Signal Processing 101 (2014) 1–10

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

A fast exact filtering approach to a family of affine projection-type algorithms

Feiran Yang, Ming Wu, Jun Yang*, Zheng Kuang

Key Laboratory of Noise and Vibration Research, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 16 October 2013
Received in revised form
6 January 2014
Accepted 30 January 2014
Available online 7 February 2014

Keywords:
Adaptive filter
Affine projection
Fast exact filtering
Low complexity

ABSTRACT

The affine projection (AP)-type algorithms produce a good tradeoff between convergence speed and complexity. As the projection order increases, the convergence rate of the AP algorithm is improved at a relatively high complexity. Many efforts have been made to reduce the complexity. However, most of the efficient versions of the AP-type algorithms are based on the fast approximate filtering (FAF) scheme originally proposed in the fast AP (FAP) algorithm. The approximation leads to degraded convergence performance. Recently, a fast exact filtering (FEF) AP (FEAP) algorithm was proposed by Y. Zakharov. In this paper, we propose a new FEF approach to further reduce the complexity of the FEAP algorithm given that the calculation of the weight vector is not the primary objective for the application at hand. The proposed FEF scheme is then extended to the dichotomous coordinate descent (DCD)-AP, affine projection sign (APS), and modified filtered-x affine projection (MFxAP) algorithms. The complexity of AP-type algorithms based on the proposed FEF approach is comparable to that based on the FAF scheme. Moreover, analysis results show that the complexity reduction of the new algorithms is achieved without any performance degradation.

1. Introduction

In adaptive filtering, the least-mean-square (LMS)-type algorithms are widely used but suffer from slow convergence for colored signals. The affine projection (AP) algorithm [1] was proposed to speed up the convergence, which produces a good tradeoff between the convergence speed and the complexity. Due to the good properties, several variants of the AP algorithm have been developed in the context of blind multiuser detection [2], acoustic echo cancellation (AEC) [3,4], active noise control (ANC) [5], and acoustic feedback cancellation (AFC) [6].

When the projection order P increases, the convergence rate of the AP algorithm is improved at the price of a considerable rise of the computational complexity. Many efforts have been made to reduce the complexity of the AP algorithm [3–20].

The complexity of the direct calculation of the error vector is proportional to the projection order. The fast AP (FAP) [3,4] algorithm and its variants [5–16] present a fast approximate filtering (FAF) scheme to reduce the complexity. Since the FAP algorithm is based on an implicit “small regularization parameter” assumption, the FAP algorithm is not exactly equal to the standard AP algorithm [17]. Most of the existing fast AP-type algorithms [3–16] are based on the FAF approach. The FAF approach reduces the complexity efficiently but also leads to degraded performance.

To overcome this limitation, a fast exact filtering (FEF) approach to the AP algorithm (FEAP) [18] was presented by Y. Zakharov. However, in the FEAP algorithm, calculation of the error vector requires the update of the weight vector explicitly that provides the largest contribution towards the

* Corresponding author.
E-mail address: jyang@mail.ioa.ac.cn (J. Yang).

http://dx.doi.org/10.1016/j.sigpro.2014.01.030
0165-1684 © 2014 Elsevier B.V. All rights reserved.

Table 1
FAP algorithm [3,4].

Equation	$\times$	$+$
$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	$2P$	$2P$
$\boldsymbol{\alpha}(n)=\mathbf{X}^T(n-1)\mathbf{x}(n)$	0	0
$e(n)=d(n)-\mathbf{x}^T(n)\hat{\mathbf{w}}(n-2)-\boldsymbol{\alpha}^T(n)\boldsymbol{\varphi}(n-1)$	$L+P$	$L+P$
$\mathbf{e}(n)=\begin{bmatrix} e(n) \\ (1-\mu)\overline{\mathbf{e}}(n-1)\end{bmatrix}$	$P-1$	0
$\boldsymbol{\varepsilon}(n)=\mu[\mathbf{R}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n)$	P_m	P_a
$\boldsymbol{\varphi}(n)=\boldsymbol{\varepsilon}(n)+\begin{bmatrix} 0 \\ \overline{\boldsymbol{\varphi}}(n-1)\end{bmatrix}$	0	$P-1$
$\hat{\mathbf{w}}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{x}(n-P+1)\varphi_{P-1}(n)$	L	L
Total $2L+4P+P_m$ multiplications $2L+4P+P_a$ additions		

algorithm complexity. Many AP-type algorithms [19–21] adopt the FEF approach to reduce the complexity and have a similar problem. In many applications such as AEC and ANC, calculation of the weight vector is not the main concern [3–15]. In this paper, we will extend the work in [18] and propose an enhanced FEF approach to the AP (EFEAP) algorithm. The complexity of the proposed EFEAP algorithm is comparable to that of the FAP algorithm. We then extend the proposed FEF approach to a family of AP-type algorithms such as the dichotomous coordinate descent (DCD)-AP [18], affine projection sign (APS) [22], and modified filtered-x affine projection (MFxAP) algorithms [14,15]. Computer simulations demonstrate the effectiveness of the proposed approach.

Notations: throughout this paper, we use uppercase and lowercase bold fonts to denote matrices and vectors, respectively, e.g., $\mathbf{R}$ and $\mathbf{r}$. Superscript T denotes the transpose operator, and $\mathbf{I}$ is the $P\times P$ identity matrix.

2. Proposed FEF approach to the AP algorithm

Consider the desired response $d(n)$ arising from the linear model

$$d(n)=\mathbf{w}_o^T\mathbf{x}(n)+v(n) \tag{1}$$

where $\mathbf{w}_o=[w_0,w_1,\ldots,w_{L-1}]^T$ is the L-length weight vector of the unknown system, $\mathbf{x}(n)=[x(n),x(n-1),\ldots,x(n-L+1)]^T$ denotes the input signal vector, and $v(n)$ represents the system noise.

The adaptive weight vector is $\mathbf{w}(n)=[w_0(n),w_1(n),\ldots,w_{L-1}(n)]^T$. To describe the AP algorithm, we define the input signal, the desired signal, the filtered-out, and the error vectors as follows:

$$\mathbf{X}(n)=[\mathbf{x}(n),\mathbf{x}(n-1),\ldots,\mathbf{x}(n-P+1)] \tag{2}$$

$$\mathbf{d}(n)=[d(n),d(n-1),\ldots,d(n-P+1)]^T \tag{3}$$

$$\begin{aligned}\mathbf{y}(n)&=[y_0(n),y_1(n),\ldots,y_{P-1}(n)]^T\\&=\mathbf{X}^T(n)\mathbf{w}(n-1)\end{aligned} \tag{4}$$

$$\begin{aligned}\mathbf{e}(n)&=[e_0(n),e_1(n),\ldots,e_{P-1}(n)]^T\\&=\mathbf{d}(n)-\mathbf{y}(n).\end{aligned} \tag{5}$$

The update equation of the AP algorithm is

$$\begin{aligned}\boldsymbol{\varepsilon}(n)&=[\varepsilon_0(n),\varepsilon_1(n),\ldots,\varepsilon_{P-1}(n)]^T\\&=\mu[\mathbf{X}^T(n)\mathbf{X}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n)\end{aligned} \tag{6}$$

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\mathbf{X}(n)\boldsymbol{\varepsilon}(n) \tag{7}$$

where μ is the step size, and δ is a regularization parameter.

The complexity of the AP algorithm is mainly due to the following three operations: (i) calculation of the filtered-out vector $\mathbf{y}(n)$ in (4), (ii) update of the weight vector $\mathbf{w}(n)$ in (7), and (iii) the matrix inversion operation in (6). For a direct implementation, the first two steps need $2PL$ operations per sample, which is very expensive especially for a long impulse response. We now briefly review the state-of-the-art fast filtering approaches.

2.1. FAF approach

The FAP algorithm [3,4] updates the error vector $\mathbf{e}(n)$ via the following approximation

$$\mathbf{e}(n)\approx\begin{bmatrix} d(n)-\mathbf{x}^T(n)\mathbf{w}(n-1) \\ (1-\mu)\overline{\mathbf{e}}(n-1)\end{bmatrix} \tag{8}$$

where $\overline{\mathbf{e}}(n-1)$ consists of the $P-1$ upper elements of $\mathbf{e}(n-1)$. For Clarity, we present the FAP algorithm in Table 1, where the definitions of $\hat{\mathbf{w}}(n)$ and $\boldsymbol{\varphi}(n)$ can be found in (13) and (14). The only difference among many variants of FAP algorithm is the calculation of the linear system of equations. We assume that solving $[\mathbf{R}(n)+\delta\mathbf{I}]\boldsymbol{\varepsilon}(n)=\mu\mathbf{e}(n)$ requires P_m multiplications and P_a additions.

Using (8), the complexity of the filtering step reduces from $O(PL)$ operations in (4) to $O(L)$ operations. But (8) is only an approximate implementation of (5) under the condition that δ is significantly smaller than the eigenvalue of the matrix $\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$ [4,17]. When the regularization parameter δ is large, the approximation in (8) can cause discrepancy between the FAP and AP algorithms. Indeed, the regularization parameter can vary from very small to very large, depending on the level of the additive noise [23]. Thus, the assumption used in the derivation of (8) has its shortcomings [17].

2.2. FEF approach

In the FAP algorithm, only the first component of the error vector is calculated, and the others are approximated. More recently, a low-complexity FEAP algorithm [18] was proposed where all the error vector components can be exactly calculated. The basic idea is summarized as follows.

Substituting (7) into (4), one has

$$\begin{aligned}\mathbf{y}(n)&=\mathbf{X}^T(n)\mathbf{w}(n-1)\\&=\mathbf{z}(n)+\mathbf{G}(n)\boldsymbol{\varepsilon}(n-1)\end{aligned} \tag{9}$$

where $\mathbf{G}(n)=\mathbf{X}^T(n)\mathbf{X}(n-1)$ and $\mathbf{z}(n)=\mathbf{X}^T(n)\mathbf{w}(n-2)$. Taking (4) into account, $\mathbf{z}(n)$ can be expressed as

$$\begin{aligned}\mathbf{z}(n)&=\mathbf{X}^T(n)\mathbf{w}(n-2)\\&=[z_0(n),y_0(n-1),\ldots,y_{P-2}(n-1)]^T\end{aligned} \tag{10}$$

Table 2
FEAP algorithm [18].

Equation	$\times$	$+$
$\mathbf{G}(n)=\mathbf{X}^T(n)\mathbf{X}(n-1)$	$2P$	$2P$
$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	2	2
$z_0(n)=\mathbf{x}^T(n)\mathbf{w}(n-2)$	L	L
$\mathbf{z}(n)=[z_0(n), y_0(n-1), \ldots, y_{P-2}(n-1)]^T$	0	0
$\mathbf{y}(n)=\mathbf{z}(n)+\mathbf{G}(n)\boldsymbol{\varepsilon}(n-1)$	P^2	P^2
$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{y}(n)$	0	P
$\boldsymbol{\varepsilon}(n)=\mu[\mathbf{R}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n)$	P_m	P_a
$\mathbf{w}(n)=\mathbf{w}(n-1)+\mathbf{X}(n)\boldsymbol{\varepsilon}(n)$	PL	PL
Total		
$(P+1)L+P^2+2P+P_m+2$ multiplications		
$(P+1)L+P^2+3P+P_a+2$ additions		

where

$$z_0(n)=\mathbf{x}^T(n)\mathbf{w}(n-2). \tag{11}$$

Calculation of the elements in $\mathbf{R}(n)$ and $\mathbf{G}(n)$ requires $2P+2$ multiplications and $2P+2$ additions using a recursive technique.

The FEAP algorithm presented in Table 2 is attractive because it provides an exact filtering. However, the calculation of $z_0(n)$ requires the update of $\mathbf{w}(n)$ that needs PL operations. Thus, the main complexity contribution of the FEAP algorithm is the filter updating part. Considering this, it is desirable to further reduce its complexity.

2.3. Proposed FEF approach

In many applications, calculation of the weight vector is not the main concern but only the error signal is required. By exploiting the shifted structure of the matrix $\mathbf{X}(n)$, we show that it is sufficient to obtain the error vector exactly even if $\mathbf{w}(n)$ is not available at every sampling period.

The weight vector $\mathbf{w}(n)$ can be alternatively rewritten as [3,4]

$$\mathbf{w}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{X}(n)\boldsymbol{\varphi}(n) \tag{12}$$

where

$$\hat{\mathbf{w}}(n-1)=\mathbf{w}(0)+\sum_{k=P}^{n-1}\mathbf{x}(n-k)\sum_{j=0}^{P-1}\varepsilon_j(n-k+j), \tag{13}$$

$$\boldsymbol{\varphi}(n)=[\varphi_0(n), \varphi_1(n), \ldots, \varphi_{P-1}(n)]^T \tag{14}$$

with

$$\varphi_m(n)=\sum_{i=0}^{m}\varepsilon_i(n-m+i). \tag{15}$$

Both $\hat{\mathbf{w}}(n)$ and $\boldsymbol{\varphi}(n)$ can be recursively computed, respectively,

$$\boldsymbol{\varphi}(n)=\boldsymbol{\varepsilon}(n)+\begin{bmatrix}0\\ \overline{\boldsymbol{\varphi}}(n-1)\end{bmatrix} \tag{16}$$

$$\hat{\mathbf{w}}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{x}(n-P+1)\varphi_{P-1}(n) \tag{17}$$

where $\overline{\boldsymbol{\varphi}}(n)$ consists of the upper $P-1$ elements of $\boldsymbol{\varphi}(n)$.

Having expressed $\mathbf{w}(n)$ in a special way in (12), we can use the auxiliary coefficient vector $\hat{\mathbf{w}}(n)$ instead of $\mathbf{w}(n)$ to calculate $z_0(n)$. Substituting (12) into (11) yields

$$\begin{aligned} z_0(n)&=\mathbf{x}^T(n)\mathbf{w}(n-2)\\ &=\mathbf{x}^T(n)[\hat{\mathbf{w}}(n-3)+\mathbf{X}(n-2)\boldsymbol{\varphi}(n-2)]\\ &=\mathbf{x}^T(n)\hat{\mathbf{w}}(n-3)+\mathbf{r}^T(n)\boldsymbol{\varphi}(n-2) \end{aligned} \tag{18}$$

where $\mathbf{r}(n)=\mathbf{X}^T(n-2)\mathbf{x}(n)$. Note that (18) is the exact calculation of $z_0(n)$ without any assumption.

Defining $\rho_m(n)=\mathbf{x}^T(n)\mathbf{x}(n-m)$, the matrices $\mathbf{R}(n)$, $\mathbf{G}(n)$, and $\mathbf{r}(n)$ can be written as

$$\begin{aligned}\mathbf{R}(n)&=\mathbf{X}^T(n)\mathbf{X}(n)\\ &=\begin{bmatrix}\rho_0(n) & \rho_1(n) & \ldots & \rho_{P-1}(n)\\ \rho_1(n) & \rho_0(n-1) & \ldots & \rho_{P-2}(n-1)\\ \ldots & \ldots & \ldots & \ldots\\ \rho_{P-1}(n) & \rho_{P-2}(n-1) & \ldots & \rho_0(n-P+1)\end{bmatrix}\end{aligned} \tag{19}$$

$$\begin{aligned}\mathbf{G}(n)&=\mathbf{X}^T(n)\mathbf{X}(n-1)\\ &=\begin{bmatrix}\rho_1(n) & \rho_2(n) & \ldots & \rho_P(n)\\ \rho_0(n-1) & \rho_1(n-1) & \ldots & \rho_{P-1}(n-1)\\ \ldots & \ldots & \ldots & \ldots\\ \rho_{P-2}(n-1) & \rho_{P-3}(n-2) & \ldots & \rho_1(n-P+1)\end{bmatrix}\end{aligned} \tag{20}$$

$$\begin{aligned}\mathbf{r}(n)&=\mathbf{X}^T(n-2)\mathbf{x}(n)\\ &=[\rho_2(n), \rho_3(n), \ldots, \rho_{P+1}(n)]^T.\end{aligned} \tag{21}$$

From (19) to (21) it is noted that the update of $\mathbf{R}(n)$, $\mathbf{G}(n)$, and $\mathbf{r}(n)$ only need to calculate $P+2$ elements, i.e., $\rho_m(n), m=0,1,\ldots,P+1$. They can be updated recursively:

$$\rho_m(n)=\rho_m(n-1)+x(n)x(n-m)-x(n-L)x(n-m-L). \tag{22}$$

Update of the three matrices only needs $2(P+2)$ multiplication and $2(P+2)$ additions. The calculation of $\mathbf{z}(n)$ requires $L+3P+4$ operations in total. The new algorithm uses $\hat{\mathbf{w}}(n)$ rather than $\mathbf{w}(n)$ to calculate the filtered-out vector $\mathbf{y}(n)$. Update of $\mathbf{w}(n)$ needs PL operations while calculation of $\hat{\mathbf{w}}(n)$ only needs L operations.

According to (17) and (18), we have to save $\hat{\mathbf{w}}(n-3)$, $\hat{\mathbf{w}}(n-2)$, and $\hat{\mathbf{w}}(n-1)$ at time index n which needs more memory size than the original AP algorithm. However, the calculation of $z_0(n+1)$ only requires $\hat{\mathbf{w}}(n-2)$ at time index $n+1$, we can update $\hat{\mathbf{w}}(n-2)$ but not $\hat{\mathbf{w}}(n)$ at time index n as follows:

$$\hat{\mathbf{w}}(n-2)=\hat{\mathbf{w}}(n-3)+\mathbf{x}(n-P-1)\varphi_{P-1}(n-2). \tag{23}$$

Consequently, memory requirements in the new algorithm remain similar to the standard AP algorithm.

Another difficulty is to solve the linear system of equations $[\mathbf{R}(n)+\delta\mathbf{I}]\boldsymbol{\varepsilon}(n)=\mu\mathbf{e}(n)$. This problem has been extensively investigated in the literature [3–20]. Detailed discussion is beyond the scope of this paper. A good overview of this problem can be found in [12]. The proposed algorithm is summarized in Table 3. Both the FEAP and EFEAP algorithms have the same performance as the standard AP algorithm. However, the complexity of the EFEAP algorithm is only slightly higher than the NLMS algorithm but much lower than the FEAP algorithm.

Table 3
Proposed EFEAP algorithm.

Equation	$\times$	$+$
$\mathbf{G}(n)=\mathbf{X}^T(n)\mathbf{X}(n-1)$	$2P$	$2P$
$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	2	2
$\mathbf{r}(n)=\mathbf{X}^T(n-2)\mathbf{x}(n)$	2	2
$z_0(n)=\mathbf{x}^T(n)\hat{\mathbf{w}}(n-3)+\mathbf{r}^T(n)\boldsymbol{\varphi}(n-2)$	$L+P$	$L+P$
$\mathbf{z}(n)=[z_0(n),y_0(n-1),\ldots,y_{P-2}(n-1)]^T$	0	0
$\mathbf{y}(n)=\mathbf{z}(n)+\mathbf{G}(n)\boldsymbol{\varepsilon}(n-1)$	P^2	P^2
$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{y}(n)$	0	P
$\boldsymbol{\varepsilon}(n)=\mu[\mathbf{R}(n)+\delta\mathbf{I}]^{-1}\mathbf{e}(n)$	P_m	P_a
$\boldsymbol{\varphi}(n)=\boldsymbol{\varepsilon}(n)+\begin{bmatrix}0\\ \overline{\boldsymbol{\varphi}}(n-1)\end{bmatrix}$	0	P
$\hat{\mathbf{w}}(n-2)=\hat{\mathbf{w}}(n-3)+\mathbf{x}(n-P-1)\varphi_{P-1}(n-2)$	L	L
Total		
$2L+P^2+3P+P_m+4$ multiplications		
$2L+P^2+4P+P_a+4$ additions		

3. Case studies

In the previous section, we have presented a new FEF approach to reduce the complexity of AP algorithm. We found that the idea can also be generalized to some other known AP-type algorithms. We will show the details and also discuss the possible applications of the proposed approach.

3.1. FEF approach to the DCD-AP algorithm

The DCD-AP algorithm [18] uses the DCD iterations to solve the linear system of equations. The number of multiplications required in the DCD-AP algorithm is significantly reduced, however, the number of additions is still large [18]. Combing the DCD iterations and the proposed FEF approach results in the DCD-EFEAP algorithm in Table 4, where we define $\tilde{\mathbf{R}}(n)=\mathbf{R}(n)+\delta\mathbf{I}$ and $\mathbf{b}=[b_0,b_1,\ldots,b_{P-1}]^T$. Parameters of the DCD iterations are as follows: N_u denotes the maximum number of "successful" iterations, M_b is the number of bits used for representation of elements of $\boldsymbol{\varepsilon}(n)$, $[-H,H]$ represents the amplitude range of $\boldsymbol{\varepsilon}(n)$, and $\tilde{\mathbf{R}}^{(q)}(n)$ is the qth column of the matrix $\tilde{\mathbf{R}}(n)$.

The DCD-EFEAP and DCD-AP algorithms with different complexities have exactly the same performance. The DCD-EFEAP algorithm takes L more multiplications but saves $(N_u-1)L$ additions compared to the DCD-AP algorithm. If the algorithm is implemented on software platforms such as DSP and ARM, both the multiplication and addition operations should be taken into account. The proposed DCD-EFEAP algorithm is more appropriate for implementation in the software.

Table 5 presents the complexity of several fast AP algorithms. Note that the proposed EFEAP algorithm needs P^2 more multiplications than the FAP algorithm if they use the same method to solve (6). In practical applications such as AEC, P is between 2 and 10 and $P \ll L$ [12], so the EFEAP algorithm is only marginally more complex than the FAP algorithm.

Table 4
Proposed DCD-EFEAP algorithm.

Equation	$\times$	$+$
$\mathbf{G}(n)=\mathbf{X}^T(n)\mathbf{X}(n-1)$	$2P$	$2P$
$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	2	2
$\mathbf{r}(n)=\mathbf{X}^T(n-2)\mathbf{x}(n)$	2	2
$z_0(n)=\mathbf{x}^T(n)\hat{\mathbf{w}}(n-3)+\mathbf{r}^T(n)\boldsymbol{\varphi}(n-2)$	$L+P$	$L+P$
$\mathbf{z}(n)=[z_0(n),y_0(n-1),\ldots,y_{P-2}(n-1)]^T$	0	0
$\mathbf{y}(n)=\mathbf{z}(n)+\mathbf{G}(n)\boldsymbol{\varepsilon}(n-1)$	P^2	P^2
$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{y}(n)$	0	P
$\boldsymbol{\varepsilon}(n)=\mathbf{0},\mathbf{b}=\mu\mathbf{e}(n),\alpha=H/2,m=1$	P	0
for $l=1,\ldots,N_u$	0	0
$q=\underset{i=0,1,\ldots,P-1}{\arg\max}\{\lvert b_i\rvert\}$	0	$P-1$
while $\lvert b_q\rvert\le(\alpha/2)\tilde{\mathbf{R}}_{q,q}(n)\&m\le M_b$	0	1
$m=m+1,\alpha=\alpha/2$		
endwhile		
if $m>M_b$ break endif	0	0
$\varepsilon_q(n)=\varepsilon_q(n)+\mathrm{sign}(b_q)\alpha$	0	1
$\mathbf{b}=\mathbf{b}-\mathrm{sign}(b_q)\alpha\tilde{\mathbf{R}}^{(q)}(n)$	0	P
endfor	0	0
$\boldsymbol{\varphi}(n)=\boldsymbol{\varepsilon}(n)+\begin{bmatrix}0\\ \overline{\boldsymbol{\varphi}}(n-1)\end{bmatrix}$	0	P
$\hat{\mathbf{w}}(n-2)=\hat{\mathbf{w}}(n-3)+\mathbf{x}(n-P-1)\varphi_{P-1}(n-2)$	L	L
Total		
$2L+P^2+3P+4$ multiplications		
$2L+P^2+(4+2N_u)P+M_b+4$ additions		

3.2. FEF approach to the APS algorithm

In many applications, the noise exhibits non-Gaussian behavior and the distributions have heavier tails than those of the Gaussian distribution [24,25]. Recently, the APS algorithm [22] is proposed to enhance the robustness against such non-Gaussian interferences.

Direct implementation of the APS algorithm requires $PL+2L$ multiplications and $2PL+L$ additions at each time instant [21]. A fast APS (FAPS) algorithm [21] was presented based on the FEF scheme [18]. The FAPS algorithm needs $2L+3P$ multiplications and $(P+1)L+2P^2+3P$ additions per sample. Although the multiplications used in the FAPS algorithm are significantly reduced, the number of additions is still high [21]. The complexity of the APS algorithm can be further reduced by using the proposed FEF approach.

Using the definitions in (2)–(5), the weight update equation of the APS algorithm can be written as [19]

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\frac{\mu\mathbf{x}_s(n)}{\sqrt{\mathbf{x}_s^T(n)\mathbf{x}_s(n)+\delta}} \tag{24}$$

where $\mathbf{x}_s(n)=\mathbf{X}(n)\mathrm{sign}[\mathbf{e}(n)]$. Rearranging (24), it follows

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\mathbf{X}(n)\boldsymbol{\varepsilon}(n) \tag{25}$$

where

$$\boldsymbol{\varepsilon}(n)=\frac{\mu\ \mathrm{sign}[\mathbf{e}(n)]}{\sqrt{\mathrm{sign}[\mathbf{e}^T(n)]\mathbf{R}(n)\mathrm{sign}[\mathbf{e}(n)]+\delta}}. \tag{26}$$

We found that the update equation of the APS algorithm is similar to that of the AP algorithm. The only difference is the calculation of $\boldsymbol{\varepsilon}(n)$ in (6) and (26). Thus, the proposed FEF approach can be applied to the APS algorithm. The

Table 5
Complexity of several fast versions of the AP algorithm.

Algorithm	Comment	Number of multiplications per sample	Number of additions per sample
AP [1]	Exact filtering	$2PL+3P+P_m$	$2PL+2P+P_a$
FAP [3,4,6–13]	Approximate filtering	$2L+4P+P_m$	$2L+4P+P_a$
FEAP [18]	Exact filtering	$(P+1)L+P^2+3P+P_m$	$(P+1)L+P^2+3P+P_a$
EFEAP	Exact filtering	$2L+P^2+3P+P_m$	$2L+P^2+4P+P_a$
DCD-FAP [11]	Approximate filtering	$2L+4P$	$2L+(2N_u+M_b+4)P$
DCD-AP [18]	Exact filtering	$L+P^2+3P$	$(N_u+1)L+P^2+(2N_u+3)P$
DCD-EFEAP	Exact filtering	$2L+P^2+3P$	$2L+P^2+(2N_u+4)P$

proposed fast APS algorithm is listed in Table 6. Note that both the multiplications and the additions required in the proposed algorithm are reduced.

3.3. FEF approach to the MFxAP algorithm

The filtered-x least mean squares (FxLMS) algorithm is commonly used in the ANC system [26] but suffers from slow convergence. The AP algorithms have been introduced to ANC applications with the aim of improving the convergence performance of the FxLMS algorithm and avoiding the instabilities of the RLS algorithm [27]. The conventional filtered-x affine projection (CFxAP) algorithm uses past samples of the error signal to build the error vector [15]. However, the delay introduced by the secondary paths reduces the upper bound for the step size that can be used, resulting in slow convergence [28]. To compensate the delay, the MFxAP algorithm is presented [5,14] but the complexity is very high. The fast MFxAP algorithm [15] is subsequently proposed to reduce the complexity based on the FAF approach. However, the convergence performance of the fast MFxAP algorithm is degraded by the approximations involved in the error vector computation. The FEF approach in [18] is also applied to the MFxAP algorithm [19] but the complexity is only mildly reduced.

3.3.1. Multichannel MFxAP algorithm

A multichannel active noise controller scheme (I reference signals, J secondary sources, and K error sensors) is depicted in Fig. 1. Notation in Table 7 will be used to describe the MFxAP algorithm. We only discuss the multichannel MFxAP algorithm [15] because the monochannel MFxAP algorithm [5] is a special case of the former at $I=J=K=1$.

The MFxAP algorithm can be described as

$$v_{i,j,k}(n)=\mathbf{h}_{j,k}^T\mathbf{x}_{Mi}(n), \quad (27)$$

$$y_j(n)=\sum_{i=1}^{I}\mathbf{w}_{i,j}^T(n-1)\mathbf{x}_i(n), \quad (28)$$

$$\hat{d}_k(n)=e_k(n)-\sum_{j=1}^{J}\mathbf{h}_{j,k}^T\mathbf{y}_j(n), \quad (29)$$

$$\begin{aligned}\hat{\mathbf{y}}_{i,j,k}(n)&=[\hat{y}_{i,j,k,0}(n),\hat{y}_{i,j,k,1}(n),\ldots,\hat{y}_{i,j,k,P-1}(n)]^T\\&=\mathbf{V}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-1),\end{aligned} \quad (30)$$

Table 6
Proposed fast APS algorithm.

Term	×	+
$\mathbf{G}(n)=\mathbf{X}^T(n)\mathbf{X}(n-1)$	$2P$	$2P$
$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	2	2
$\mathbf{r}(n)=\mathbf{X}^T(n-2)\mathbf{x}(n)$	2	2
$z_0(n)=\mathbf{x}^T(n)\hat{\mathbf{w}}(n-3)+\mathbf{r}^T(n)\boldsymbol{\varphi}(n-2)$	$L+P$	$L+P$
$\mathbf{z}(n)=[z_0(n),y_0(n-1),\ldots,y_{P-2}(n-1)]^T$	0	0
$\mathbf{y}(n)=\mathbf{z}(n)+\mathbf{G}(n)\boldsymbol{\varepsilon}(n-1)$	P^2	P^2
$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{y}(n)$	0	P
$\boldsymbol{\varepsilon}(n)=\dfrac{\mu\,\mathrm{sign}[\mathbf{e}(n)]}{\sqrt{\mathrm{sign}[\mathbf{e}^T(n)]\mathbf{R}(n)\,\mathrm{sign}[\mathbf{e}(n)]+\delta}}$	0	P^2
$\boldsymbol{\varphi}(n)=\boldsymbol{\varepsilon}(n)+\begin{bmatrix}0\\ \overline{\boldsymbol{\varphi}}(n-1)\end{bmatrix}$	0	P
$\hat{\mathbf{w}}(n-2)=\hat{\mathbf{w}}(n-3)+\mathbf{x}(n-P-1)\varphi_{P-1}(n-2)$	L	L
Total $2L+P^2+3P+4$ multiplications $2L+2P^2+4P+4$ additions		

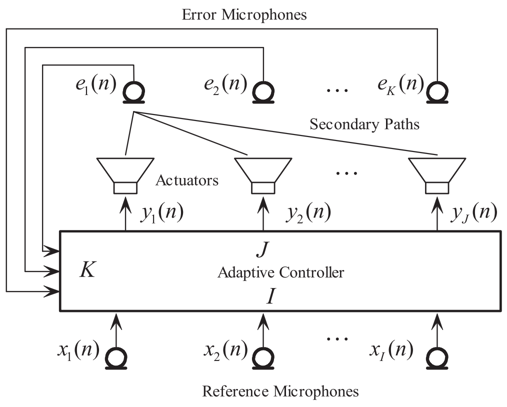

Fig. 1. Multichannel feedback active noise control system with adaptive filters.

$$\mathbf{R}_{i,j,k}(n)=\mathbf{V}_{i,j,k}^T(n)\mathbf{V}_{i,j,k}(n), \quad (31)$$

$$\hat{\mathbf{e}}_k(n)=\hat{\mathbf{d}}_k(n)+\sum_{i=1}^{I}\sum_{j=1}^{J}\hat{\mathbf{y}}_{i,j,k}(n), \quad (32)$$

$$\begin{aligned}\boldsymbol{\varepsilon}_{i,j,k}(n)&=[\varepsilon_{i,j,k,0}(n),\varepsilon_{i,j,k,1}(n),\ldots,\varepsilon_{i,j,k,P-1}(n)]^T\\&=\mu[\mathbf{R}_{i,j,k}(n)+\delta\mathbf{I}]^{-1}\hat{\mathbf{e}}_k(n),\end{aligned} \quad (33)$$

$$\mathbf{w}_{i,j}(n)=\mathbf{w}_{i,j}(n-1)-\sum_{k=1}^{K}\mathbf{V}_{i,j,k}(n)\boldsymbol{\varepsilon}_{i,j,k}(n) \quad (34)$$

Table 7
Notations for the MFxAP algorithm.

Symbol	Definition
I	Number of reference sensors
J	Number of actuators
K	Number of error sensors
L	Length of the FIR adaptive filters
P	Projection order
M	Length of (fixed) FIR filters
$x_i(n)$	Value at time n of the ith reference signal
$y_j(n)$	Value at time n of the jth actuator signal
$e_k(n)$	Value at time n of the kth error sensor
$\hat{d}_k(n)$	Estimate of $d_k(n)$
$\hat{e}_k(n)$	Value at time n of the kth error sensor used for adaptation of the weight vector
$w_{i,j,l}(n)$	Value at time n of the lth coefficient in the adaptive FIR filter linking $x_i(n)$ and $y_j(n)$
$h_{j,k,m}$	Value of the mth coefficient in the (fixed) FIR filter modeling the plant between $y_j(n)$ and $e_k(n)$
$v_{i,j,k}(n)$	Value at time n of the filtered reference signal, i.e., the signal obtained by filtering the signal $x_i(n)$ with the plant model filter $\mathbf{h}_{j,k}$
$\mathbf{w}_{i,j}(n)$	$[w_{i,j,0}(n), w_{i,j,1}(n), \ldots, w_{i,j,L-1}(n)]^T$
$\mathbf{h}_{j,k}$	$[h_{j,k,0}, h_{j,k,1}, \ldots, h_{j,k,M-1}]^T$
$\mathbf{v}_{i,j,k}(n)$	$[v_{i,j,k}(n), v_{i,j,k}(n-1), \ldots, v_{i,j,k}(n-L+1)]^T$
$\mathbf{V}_{i,j,k}(n)$	$[\mathbf{v}_{i,j,k}(n), \mathbf{v}_{i,j,k}(n-1), \ldots, \mathbf{v}_{i,j,k}(n-P+1)]$
$\mathbf{x}_i(n)$	$[x_i(n), x_i(n-1), \ldots, x_i(n-L+1)]^T$
$\mathbf{x}_{Mi}(n)$	$[x_i(n), x_i(n-1), \ldots, x_i(n-M+1)]^T$
$\mathbf{y}_i(n)$	$[y_i(n), y_i(n-1), \ldots, y_i(n-M+1)]^T$
$\hat{\mathbf{d}}_k(n)$	$[\hat{d}_k(n), \hat{d}_k(n-1), \ldots, \hat{d}_k(n-P+1)]^T$
$\hat{\mathbf{e}}_k(n)$	$[\hat{e}_k(n), \hat{e}_k(n-1), \ldots, \hat{e}_k(n-P+1)]^T$

where μ is the step size and δ is a regularization parameter. Among the different available definitions of $\boldsymbol{\varepsilon}_{i,j,k}(n)$ [29], we use the approach in (33).

From (31) it is noted that the lower $(P-1)\times(P-1)$ block of $\mathbf{R}_{i,j,k}(n)$ can be obtained by copying the upper $(P-1)\times(P-1)$ block of $\mathbf{R}_{i,j,k}(n-1)$. The only part of the matrix $\mathbf{R}_{i,j,k}(n)$ that should be directly updated is the elements in the first row. The update of $\mathbf{R}_{i,j,k}(n)$ only needs $2P$ multiplications per sample by using a recursive scheme.

3.3.2. *Proposed fast MFxAP algorithm*

Expanding the matrix/vector multiplications in (34), the weight vector $\mathbf{w}_{i,j}(n)$ can be rewritten as

$$\mathbf{w}_{i,j}(n) = \mathbf{w}_{i,j}(n-1) - \sum_{k=1}^{K}\sum_{m=0}^{P-1} \mathbf{v}_{i,j,k}(n-m)\varepsilon_{i,j,k,m}(n). \tag{35}$$

Continuing to recursively expand Eq. (35) yields

$$\mathbf{w}_{i,j}(n) = \mathbf{w}_{i,j}(0) - \sum_{k=1}^{K}\sum_{l=0}^{n-1}\sum_{m=0}^{P-1} \mathbf{v}_{i,j,k}(n-l-m)\varepsilon_{i,j,k,m}(n-l). \tag{36}$$

Assuming that $\mathbf{v}_{i,j,k}(n) = \mathbf{0}$ for $n<0$, (36) can be rewritten as

$$\mathbf{w}_{i,j}(n) = \mathbf{w}_{i,j}(0) - \sum_{k=1}^{K}\sum_{l=0}^{P-1} \mathbf{v}_{i,j,k}(n-l) \sum_{m=0}^{l} \varepsilon_{i,j,k,m}(n-l+m) - \sum_{k=1}^{K}\sum_{l=P}^{n-1} \mathbf{v}_{i,j,k}(n-l) \sum_{m=0}^{P-1} \varepsilon_{i,j,k,m}(n-l+m). \tag{37}$$

The weight vector $\mathbf{w}_{i,j}(n)$ can be alternatively rewritten as

$$\mathbf{w}_{i,j}(n) = \hat{\mathbf{w}}_{i,j}(n-1) - \sum_{k=1}^{K} \mathbf{V}_{i,j,k}(n)\boldsymbol{\varphi}_{i,j,k}(n) \tag{38}$$

where

$$\hat{\mathbf{w}}_{i,j}(n-1) = \mathbf{w}_{i,j}(0) - \sum_{k=1}^{K}\sum_{l=P}^{n-1} \mathbf{v}_{i,j,k}(n-l) \sum_{m=0}^{P-1} \varepsilon_{i,j,k,m}(n-l+m) \tag{39}$$

$$\boldsymbol{\varphi}_{i,j,k}(n) = [\varphi_{i,j,k,0}(n), \varphi_{i,j,k,1}(n), \ldots, \varphi_{i,j,k,P-1}(n)]^T \tag{40}$$

with

$$\varphi_{i,j,k,q}(n) = \sum_{m=0}^{q} \varepsilon_{i,j,k,m}(n-q+m). \tag{41}$$

It is interesting that both $\hat{\mathbf{w}}_{i,j}(n)$ and $\boldsymbol{\varphi}_{i,j,k}(n)$ can be recursively computed:

$$\boldsymbol{\varphi}_{i,j,k}(n) = \boldsymbol{\varepsilon}_{i,j,k}(n) + \begin{bmatrix} 0 \\ \overline{\boldsymbol{\varphi}}_{i,j,k}(n-1) \end{bmatrix}, \tag{42}$$

$$\hat{\mathbf{w}}_{i,j}(n) = \hat{\mathbf{w}}_{i,j}(n-1) - \sum_{k=1}^{K} \mathbf{v}_{i,j,k}(n-P+1)\varphi_{i,j,k,P-1}(n) \tag{43}$$

where $\overline{\boldsymbol{\varphi}}_{i,j,k}(n)$ consists of the upper $P-1$ elements of $\boldsymbol{\varphi}_{i,j,k}(n)$.

Substituting (34) into (30) yields

$$\begin{aligned}\hat{\mathbf{y}}_{i,j,k}(n) &= \mathbf{V}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-1) \\ &= \mathbf{V}_{i,j,k}^T(n)\left[\mathbf{w}_{i,j}(n-2) - \sum_{q=1}^{K} \mathbf{V}_{i,j,q}(n-1)\boldsymbol{\varepsilon}_{i,j,q}(n-1)\right] \\ &= \mathbf{z}_{i,j,k}(n) - \sum_{q=1}^{K} \mathbf{G}_{i,j,k,q}(n)\boldsymbol{\varepsilon}_{i,j,q}(n-1)\end{aligned} \tag{44}$$

where $\mathbf{G}_{i,j,k,q}(n) = \mathbf{V}_{i,j,k}^T(n)\mathbf{V}_{i,j,q}(n-1)$, and

$$\begin{aligned}\mathbf{z}_{i,j,k}(n) &= \mathbf{V}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-2) \\ &= [\mathbf{v}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-2), \mathbf{v}_{i,j,k}^T(n-1)\mathbf{w}_{i,j}(n-2), \\ &\quad \ldots, \mathbf{v}_{i,j,k}^T(n-P+1)\mathbf{w}_{i,j}(n-2)]^T \\ &= [z_{i,j,k}(n), \hat{y}_{i,j,k,0}(n-1), \ldots, \hat{y}_{i,j,k,P-2}(n-1)]^T\end{aligned} \tag{45}$$

with $z_{i,j,k}(n) = \mathbf{v}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-2)$. Using (38), $z_{i,j,k}(n)$ can be calculated using the auxiliary coefficient vector $\hat{\mathbf{w}}_{i,j}(n)$ rather than using $\mathbf{w}_{i,j}(n)$:

$$\begin{aligned}z_{i,j,k}(n) &= \mathbf{v}_{i,j,k}^T(n)\mathbf{w}_{i,j}(n-2) \\ &= \mathbf{v}_{i,j,k}^T(n)\left[\hat{\mathbf{w}}_{i,j}(n-3) - \sum_{q=1}^{K} \mathbf{V}_{i,j,q}(n-2)\boldsymbol{\varphi}_{i,j,q}(n-2)\right] \\ &= \mathbf{v}_{i,j,k}^T(n)\hat{\mathbf{w}}_{i,j}(n-3) - \sum_{q=1}^{K} \boldsymbol{\psi}_{i,j,k,q}^T(n)\boldsymbol{\varphi}_{i,j,q}(n-2)\end{aligned} \tag{46}$$

where $\boldsymbol{\psi}_{i,j,k,q}(n) = \mathbf{V}_{i,j,q}^T(n-2)\mathbf{v}_{i,j,k}(n)$.

The auxiliary coefficient strategy can also be applied to compute the cancelling signal;

$$\begin{aligned}y_j(n) &= \sum_{i=1}^{I} \mathbf{w}_{i,j}^T(n-1)\mathbf{x}_i(n) \\ &= \sum_{i=1}^{I} \mathbf{x}_i^T(n)\left[\hat{\mathbf{w}}_{i,j}(n-2) - \sum_{k=1}^{K} \mathbf{V}_{i,j,k}(n-1)\boldsymbol{\varphi}_{i,j,k}(n-1)\right] \\ &= \sum_{i=1}^{I}\left[\mathbf{x}_i^T(n)\hat{\mathbf{w}}_{i,j}(n-2) - \sum_{k=1}^{K} \mathbf{r}_{i,j,k}^T(n)\boldsymbol{\varphi}_{i,j,k}(n-1)\right].\end{aligned} \tag{47}$$

where $\mathbf{r}_{i,j,k}(n) = \mathbf{V}_{i,j,k}^T(n-1)\mathbf{x}_i(n)$. The vector $\mathbf{r}_{i,j,k}(n)$ can be recursively computed and only needs $2P$ multiplications:

$$\mathbf{r}_{i,j,k}(n) = \mathbf{r}_{i,j,k}(n-1) + \boldsymbol{\eta}_{i,j,k}(n-1)x_i(n) - \boldsymbol{\eta}_{i,j,k}(n-1-L)x_i(n-L) \tag{48}$$

where $\boldsymbol{\eta}_{i,j,k}(n) = [v_{i,j,k}(n), v_{i,j,k}(n-1), \ldots, v_{i,j,k}(n-P+1)]^T$.

The remaining problem is the calculation of $\mathbf{G}_{i,j,k,q}(n)$ and $\boldsymbol{\psi}_{i,j,k,q}(n)$. We firstly define the cross-correlation element

$$\rho_{i,j,k,q}^{(m)}(n) = \mathbf{v}_{i,j,k}^T(n)\mathbf{v}_{i,j,q}(n-m). \tag{49}$$

Using (49), we can write $\mathbf{G}_{i,j,k,q}(n)$ and $\boldsymbol{\psi}_{i,j,k,q}(n)$ as follows:

$$\mathbf{G}_{i,j,k,q}(n) = \mathbf{V}_{i,j,k}^T(n)\mathbf{V}_{i,j,q}(n-1) = \begin{bmatrix} \rho_{i,j,k,q}^{(1)}(n) & \rho_{i,j,k,q}^{(2)}(n) & \ldots & \rho_{i,j,k,q}^{(P)}(n) \\ \rho_{i,j,q,k}^{(0)}(n-1) & \rho_{i,j,k,q}^{(1)}(n-1) & \ldots & \rho_{i,j,k,q}^{(P-1)}(n-1) \\ \vdots & \vdots & \ldots & \vdots \\ \rho_{i,j,q,k}^{(P-2)}(n-1) & \rho_{i,j,q,k}^{(P-3)}(n-2) & \ldots & \rho_{i,j,k,q}^{(1)}(n-P+1) \end{bmatrix}, \tag{50}$$

$$\begin{aligned} \boldsymbol{\psi}_{i,j,k,q}(n) &= \mathbf{V}_{i,j,q}^T(n-2)\mathbf{v}_{i,j,k}(n) \\ &= [\mathbf{v}_{i,j,k}^T(n)\mathbf{v}_{i,j,q}(n-2), \mathbf{v}_{i,j,k}^T(n)\mathbf{v}_{i,j,q}(n-3), \\ &\quad \ldots, \mathbf{v}_{i,j,k}^T(n)\mathbf{v}_{i,j,q}(n-P-1)]^T \\ &= [\rho_{i,j,k,q}^{(2)}(n), \rho_{i,j,k,q}^{(3)}(n), \ldots, \rho_{i,j,k,q}^{(P+1)}(n)]. \end{aligned} \tag{51}$$

It can be seen from (50) and (51) that the update of $\mathbf{G}_{i,j,k,q}(n)$ and $\boldsymbol{\psi}_{i,j,k,q}(n)$ only needs to compute $\rho_{i,j,k,q}^{(m)}(n)$, $m = 0, 1, \ldots, P+1$. They can be recursively computed as

$$\rho_{i,j,k,q}^{(m)}(n) = \rho_{i,j,k,q}^{(m)}(n-1) + v_{i,j,k}(n)v_{i,j,q}(n-m) - v_{i,j,k}(n-L)v_{i,j,q}(n-m-L). \tag{52}$$

Thus, joint update of $\mathbf{G}_{i,j,k,q}(n)$ and $\boldsymbol{\psi}_{i,j,k,q}(n)$ only needs $2P+4$ multiplications. The proposed fast MFxAP algorithm is presented in Table 8.

In the special case $I = J = K = 1$, (27)–(34) becomes the monochannel structure used in [5]. It is noted from Table 8 that the proposed fast MFxAP algorithm needs $3L + 2M + P^2 + 8P + P_m$ multiplications per sample for a monochannel case. The monochannel fast MFxAP algorithm [5] needs $3L + 2M + 5P + P_m$ multiplications per sample. Thus, the proposed algorithm has a similar complexity with the fast algorithm in [5], but the former provides an exact filtering. Table 9 evaluates the complexity of several fast algorithmic variants of the FxAP algorithms.

3.4. Other applications

In fact, the core idea of the proposed FEF approach lies in the time-shift property of the vector in (4) and (7). In this manner, the proposed FEF scheme can also be extended to many other adaptive filtering algorithms that have a similar filtering and weight update structure without fundamental problems, e.g., the proportionate-type AP [30], improved multiband-structured subband adaptive filter (IMSAF) [31], and affine projection sign subband adaptive filter (AP-SSAF) [32] algorithms.

Table 8
Proposed fast MFxAP algorithm.

Equation	$\times$
$v_{i,j,k}(n) = \mathbf{h}_{j,k}^T\mathbf{x}_{Mi}(n)$	$IJKM$
$\mathbf{r}_{i,j,k}(n) = \mathbf{V}_{i,j,k}^T(n-1)\mathbf{x}_i(n)$	$2IJKP$
$y_j(n) = \sum_{i=1}^{I}\left[\mathbf{x}_i^T(n)\hat{\mathbf{w}}_{i,j}(n-2) - \sum_{k=1}^{K}\mathbf{r}_{i,j,k}^T(n)\boldsymbol{\varphi}_{i,j,k}(n-1)\right]$	$IJ(L+PK)$
$\hat{d}_k(n) = e_k(n) - \sum_{j=1}^{J}\mathbf{h}_{j,k}^T\mathbf{y}_j(n)$	JKM
$\boldsymbol{\psi}_{i,j,k,q}(n) = \mathbf{V}_{i,j,k}^T(n-2)\mathbf{v}_{i,j,q}(n)$	$2IJK^2$
$\mathbf{G}_{i,j,k,q}(n) = \mathbf{V}_{i,j,k}^T(n)\mathbf{V}_{i,j,q}(n-1)$	$IJK^2(2P+2)$
$z_{i,j,k}(n) = \mathbf{v}_{i,j,k}^T(n)\hat{\mathbf{w}}_{i,j}(n-3) - \sum_{q=1}^{K}\boldsymbol{\psi}_{i,j,k,q}^T(n)\boldsymbol{\varphi}_{i,j,q}(n-2)$	$IJK(L+KP)$
$\mathbf{z}_{i,j,k}(n) = [z_{i,j,k}(n), \hat{y}_{i,j,k,0}(n-1), \ldots, \hat{y}_{i,j,k,P-2}(n-1)]^T$	0
$\hat{\mathbf{y}}_{i,j,k}(n) = \mathbf{z}_{i,j,k}(n) - \sum_{q=1}^{K}\mathbf{G}_{i,j,k,q}(n)\boldsymbol{\varepsilon}_{i,j,q}(n-1)$	IJK^2P^2
$\hat{\mathbf{e}}_k(n) = \hat{\mathbf{d}}_k(n) + \sum_{i=1}^{I}\sum_{j=1}^{J}\hat{\mathbf{y}}_{i,j,k}(n)$	0
$\mathbf{R}_{i,j,k}(n) = \mathbf{V}_{i,j,k}^T(n)\mathbf{V}_{i,j,k}(n)$	$2IJKP$
$\boldsymbol{\varepsilon}_{i,j,k}(n) = [\mathbf{R}_{i,j,k}(n) + \delta\mathbf{I}]^{-1}\hat{\mathbf{e}}_k(n)$	P_m
$\boldsymbol{\varphi}_{i,j,k}(n) = \boldsymbol{\varepsilon}_{i,j,k}(n) + \begin{bmatrix} 0 \\ \overline{\boldsymbol{\varphi}}_{i,j,k}(n-1) \end{bmatrix}$	0
$\hat{\mathbf{w}}_{i,j}(n) = \hat{\mathbf{w}}_{i,j}(n-1) - \sum_{k=1}^{K}\mathbf{v}_{i,j,k}(n-P+1)\varphi_{i,j,k,P-1}(n)$	$IJKL$

Table 9
Complexity of several FxAP algorithms.

Algorithm	Number of multiplications per sample
MFxAP	$LIJ(2KP+1) + JKM(I+1) + 2IJKP + P_m$
Fast MFxAP	$IJL(2K+1) + JKM(I+1) + 5IJKP + P_m$
Proposed MFxAP	$IJL(2K+1) + JKM(I+1) + IJKP(5+3K+KP) + P_m$
CFxAP	$IJL(KP+1) + IJKM + 2IJKP + P_m$
Fast CFxAP	$IJL(K+1) + IJKM + 5IJKP + P_m$

4. Complexity comparisons

Fig. 2 presents the complexity of several variants of AP algorithm based on the total number of operations (multiplications plus additions), where we use the $\mathbf{LDL}^T$ approach [12] to solve the linear system of equation in the first four algorithms in Table 5. The parameters are $L = 1024$, $M_b = 16$, and $N_u = P$. In the exact filtering methods, the complexity of the EFEAP algorithm is much lower than the FEAP and the standard AP algorithms especially for a large projection order. Note that for $P = 20$, the proposed EFEAP algorithm attains a complexity that is only 11% and 19% of the AP and FEAP algorithms, respectively. The complexity of the EFEAP algorithm is comparable to that of the FAP algorithm but the latter only provides an approximate filtering. The DCD-EFEAP algorithm achieves the least complexity among the three DCD-based algorithms in Table 5.

Fig. 3 illustrates the total number of operations (multiplications plus additions) required by three versions of the APS algorithm. The length of the adaptive filter is $L = 1024$. All the three algorithms are mathematically equal, but the proposed fast version achieves the lowest complexity. The computational load of the APS and FAPS algorithms increases linearly with the projection order while that of the proposed APS algorithm does not.

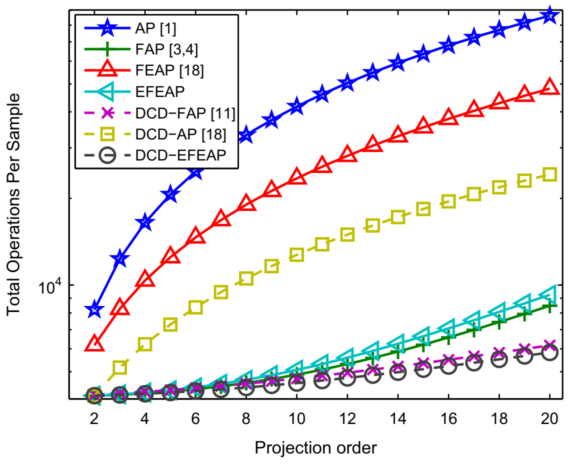

Fig. 2. Complexity comparison of several fast AP algorithms, $L=1024$.

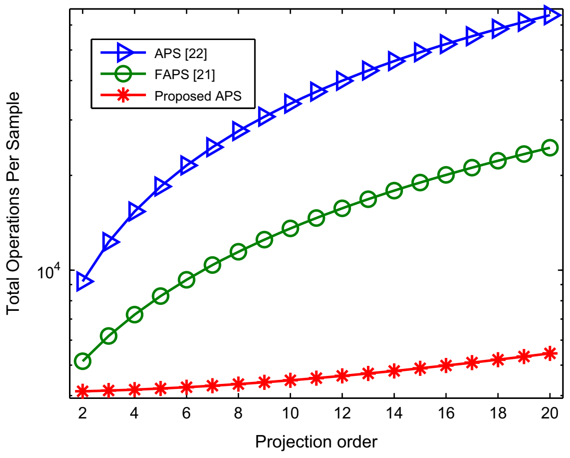

Fig. 3. Complexity comparison of several fast APS algorithms, $L=1024$.

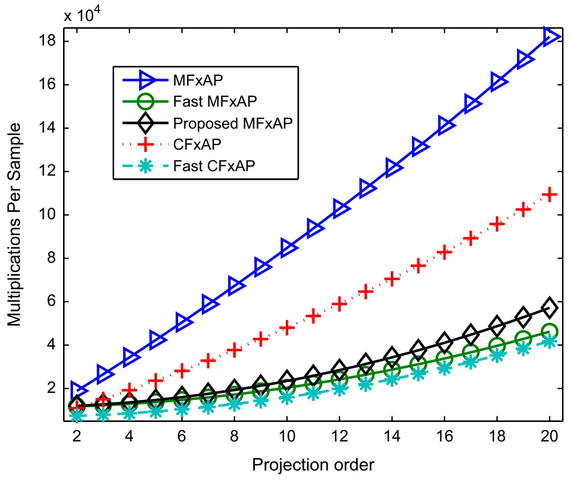

Fig. 4. Numerical complexity of the MFxAP, CFxAP, fast MFxAP, fast CFxAP and proposed fast MFxAP algorithms. The parameters are $I=2$, $J=3$, $K=2$, $L=300$, $M=128$.

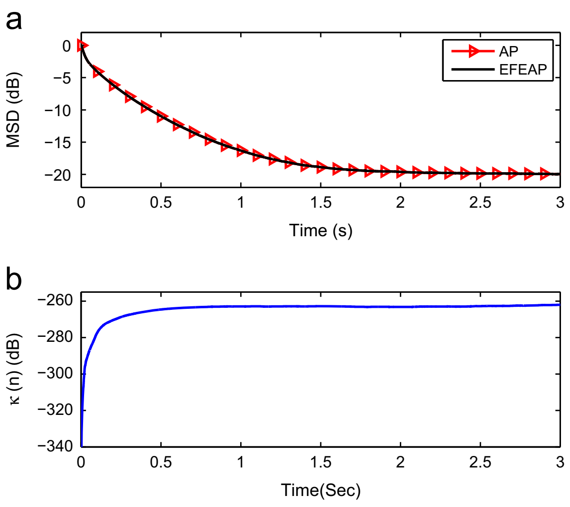

Fig. 5. Comparison between the AP and proposed EFEAP algorithms with $L=512$, $\mu=0.3$, $P=6$. (a) MSD of the two algorithms and (b) the Euclidean norm of the difference of the two computed solutions.

Fig. 4 presents a computational comparison of different FxAP algorithms, where the parameters are $I=2$, $J=3$, $K=2$, $L=300$, and $M=128$. To simplify, we use the recursive approach to solve the linear system of equation [15], i.e., $P_m=IJK(7P^2+12P)$. From Fig. 4 it is clear that the complexity of the proposed fast MFxAP algorithm is significantly lower than that of the standard MFxAP and CFxAP algorithms but only slightly higher than that of the Fast MFxAP and Fast CFxAP algorithms. However, the proposed fast MFxAP algorithm is an exact version of the standard MFxAP algorithm.

5. Simulation results

Computer simulations are carried out in the context of AEC. The impulse response $\mathbf{w}_o$ is generated according to $w_i=e^{-\tau i}r(i)$, $i=0,1,\ldots,L-1$, where $r(i)$ is a zero-mean white noise sequence and τ is the envelope decay rate. The sampling rate is 8 kHz. An independent white noise signal is added to the echo signal, with 30-dB signal-to-noise ratio (SNR). The convergence performance is evaluated in terms of the normalized mean square deviation (MSD), defined as $10\log_{10}[\|\mathbf{w}_o-\mathbf{w}(n)\|^2/\|\mathbf{w}_o\|^2]$. The length of the adaptive filter is $L=512$. The input signal is 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). Exact solution of the linear system of equations has been used in all the algorithms involved in the simulations except the original FAP algorithm. The results are obtained by averaging over 100 Monte Carlo trials.

Fig. 5 shows the convergence performance of the EFEAP and standard AP algorithms. Note from Fig. 5(a) that the MSD of the EFEAP algorithm matches well with that of the standard AP algorithm. Fig. 5(b) verifies that the two implementations are theoretically and practically equivalent by the extremely small values of the quantity $\kappa(n)=10\log_{10}[\|\mathbf{w}_{\mathrm{AP}}(n)-\mathbf{w}_{\mathrm{EFEAP}}(n)\|^2]$, i.e., $\kappa(n)=0$ within the numerical limits of the computations.

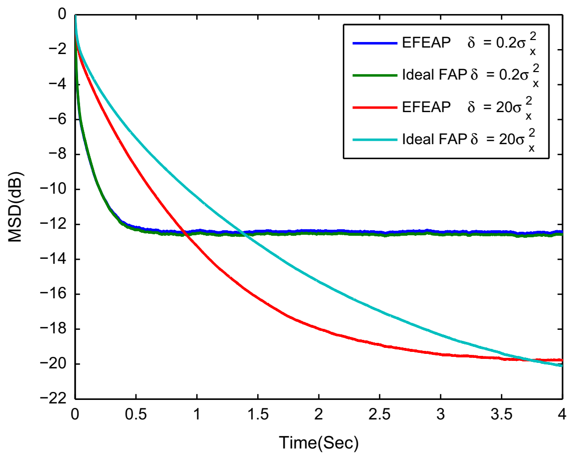

Fig. 6. Learning curves of the ideal FAP and EFEAP algorithms with $L = 512$, $\mu = 0.5$, $P = 4$.

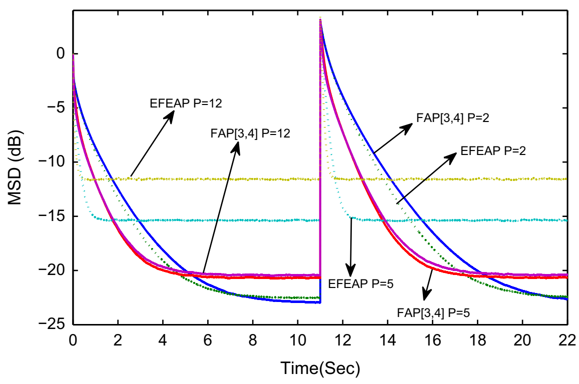

Fig. 7. Convergence characteristics of the original FAP and EFEAP algorithms with $L = 512$, $\mu = 1.0$.

The FAP algorithm based on an ideal matrix inversion is called ideal FAP [11]. Fig. 6 compares the convergence performance of the EFEAP and ideal FAP algorithms using different regularization parameters. We select $\delta = 0.2\sigma_x^2$ and $\delta = 20\sigma_x^2$, where σ_x^2 is the variance of the input signal. It can be seen that when $\delta = 0.2\sigma_x^2$, the ideal FAP and EFEAP algorithms perform indistinguishably. However, when a relatively large regularization parameter δ is employed, the ideal FAP algorithm has slower convergence rate than the EFEAP algorithm.

Fig. 7 shows the convergence and tracking behavior of the EFEAP and original FAP [3,4] algorithms. The impulse response is changed at the middle of the iterations. To achieve the fastest convergence, the step size is set to $\mu = 1.0$. It is seen that the EFEAP has faster convergence rate than the original FAP algorithm. However, the steady-state misadjustment of the original FAP algorithm is smaller than that of the EFEAP algorithm with the same step size, which is an interesting discovery and worthwhile of further study. Also, it is well recognized that the original FAP algorithm suffers from numerical instability especially in a fixed-point implementation because of the propagating errors [13,14]. These facts demonstrate the strengths of the proposed algorithm.

6. Conclusion

This paper provides a new FEF approach to the AP-type algorithms. We firstly introduce the FEF approach to the standard AP algorithm. As a general framework, we also apply the approach to some known AP algorithms, e.g., DCD-AP, APS and MFxAP algorithms. There is an exact mathematical equivalence between the original AP-types algorithms and the proposed fast versions, thereby making the proposed algorithms attractive for a real-time implementation. A detailed comparison (memory size, computational complexity) of different implementation methods in hardware platforms such as FPGA and DSP is our future work.

Acknowledgments

We would like to thank the handling editor and three anonymous reviewers for their constructive suggestions. 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.

References

[1] K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties, Electron. Commun. Jpn. 67-A (1984) 19–27.
[2] D. Zhang, K. Wang, X. Zhang, Blind adaptive affine projection algorithm-based multiuser detector over a multipath fading channel, Signal Process. 90 (2010) 2102–2106.
[3] M. Tanaka, Y. Kaneda, S. Makino, J. Kojima, Fast projection algorithm and its step size control, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, May, 1995, pp. 945–948.
[4] S.L. Gay, S. Tavathia, The fast affine projection algorithm, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, May, 1995, pp. 3023–3026.
[5] S. Douglas, The fast affine projection algorithm for active noise control, in: Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 1995, pp. 1245–1249.
[6] S. Lee, I.Y. Kim, Y.C. Park, Approximated affine projection algorithm for feedback cancellation in hearing aids, Comput. Methods Progr. Biomed. 87 (2007) 254–261.
[7] S.C. Douglas, Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, GA, May, 1996, pp. 1656–1659.
[8] Q.G. Liu, B. Champagne, K.C. Ho, On the use of a modified fast affine projection algorithm in subbands for acoustic echo cancellation, in: Proceedings of the IEEE Digital Signal Processing Workshop, Loen, Norway, September, 1996, pp. 354–357.
[9] S. Oh, D. Linebarger, B. Priest, B. Raghothaman, A fast affine projection algorithm for an acoustic echo cancellation using a fixed-point DSP processor, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, November, 1997, pp. 4121–4124.
[10] F. Albu, J. Kadlec, N. Coleman, A. Fagan, The Gauss–Seidel fast affine projection algorithm, in: Proceedings of the IEEE Workshop on Signal Processing Systems, San Diego, CA, October, 2002, pp. 109–114.
[11] Y. Zakharov, F. Albu, Coordinate descent iterations in fast affine projection algorithm, IEEE Signal Process. Lett. 12 (2005) 353–356.
[12] H. Ding, Fast affine projection adaptation algorithms with stable and robust symmetric linear system solvers, IEEE Trans. Signal Process. 55 (2007) 1730–1740.

[13] K. Chen, J. Lu, X. Qiu, B. Xu, Stability improvement of relaxed FAP algorithm, Electron. Lett. 43 (2007) 1119–1121.
[14] M. Bouchard, Multichannel affine and fast affine projection algorithms for active noise control and acoustic equalization systems, IEEE Trans. Speech Audio Process. 19 (2003) 54–60.
[15] M. Ferrer, A. Gonzalez, M. de Diego, G. Piñero, Fast affine projection algorithms for filtered-x multichannel active noise control, IEEE Trans. Speech Audio Process. 16 (2008) 1396–1408.
[16] M.C. Tsakiris, P.A. Naylor, Fast exact affine projection algorithm using displacement structure theory, in: Proceedings of the 16th International Conference on Digital Signal Processing, Santorini, Greece, July, 2009, pp. 69–74.
[17] G. Rombouts, M. Moonen, A sparse block exact affine projection algorithm, IEEE Trans. Speech Audio Process. 10 (2002) 100–108.
[18] Y. Zakharov, Low complexity implementation of the affine projection algorithm, IEEE Signal Process. Lett. 15 (2008) 557–560.
[19] F. Albu, Y. Zakharov, C. Paleologu, Modified filtered-x dichotomous coordinate descent recursive affine projection algorithm, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Taipei, 2009, pp. 257–260.
[20] F. Albu, A proportionate affine projection algorithm using fast recursive filtering and dichotomous coordinate descent iterations, in: Proceedings of the Signal Processing and Applied Mathematics for Electronics and Communications, Cluj-Napoca, Romania, 2011, pp. 93–96.
[21] J. Ni, Efficient implementation of the affine projection sign algorithm, IEEE Signal Process. Lett. 19 (2012) 24–26.
[22] T. Shao, Y.R. Zheng, J. Benesty, An affine projection sign algorithm robust against impulsive interferences, IEEE Signal Process. Lett. 17 (2010) 327–330.
[23] C. Paleologu, J. Benesty, S. Ciochină, Regularization of the affine projection algorithm, IEEE Trans. Circuits Syst. II 58 (2011) 366–370.
[24] M. Shao, C.L. Nikias, Signal processing with fractional lower order moments: stable processes and their applications, Proc. IEEE 81 (1993) 986–1010.
[25] N.V. George, G. Panda, Advances in active noise control: a survey, with emphasis on recent nonlinear techniques, Signal Process. 93 (2013) 363–377.
[26] S.M. Kuo, D. Morgan, Active noise control systems: algorithms and DSP implementations, John Wiley & Sons, Inc., 1995.
[27] M. Bouchard, Numerically stable fast convergence least-squares algorithms for multichannel active sound cancellation systems and sound deconvolution systems, Signal Process. 82 (2002) 721–736.
[28] F. Yang, M. Wu, J. Yang, A computationally efficient delayless frequency-domain adaptive filter algorithm, IEEE Trans. Circuits Syst. II 60 (2013) 222–226.
[29] A. Carini, G. Sicuranza, Transient and steady-state analysis of filtered-x affine projection algorithm, IEEE Trans. Signal Process. 54 (2006) 665–678.
[30] C. Paleologu, S. Ciochină, J. Benesty, An efficient proportionate affine projection algorithm for echo cancellation, IEEE Signal Process. Lett. 17 (2010) 165–168.
[31] F. Yang, M. Wu, P. Ji, J. Yang, An improved multiband-structured subband adaptive filter algorithm, IEEE Signal Process. Lett. 19 (2012) 647–650.
[32] J. Ni, X. Chen, J. Yang, Two variants of the sign subband adaptive filter with improved convergence rate, Signal Process. 96 (2014) 325–331.