNR = 10 dB. (b) AR(10) as input, SNR = 30 dB.

as the regularization parameter increases, the IMSAF and IMSAF-FAF can achieve smaller steady-state misalignment at the cost of slower convergence rate. Thus, the regularization parameter can play a role similar to that of the step size. If a variable regularization parameter is adopted, the algorithms can achieve both faster convergence and smaller steady-state misalignment.

B. Example 2: Comparison of the IMSAF and SIMSAF Algorithms

We now evaluate the performance of the SIMSAF. In fact, the performance difference between the SIMSAF and IMSAF

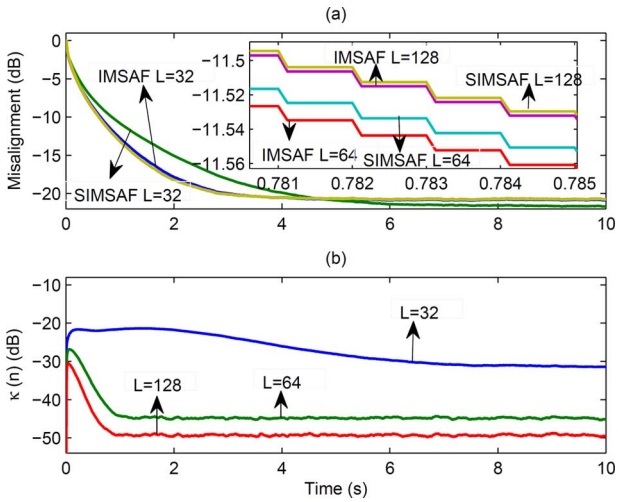


Fig. 4. Comparison of the IMSAF and SIMSAF algorithms with different length of the analysis filters; $M = 512, \mu = 0.2, N = 8, P = 4$, and SNR $= 30$ dB. (a) Learning curves for the IMSAF and SIMSAF algorithms with $L = 32, 64, 128$. (b) Euclidean norm of the difference of impulse responses from the IMSAF and SIMSAF algorithms.

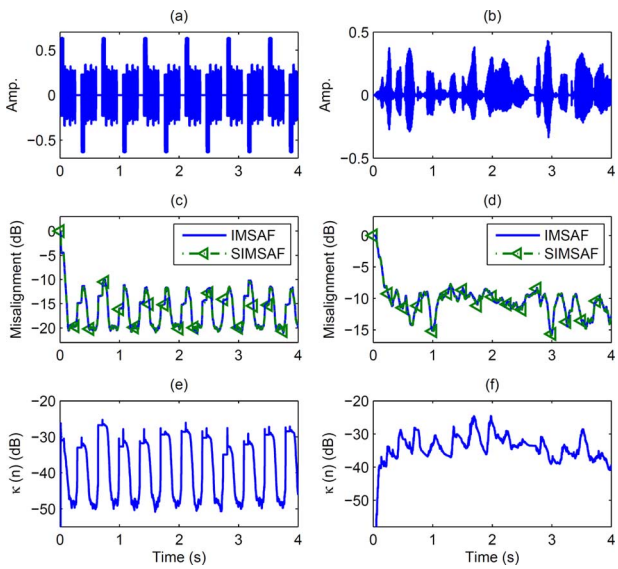


Fig. 5. Comparison of the IMSAF and SIMSAF algorithms; $M = 256, \mu = 0.5, N = 4, P = 4$, and SNR $= 20$ dB. (a) CSS. (b) Speech signal. Learning curves for the IMSAF and SIMSAF algorithms: (c) CSS as input, (d) speech as input. (e) and (f) are the Euclidean norm of the difference of impulse responses from the IMSAF and SIMSAF algorithms in (c) and (d).

depends mainly on the design quality of the analysis filters. If the analysis filter banks are "ideal", the SIMSAF and IMSAF should exhibit the same behavior. Fig. 4 shows the convergence performance of the IMSAF and SIMSAF algorithms with different length of the analysis filters. The parameters are $M = 512, \mu = 0.2, N = 8, P = 4$. The length of the analysis filters is set to $L = 32$, 64 and 128. The learning curves of the IMSAF and SIMSAF algorithms for the AR(10) process input are shown in Fig. 4(a). We also present the Euclidean norm of the difference of impulse responses in Fig. 4(b), defined as $\kappa(n) = 10\log_{10}\left[\|\mathbf{w}_{\mathrm{IMSAF}}(n) - \mathbf{w}_{\mathrm{SIMSAF}}(n)\|^2 / \|\mathbf{w}_o\|^2\right]$, where $\mathbf{w}_{\mathrm{IMSAF}}(n)$ and $\mathbf{w}_{\mathrm{SIMSAF}}(n)$ are the weight vectors of the IMSAF and SIMSAF algorithms, respectively. These two

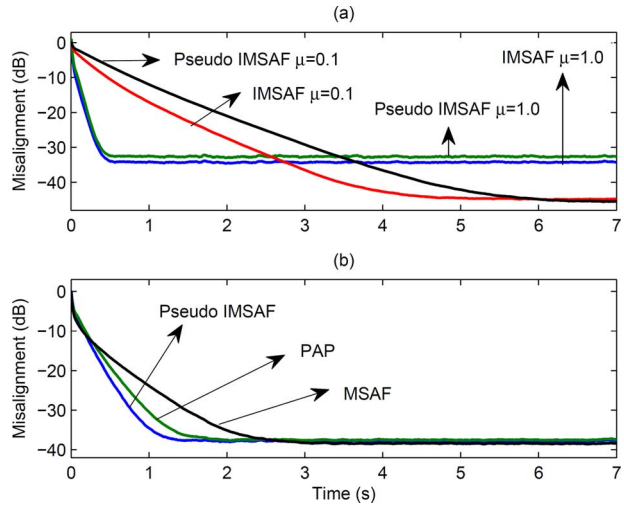


Fig. 6. Learning curves of the IMSAF, Pseudo IMSAF, PAP and MSAF algorithms with AR(8) as input; and SNR $= 40$ dB. (a) $N = 4, P = 2, \mu = 1.0, 0.1$; (b) $N = 2, P = 3, \mu = 1.0$.

algorithms differed significantly when $L = 32$. However, $\kappa(n)$ is very small and the convergence performance of the SIMSAF algorithm is very close to that of the IMSAF algorithm when $L = 64$ and $L = 128$.

We also carried out experiments with the composite source signal (CSS) [39] and speech as input signals. The parameters are $M = 256, N = 4, P = 4, L = 32, \mu = 0.5$, and SNR $= 20$ dB. The simulation results are displayed in Fig. 5. Once more, we see that the IMSAF and SIMSAF algorithms have almost the same performance. The length of analysis filters $L = 8N$ is enough to get good result. This verifies that the simplification in (45) does not deteriorate the system performance significantly if the analysis filter banks are properly designed.

C. Example 3: Comparison of the Pseudo IMSAF and IMSAF Algorithms

In this example, a simulation is conducted to evaluate the performance of pseudo IMSAF. The IMSAF, PAP, and MSAF algorithms are also included in the comparison. The input signal is an AR(8) process taken from [31]. The parameters are $M = 512$ and SNR $= 40$ dB. Fig. 6(a) presents the learning curves of IMSAF and pseudo IMSAF with $N = 4, P = 2$. Step sizes are $\mu = 1.0$ and $\mu = 0.1$, respectively. When the step size is 1.0, the pseudo IMSAF and original IMSAF exhibit the similar performance. Given a small step size $\mu = 0.1$, the pseudo IMSAF does not maintain a close approximation to the original IMSAF. Fig. 6(b) compares the convergence performance of the pseudo IMSAF, PAP and MSAF algorithms with $N = 2, P = 3$. It is observed the Pseudo IMSAF has faster convergence speed than the PAP and MSAF algorithms.

D. Example 4: Comparison of the IMSAF and AP Algorithms

Fig. 7 compares the performance of the IMSAF and AP algorithms with an AR(10) process as input at SNR $= 30$ dB. The length of the adaptive filter is $M = 1024$. The projection order of the AP algorithm is $P = 8$. Given $N = 2$, the IMSAF algorithm with $P = 5$ demonstrates the similar performance as

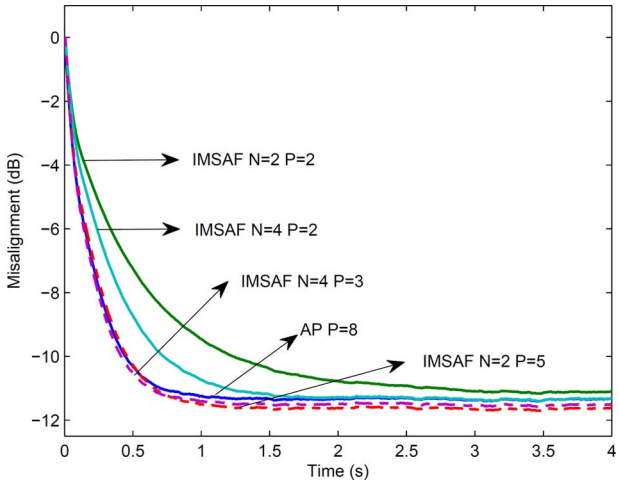


Fig. 7. Learning curves for the AP and IMSAF algorithms with an AR(10) process as input; $M = 1024$, $\mu = 1.0$, and SNR $= 30$ dB.

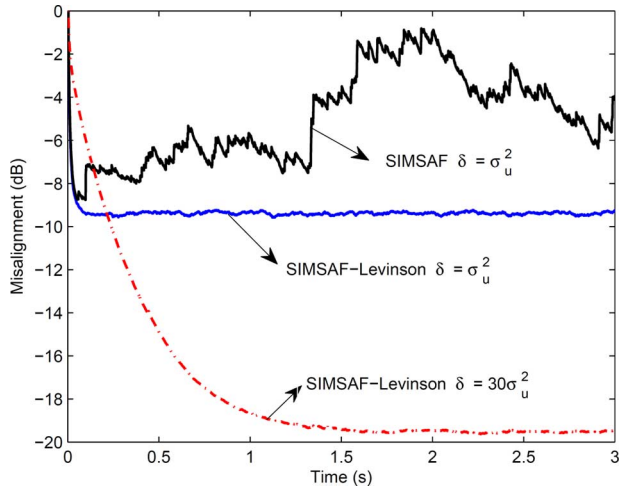


Fig. 8. Learning curves for SIMSAF and SIMSAF-Levinson algorithms with an AR(10) process as input; $M = 256$, $\mu = 0.9$, $N = 2$, $P = 6$, SNR $=$ 30 dB.

the AP algorithm. When $N = 4$, the SIMSAF algorithm with $P = 3$ is enough to provide the similar convergence rate. This finding demonstrates that the performance of the IMSAF algorithm with a smaller projection order approaches that of the AP algorithm with a larger projection order.

E. Example 5: Convergence Performance of the SIMSAF-Levinson Algorithm

Fig. 8 compares the convergence performance of SIMSAF and SIMSAF-Levinson algorithms. The input signal is the AR(10) process. Other parameters are $M = 256$, $\mu = 0.9$, and $P = 6$. When the regularization parameter is small $\delta = \sigma_u^2$, the learning curve of the SIMSAF-Levinson algorithm fluctuates strongly over time. We checked each of the learning curves and noted that this algorithm even diverges in some runs. A large regularization parameter $\delta = 30\sigma_u^2$ can ensure the stability of the SIMSAF-Levinson algorithm but lead to slower convergence rate. When $\mathbf{R}_{i,i}(k)$ is assumed to be a Toeplitz matrix, a large regularization parameter δ should be set to ensure the algorithm stability especially for highly colored signals. This requirement indicates that the SIMSAF-Levinson algorithm is less tolerant to the matrix $\mathbf{R}_{i,i}(k)$ being ill-conditioned. Similar results were also observed in the FAP algorithm [17]. Moreover, the complexity saved by the Levinson algorithm is insignificant at a small projection order, as argued previously. Thus, the use of the $\mathbf{LDL}^T$ approach may be preferable when the projection order is small $(P < 8)$.

F. Example 6: Comparison of the SIMSAF-CG and SIMSAF-DCD Algorithms

Fig. 9 shows the convergence performance of the SIMSAF-CG and SIMSAF-DCD algorithms with AR(10) process as input. We use $M = 1024$, $N = 4$, $P = 6$, and $M_b = 16$ in this case. The SNRs in Figs. 9(a)–9(c) are set to 30, 20, and 10 dB, respectively. Step sizes are chosen such that identical steady-state misalignment is achieved for all the algorithms. The performance of SIMSAF-CG and SIMSAF-DCD algorithms with $N_u = 6$ is similar to that of the standard IMSAF. Given a fixed N_u, the SIMSAF-CG algorithm converges faster than the SIMSAF-DCD algorithm does. The

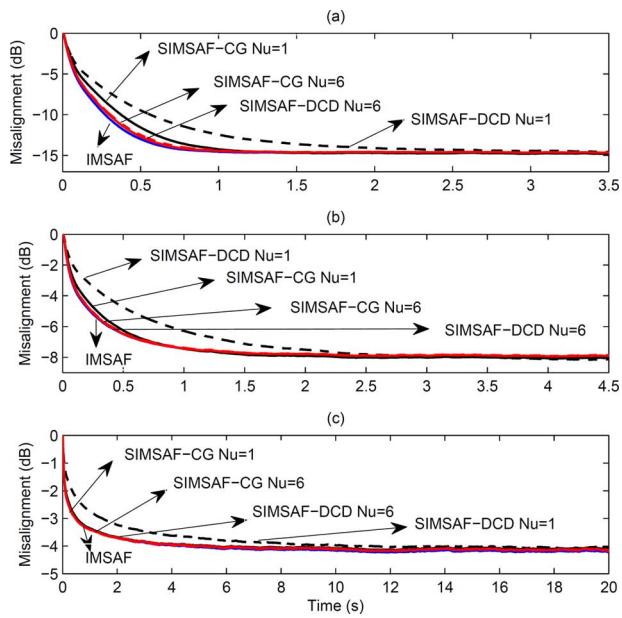


Fig. 9. Learning curves for the SIMSAF-CG and SIMSAF-DCD algorithms; $M = 1024, N = 4, P = 6, M_b = 16$. The excited signal is an AR(10) process. (a) SNR $= 30$ dB, (b) SNR $= 20$ dB, (c) SNR $= 10$ dB.

faster convergence of the CG algorithm is also demonstrated in [36]. However, the better convergence performance of the CG method is achieved at the cost of increased complexity. Moreover, the steady-state misalignments of the IMSAF, IMSAF-CG, and IMSAF-DCD algorithms increase as SNR decreases.

G. Example 7: Comparison With Other Fast Adaptive Filtering Algorithms

Simulations are conducted to compare the performance of the IMSAF with that of other fast adaptive algorithms having $O(M)$ complexity. The classical NLMS, FAP, and the two fast versions of RLS, i.e., QRD-based lattice filter (Algorithm 43.1 in [1]) and stabilized FTF algorithms [9] are involved in the comparison. The array lattice form is selected because it is the

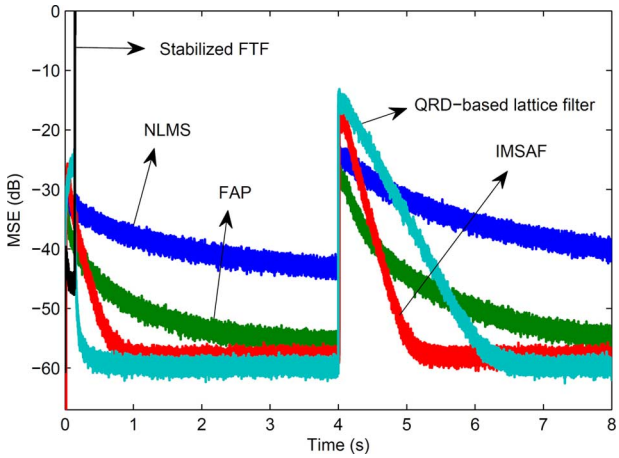


Fig. 10. MSE curves of the NLMS, FAP, IMSAF, QRD-based lattice filter and stabilized FTF algorithms with an AR(10) process as input; $M = 1024$, $\mathrm{SNR} = 40$ dB.

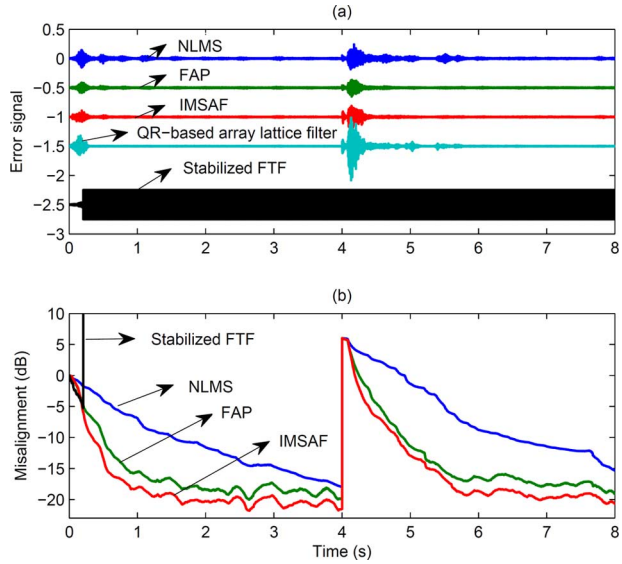


Fig. 11. Comparison of the NLMS, FAP, IMSAF, QRD-based lattice filter and stabilized FTF algorithms given a real speech signal as input at $\mathrm{SNR} = 30$ dB. (a) Error signals and (b) misalignments.

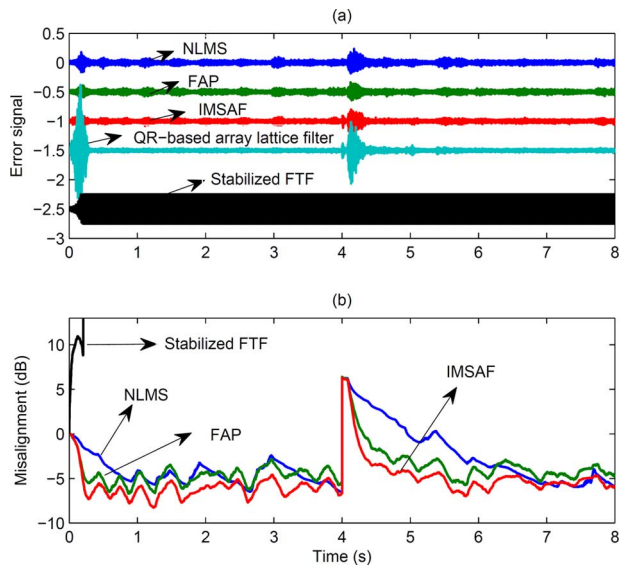


Fig. 12. Comparison of the NLMS, FAP, IMSAF, QRD-based lattice filter and stabilized FTF algorithms given a real speech signal as input at $\mathrm{SNR} = 15$ dB. (a) Error signals and (b) misalignments.

most reliable in terms of finite precision in all the order-recursive implementations of RLS [1]. The length of the adaptive filter is $M = 1024$. Other parameters are set as follows: NLMS ($\mu = 1.0$, $\delta = 10\sigma_u^2$), FAP ($\mu = 1.0$, $P = 8$, $\delta = 10\sigma_u^2$), IMSAF ($\mu = 0.3$, $N = 8$, $P = 8$, $\delta = 10\sigma_u^2$), stabilized FTF ($\lambda = 1 - 1/(3M)$, $\eta = 10^6$), and QRD-based lattice filter ($\lambda = 1 - 1/(3M)$, $\eta = 10^6$). A sudden change of the impulse response is introduced in the middle of the iterations. Fig. 10 shows the MSE curves with an AR(10) process as input at $\mathrm{SNR} = 40$ dB. The stabilized FTF diverges even in MATLAB precision. The QRD-based lattice filter exhibits the fastest convergence; however, its tracking rate is slower than the IMSAF which can limit the performance in nonstationary environments. The convergence speed of the IMSAF is lower than that of the QRD-based filter but faster than those of the NLMS and FAP algorithms. The NLMS, FAP, IMSAF-FEF2-$\mathbf{LDL}^T$, stabilized FTF and QRD-based lattice filter require 2048, 2208, 4754, 8192, and 23552 multiplication operations per sample, respectively. Thus, the complexity of IMSAF is much lower than those of the QRD-based lattice filter and the stabilized FTF. The proposed IMSAF achieves a good tradeoff between the convergence and complexity.

Figs. 11 and 12 present the comparison results of the five adaptive algorithms with a real speech signal as input at $\mathrm{SNR} =$ 30 and 15 dB, respectively. The other parameters used are similar to those in Fig. 10, except that a large regularization factor $\delta = 30\sigma_u^2$ is adopted for IMSAF. It is observed that a similar conclusion is reached regarding the performance of the five adaptive algorithms in a realistic SNR conditions using the speech signal.

VIII. Conclusions

This paper discusses the low-complexity implementation of the IMSAF algorithm in detail. We first proposed three approaches to reduce filtering complexity; these approaches can also be generalized to many adaptive filters that use similar filtering and update equations. Then, we presented the SIMSAF algorithm that facilitates the conversion of the matrix inversion operation with size $NP \times NP$ into N matrix inversion operations with size $P \times P$. The performance of the SIMSAF algorithm is close to that of the original IMSAF algorithm. Interestingly, the IMSAF algorithm adopts two prewhitening schemes to accelerate the convergence for colored signals. Based on this property, we can set up a new variant algorithm (pseudo IMSAF) that resembles the NLMS-type algorithm. The solution of the linear system of equations has a great influence on the overall performance of IMSAF and a proper method should be chosen according to the real application.

The complexity of the proposed several low-complexity IMSAF variants is only slightly higher than that of FAP but is considerably lower than that of RLS-type algorithms. The proposed fast IMSAF algorithms provide an alternative solution for many applications.

APPENDIX A
ON THE APPROXIMATION IN (28)

We now start to explore the effect of the approximation in (28) on the convergence performance. By disregarding $\mathbf{R}_{i,j}(k-1), i \neq j$ in the matrix $\mathbf{U}^T(k-1)\mathbf{U}(k-1)$, we can rewrite (25) as

$$\boldsymbol{\xi}_{i,\mathrm{D}}(k-1) = \{\mathbf{I} - \mu\mathbf{U}_i^T(k)\mathbf{U}_i(k-1) \times[\mathbf{U}_i^T(k-1)\mathbf{U}_i(k-1) + \delta\mathbf{I}]^{-1}\}\mathbf{e}_{i,\mathrm{D}}(k-1). \quad (59)$$

The matrix $\mathbf{U}_i^T(k-1)\mathbf{U}_i(k-1)$ has the following similarity decomposition

$$\mathbf{U}_i^T(k-1)\mathbf{U}_i(k-1) = \mathbf{G}^T\Lambda\mathbf{G} \quad (60)$$

where $\mathbf{G}$ is a unitary matrix with size $P \times P$, and $\Lambda = \mathrm{diag}\{\lambda_0, \lambda_1, \ldots, \lambda_{P-1}\}$ is the diagonal matrix with eigenvalue λ_i. Substituting (60) into (59) yields

$$\boldsymbol{\xi}_{i,\mathrm{D}}(k-1) = [\mathbf{I} - \mu\mathbf{G}^T\Lambda(\Lambda+\delta\mathbf{I})^{-1}\mathbf{G}]\mathbf{e}_{i,\mathrm{D}}(k-1). \quad (61)$$

As per (61), we can see that the ratio δ/λ_i determines the approximation accuracy of (28). To simplify the analysis, we assume that $\lambda_0 = \lambda_1 = \cdots = \lambda_{P-1} = \lambda$. We can rewrite (61) as

$$\boldsymbol{\xi}_{i,\mathrm{D}}(k-1) = (1 - \mu\frac{\lambda}{\lambda+\delta})\mathbf{e}_{i,\mathrm{D}}(k-1). \quad (62)$$

Substituting (62) into (28) results in

$$\mathbf{e}_{i,\mathrm{D}}(k) = \begin{bmatrix} d_{i,\mathrm{D}}(k) - \mathbf{u}_i^T(k)\mathbf{w}(k) \\ Q\boldsymbol{\xi}_{i,\mathrm{D}}(k-1) \end{bmatrix}, \quad (63)$$

where $Q = \frac{1-\mu}{1-\mu\lambda/(\lambda+\delta)}$. We now present two special cases to illustrate the difference between (63) and (28).

Case A: When $\lambda \gg \delta$, we have $Q \approx 1$. Thus, (63) becomes (22). That is, the approximation used in (28) exerts a weak effect on overall performance.

Case B: When λ is comparable to δ or $\lambda \ll \delta$, and then $Q \approx 1$ does not hold again. Defining the weight error vector as $\tilde{\mathbf{w}}(k) = \mathbf{w}_{\mathrm{o}} - \mathbf{w}(k)$, (63) can be expressed as

$$\mathbf{e}_{i,\mathrm{D}}(k) = \mathbf{B}\mathbf{U}_i^T(k)\tilde{\mathbf{w}}(k) + \mathbf{B}\mathbf{v}_{i,\mathrm{D}}(k) \quad (64)$$

where $\mathbf{B} = \mathrm{diag}[1, Q, \ldots, Q]$, and $\mathbf{v}_{i,\mathrm{D}}(k) = [v_{i,\mathrm{D}}(k), \ldots, v_{i,\mathrm{D}}(k-P+1)]^T$ ($v_{i,\mathrm{D}}(k)$ being the subband noise signal). Subtracting $\mathbf{w}_{\mathrm{o}}$ from both sides of (46) yields

$$\tilde{\mathbf{w}}(k+1) = \tilde{\mathbf{w}}(k) - \mu\sum_{i=0}^{N-1}\mathbf{U}_i(k)\tilde{\mathbf{R}}_{i,i}^{-1}(k)\mathbf{e}_{i,\mathrm{D}}(k). \quad (65)$$

Substituting (64) into (65) yields

$$\tilde{\mathbf{w}}(k+1) = [\mathbf{I} - \mu\sum_{i=0}^{N-1}\mathbf{U}_i(k)\tilde{\mathbf{R}}_{i,i}^{-1}(k)\mathbf{B}\mathbf{U}_i^T(k)]\tilde{\mathbf{w}}(k) - \mu\sum_{i=0}^{N-1}\mathbf{U}_i(k)\tilde{\mathbf{R}}_{i,i}^{-1}(k)\mathbf{B}\mathbf{v}_{i,\mathrm{D}}(k). \quad (66)$$

To compute $E[\|\tilde{\mathbf{w}}(k)\|^2]$, we use the following assumptions: A.1) $\tilde{\mathbf{w}}(k)$ is independent with $\mathbf{U}_i(k)\mathbf{U}_i^T(k)$, A.2) $u(n)$ and $v(n)$ are the independent and identically distributed signals. Using A.1 and ignoring the correlation between different subband signals, the expected squared Euclidean of both sides of (66) is

$$E[\|\tilde{\mathbf{w}}(k+1)\|^2] = E[\tilde{\mathbf{w}}^T(k)E[\mathbf{F}(k)]\tilde{\mathbf{w}}(k)] + \mu^2\sum_{i=0}^{N-1}E[\mathbf{v}_{i,\mathrm{D}}^T(k)\mathbf{B}E[\mathbf{H}(k)]\mathbf{B}\mathbf{v}_{i,\mathrm{D}}(k)] \quad (67)$$

where

$$E[\mathbf{F}(k)] = E[\mathbf{I} - 2\mu\sum_{i=0}^{N-1}\mathbf{U}_i(k)\mathbf{B}\tilde{\mathbf{R}}_{i,i}^{-1}(k)\mathbf{U}_i^T(k)] + \mu^2\sum_{i=0}^{N-1}\mathbf{U}_i(k)\mathbf{B}[\tilde{\mathbf{R}}_{i,i}^{-1}(k) - \delta\tilde{\mathbf{R}}_{i,i}^{-2}(k)]\mathbf{B}\mathbf{U}_i^T(k)], \quad (68)$$

$$E[\mathbf{H}(k)] = E[\tilde{\mathbf{R}}_{i,i}^{-1}(k)\mathbf{U}_i^T(k)\mathbf{U}_i(k)\tilde{\mathbf{R}}_{i,i}^{-1}(k)]. \quad (69)$$

By using A.2, we obtain the approximation $E[\tilde{\mathbf{R}}_{i,i}^{-1}(k)] = (M\sigma_{su}^2 + \delta)^{-1}\mathbf{I}$ and $E[\tilde{\mathbf{R}}_{i,i}^{-2}(k)] = (M\sigma_{su}^2 + \delta)^{-2}\mathbf{I}$, where σ_{su}^2 is the variance of the subband input signal. Thus, we get

$$E[\mathbf{F}(k)] = [1 - \alpha_r(2\beta_1 - \alpha_r\beta_2)N/M]\mathbf{I} \triangleq \vartheta\mathbf{I}, \quad (70)$$

$$E[\mathbf{H}(k)] = \alpha_r^2/(M\mu^2\sigma_{su}^2)\mathbf{I}, \quad (71)$$

where σ_{sv}^2 is the variance of the subband noise signal and $\beta_1 = 1 + (P-1)Q$, $\beta_2 = 1 + (P-1)Q^2$, $\alpha_r = \mu M\sigma_{su}^2/(M\sigma_{su}^2 + \delta)$. Using the results above, we get a recursive expression for $E[\|\tilde{\mathbf{w}}(k)\|^2]$

$$E[\|\tilde{\mathbf{w}}(k+1)\|^2] = [1 - \alpha_r(2\beta_1 - \alpha_r\beta_2)N/M]E[\|\tilde{\mathbf{w}}(k)\|^2] + N\beta_2\alpha_r^2\sigma_{sv}^2/(M\sigma_{su}^2). \quad (72)$$

Assuming the algorithm convergence, one can write $\lim_{k\to\infty}E[\|\tilde{\mathbf{w}}(k+1)\|^2] = \lim_{k\to\infty}E[\|\tilde{\mathbf{w}}(k)\|^2]$. From (72), we get the steady-state misalignment:

$$E[\|\tilde{\mathbf{w}}(\infty)\|^2] = \frac{\alpha_r\sigma_{sv}^2}{(2\beta_1/\beta_2 - \alpha_r)\sigma_{su}^2}. \quad (73)$$

We can compare the performance of IMSAF and IMSAF-FAF algorithms based on (72) and (73). For IMSAF, we have $Q = 1$, thus $\beta_1 = \beta_2 = P$. For IMSAF-FAF, we have $0 < Q < 1$, thus $\beta_2 < \beta_1 < P$. From (72), it is seen that ϑ controls the algorithm convergence and

$$\vartheta_{\mathrm{IMSAF}} = 1 - \alpha_r(2 - \alpha_r)PN/M, \quad (74)$$

$$\vartheta_{\mathrm{IMSAF-FAF}} = 1 - \alpha_r(2\beta_1 - \alpha_r\beta_2)N/M. \quad (75)$$

It is easy to verify that $\vartheta_{\mathrm{IMSAF}} < \vartheta_{\mathrm{IMSAF-FAF}}$. From (73), we obtain

$$E[\|\tilde{\mathbf{w}}(\infty)\|^2]_{\mathrm{IMSAF}} = \frac{\sigma_{sv}^2\alpha_r}{\sigma_{su}^2(2 - \alpha_r)}, \quad (76)$$

$$E[\|\tilde{\mathbf{w}}(\infty)\|^2]_{\mathrm{IMSAF-FAF}} = \frac{\sigma_{sv}^2\alpha_r}{\sigma_{su}^2(2\beta_1/\beta_2 - \alpha_r)}. \quad (77)$$

Since $\beta_2 < \beta_1$ holds, we get $E[\|\tilde{\mathbf{w}}(\infty)\|^2]_{\mathrm{IMSAF-FAF}} < E[\|\tilde{\mathbf{w}}(\infty)\|^2]_{\mathrm{IMSAF}}$. Therefore, the IMSAF-FAF algorithm can achieve smaller steady-state misalignment but also slower convergence rate than the IMSAF algorithm.

Note that the analysis is based on a relatively simple signal mode, and it is worthwhile to study the more general case where the matrix $\mathbf{U}_i^T(k)\mathbf{U}_i(k)$ has an arbitrary set of eigenvalues.

APPENDIX B
DERIVATION OF (47)

In the special case wherein $\mu = 1.0$ and $\delta = 0$, (28) becomes

$$\mathbf{e}_{i,\mathrm{D}}(k) = \begin{bmatrix} e_{i,\mathrm{D}}(k) \\ \mathbf{0} \end{bmatrix}. \tag{78}$$

Using (7) and (49), $\mathbf{U}_i^T(k)\mathbf{U}_i(k)$ can be expressed in a block matrix form

$$\mathbf{U}_i^T(k)\mathbf{U}_i(k) = \begin{bmatrix} \mathbf{u}_i^T(k)\mathbf{u}_i(k) & \mathbf{u}_i^T(k)\mathbf{D}_i(k) \\ \mathbf{D}_i^T(k)\mathbf{u}_i(k) & \mathbf{D}_i^T(k)\mathbf{D}_i(k) \end{bmatrix}. \tag{79}$$

An inverse matrix can be calculated from its parts by [40]

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{F}_{11}^{-1} & -\mathbf{F}_{11}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{F}_{11}^{-1} & \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{F}_{11}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \end{bmatrix} \tag{80}$$

where $\mathbf{F}_{11} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$. Using (79) and (80), we can express the inversion of $\mathbf{U}_i^T(k)\mathbf{U}_i(k)$ as

$$[\mathbf{U}_i^T(k)\mathbf{U}_i(k)]^{-1} = \frac{1}{\Omega_1}\begin{bmatrix} 1 & \mathbf{\Omega}_2 \\ \mathbf{\Omega}_3 & \mathbf{\Omega}_4 \end{bmatrix} \tag{81}$$

where

$$\Omega_1 = \mathbf{u}_i^T(k)\mathbf{u}_i(k) - \mathbf{u}_i^T(k)\mathbf{D}_i(k) \times \left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1}\mathbf{D}_i^T(k)\mathbf{u}_i(k), \tag{82}$$

$$\mathbf{\Omega}_2 = -\mathbf{u}_i^T(k)\mathbf{D}_i(k)\left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1}, \tag{83}$$

$$\mathbf{\Omega}_3 = -\left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1}\mathbf{D}_i^T(k)\mathbf{u}_i(k), \tag{84}$$

$$\mathbf{\Omega}_4 = \Omega_1\left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1} + \left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1} \times \mathbf{D}_i^T(k)\mathbf{u}_i(k)\mathbf{u}_i^T(k)\mathbf{D}_i(k)\left[\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right]^{-1}. \tag{85}$$

Using (50) and (84), we obtain:

$$\mathbf{\Omega}_3 = -\mathbf{a}_i(k). \tag{86}$$

By applying (78), (81) and (86), we have

$$\begin{aligned} &\mathbf{U}_i(k)[\mathbf{U}_i^T(k)\mathbf{U}_i(k)]^{-1}\mathbf{e}_{i,\mathrm{D}}(k) \\ &= [\mathbf{u}_i(k) + \mathbf{D}_i(k)\mathbf{\Omega}_3]e_{i,\mathrm{D}}(k)/\Omega_1 \\ &= [\mathbf{u}_i(k) - \mathbf{D}_i(k)\mathbf{a}_i(k)]e_{i,\mathrm{D}}(k)/\Omega_1 \\ &= \mathbf{\Phi}_i(k)e_{i,\mathrm{D}}(k)/\Omega_1. \end{aligned} \tag{87}$$

Substituting (50) into (48), $\mathbf{\Phi}_i(k)$ can be rewritten as

$$\mathbf{\Phi}_i(k) = [\mathbf{I} - \mathbf{D}_i(k)\left(\mathbf{D}_i^T(k)\mathbf{D}_i(k)\right)^{-1}\mathbf{D}_i^T(k)]\mathbf{u}_i(k). \tag{88}$$

By evaluating the energies of both sides of (88), it follows

$$\begin{aligned} &\mathbf{\Phi}_i^T(k)\mathbf{\Phi}_i(k) \\ &= \mathbf{u}_i^T(k)[\mathbf{I} - \mathbf{D}_i(k)(\mathbf{D}_i^T(k)\mathbf{D}_i(k))^{-1}\mathbf{D}_i^T(k)]\mathbf{u}_i(k) \\ &= \mathbf{u}_i^T(k)\mathbf{\Phi}_i(k) \\ &= \Omega_1. \end{aligned} \tag{89}$$

Substituting (89) into (87) yields

$$\mathbf{U}_i(k)\left[\mathbf{U}_i^T(k)\mathbf{U}_i(k)\right]^{-1}\mathbf{e}_{i,\mathrm{D}}(k) = \frac{\mathbf{\Phi}_i(k)e_{i,\mathrm{D}}(k)}{\mathbf{\Phi}_i^T(k)\mathbf{\Phi}_i(k)}. \tag{90}$$

Substituting (90) into (46), (47) is obtained. End of proof.

ACKNOWLEDGMENT

We would like to thank the associate editor and anonymous reviewers for their constructive suggestions and comments, which improved the presentation of this paper.

REFERENCES

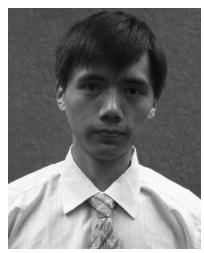
[1] A. H. Sayed, *Adaptive Filters*. New York: Wiley, 2008.
[2] K. A. Lee, W. S. Gan, and S. M. Kuo, *Subband Adaptive Filtering: Theory and Implementation*. Chichester, U.K.: Wiley, 2009.
[3] M. D. Courville and P. Duhamel, "Adaptive filtering in subbands using a weighted criterion," *IEEE Trans. Signal Process.*, vol. 46, no. 9, pp. 2359–2371, Sep. 1998.
[4] S. S. Pradhan and V. E. Reddy, "A new approach to subband adaptive filtering," *IEEE Trans. Signal Process.*, vol. 47, no. 3, pp. 655–664, Mar. 1999.
[5] K. A. Lee and W. S. Gan, "Improving convergence of the NLMS algorithm using constrained subband updates," *IEEE Signal Process. Lett.*, vol. 11, no. 9, pp. 736–739, Sep. 2004.
[6] V. DeBrunner, L. DeBrunner, and L. Wang, "Sub-band adaptive filtering with delay compensation for active control," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2932–2937, Oct. 2004.
[7] F. Yang, M. Wu, P. Ji, and J. Yang, "An improved multiband-structured subband adaptive filter algorithm," *IEEE Signal Process. Lett.*, vol. 19, no. 10, pp. 647–650, Oct. 2012.
[8] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn.*, vol. 67-A, no. 5, pp. 19–27, May 1984.
[9] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Process.*, vol. 39, no. 1, pp. 92–113, Jan. 1991.
[10] S. L. Gay and S. Tavathia, "The fast affine projection algorithm," in *Proc. ICASSP*, May 1995, pp. 3023–3026.
[11] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "A fast projection algorithm for adaptive filtering," *IEICE Trans. Fundam.*, vol. E78-A, no. 10, pp. 1355–1361, Oct. 1995.
[12] G. Rombouts and M. Moonen, "A sparse block exact affine projection algorithm," *IEEE Trans. Speech, Audio Process*, vol. 10, no. 2, pp. 100–108, Feb. 2002.
[13] Q. G. Liu, B. Champagne, and K. C. Ho, "On the use of a modified fast affine projection algorithm in subbands for acoustic echo cancellation," in *Proc. IEEE Digit. Signal Process. Workshop*, Sep. 1996, pp. 354–357.
[14] S. Oh, D. Linebarger, B. Priest, and B. Raghothaman, "A fast affine projection algorithm for an acoustic echo cancellation using a fixed-point DSP processor," in *Proc. ICASSP*, Nov. 1997, pp. 4121–4124.
[15] F. Albu, J. Kadlec, N. Coleman, and A. Fagan, "The gauss-seidel fast affine projection algorithm," in *Proc. SIPS*, Oct. 2002, pp. 109–114.
[16] Y. Zakharov and F. Albu, "Coordinate descent iterations in fast affine projection algorithm," *IEEE Signal Process. Lett.*, vol. 12, no. 5, pp. 353–356, May 2005.
[17] H. Ding, "Fast affine projection adaptation algorithms with stable and robust symmetric linear system solvers," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 1730–1740, May 2007.
[18] K. Chen, J. Lu, X. Qiu, and B. Xu, "Stability improvement of relaxed FAP algorithm," *Electron. Lett.*, vol. 43, no. 20, pp. 1119–1121, Sep. 2007.
[19] Y. Zakharov, "Low complexity implementation of the affine projection algorithm," *IEEE Signal Process. Lett.*, vol. 15, pp. 557–560, 2008.
[20] M. C. Tsakiris and P. A. Naylor, "Fast exact affine projection algorithm using displacement structure theory," in *Proc. DSP 2009*, Nov. 2009, pp. 2–5.
[21] R. Merched and A. H. Sayed, "RLS-laguerre lattice adaptive filtering: Error-feedback, normalized and array-based algorithms," *IEEE Trans. Signal Process.*, vol. 42, no. 4, pp. 2565–2576, Nov. 2001.
[22] R. Merched and A. H. Sayed, "Extended fast fixed order RLS adaptive filtering," *IEEE Trans. Signal Process.*, vol. 49, no. 12, pp. 3015–3031, Dec. 2001.
[23] R. Merched, "Extended RLS-lattice adaptive filters," *IEEE Trans. Signal Process.*, vol. 51, no. 9, pp. 2294–2309, Sep. 2003.

[24] R. Merched, "A unified approach to structured covariances: fast generalized sliding window RLS recursions for arbitrary basis," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 6060–6075, Dec. 2013.
[25] J. Benesty, C. Paleologu, and S. Ciochina, "On regularization in adaptive filtering," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 6, pp. 1734–1742, Sep. 2011.
[26] F. Yang, M. Wu, J. Yang, and Z. Kuang, "A fast exact filtering approach to a family of affine projection-type algorithms," *Signal Process.*, vol. 101, pp. 1–10, 2014.
[27] F. Yang and J. Yang, "Fast implementation of a family of memory proportionate affine projection algorithm," in *Proc. IEEE ICASSP*, Apr. 2015, pp. 3507–3511.
[28] S. K. Mitra, *Digital Signal Processing: A Computer Based Approach*. New York, NY, USA: McGraw-Hill, 2006.
[29] M. Rupp, "A family of adaptive filter algorithms with decorrelating properties," *IEEE Trans. Signal Process.*, vol. 46, no. 3, pp. 771–775, Mar. 1998.
[30] S. Rao and W. A. Pearlman, "Analysis of linear prediction, coding, and spectral estimation from subbands," *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1160–1178, Jul. 1996.
[31] S. J. M. de Almeida, J. C. M. Bermudez, and N. J. Bershad, "A stochastic model for a pseudo affine projection algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 107–118, Jan. 2009.
[32] F. Bouteille, P. Scalart, and M. Corazza, "Pseudo affine projection algorithm new solution for adaptive identification," in *Proc. Eurospeech*, Sep. 1999, pp. 427–430.
[33] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[34] G. K. Boray and M. D. Srinath, "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuits Syst. I*, vol. 39, no. 1, pp. 1–10, Jan. 1992.
[35] J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon Univ, Pittsburgh, PA, USA, Tech. Rep., 1994.
[36] Y. Zakharov, G. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3150–3161, Jul. 2008.
[37] Y. V. Zakharov and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem," *Electron. Lett.*, vol. 40, no. 9, pp. 567–569, 2004.
[38] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 11, pp. 2425–2438, Nov. 2009.
[39] Digital Network Echo Cancellers, ITU-T G.168 Jun. 2002.
[40] C. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.

Feiran Yang (M'14) received the B.S. degree in Electrical Engineering from the Shandong University, Jinan, China, in 2005, the M.S. degree from Southeast University, Nanjing, China, in 2008, and the Ph.D. degree from the Institute of Acoustics, Chinese Academy of Sciences (IACAS), Beijing, China, in 2013. From March 2008 to June 2010, He was with Fortemedia Inc. first as a DSP engineer and then as a senior R&D engineer, where he was engaged in dual-microphone based speech enhancement, echo cancellation and algorithms for finite precision implementations. Currently, he is an assistant professor in IACAS. His research interests are adaptive filtering, echo control, speech enhancement, acoustic feedback cancellation, and 3-D audio system.

Dr. Yang is the recipient of the President's Award from the Chinese Academy of Sciences in 2013.

Ming Wu (M'13) graduated with a degree in electronics from Nanjing University, Nanjing, China, in 2002 and received the Ph.D. degree from Nanjing University in 2007 with a dissertation on active noise control. He was with the Institute of Acoustics, Nanjing University, as a Postdoctoral Researcher from 2007 to 2008. He has been with the Institute of Acoustics, Chinese Academy of Sciences (IACAS), Beijing, as an Associate Professor since 2008. His main research areas include noise and vibration control, electroacoustics, and audio signal processing.



Peifeng Ji (M'14) received the B. Eng. and M. Eng. degrees from Shandong University of Science and Technology, Qingdao, China, and the Ph.D. degree from Institute of Acoustics, Chinese Academy of Sciences (IACAS), Beijing, China, in 2002, 2005, 2008, respectively. From May 2008 to Nov 2010, he was with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as a research fellow. Since Nov 2010, he has been an associate professor at IACAS. His research interests are mainly in acoustic signal processing, musical acoustics and nonlinear acoustics.

Jun Yang (M'99–SM'04) received the B. Eng. and M. Eng. degrees from Harbin Engineering University, Harbin, China, and the Ph.D. degree in acoustics from Nanjing University, Nanjing, China, in 1990, 1993, and 1996, respectively. From 1996 to 1998, he was a Postdoctoral Fellow at the Institute of Acoustics, Chinese Academy of Sciences (IACAS), Beijing, China. From October 1998 to April 1999, he was with Hong Kong Polytechnic University as a Visiting Scholar. From Jan. 1997 to May 1999, he was with IACAS as an Associate Professor. He joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as a Research Fellow, a Teaching Fellow, Assistant Professor, and Associate Professor in 1999, 2001, 2003, and 2005, respectively. Since Nov. 2003, he has been a Professor at IACAS. Currently, he is the Director of the Key Laboratory of Noise and Vibration Research, Institute of Acoustics, Chinese Academy of Sciences. His main areas of research interests include communication acoustics, 3-D audio systems, acoustic signal processing, sound field control, and nonlinear acoustics.