Fast Affine Projection Algorithms for Simplified Structure of Multichannel Active Noise Control

Jiaming Liu [a], Binwen Fan [b] and Fuchen Ding [c]

School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518000, China.

[a] 18601198971@163.com
[b] Corresponding author: 943260971@qq.com
[c] 1191915012@qq.com

Abstract. This article focuses on the application of AP algorithms in active noise control systems. The active noise control system is a real-time tracking noise signal, which is then cancelled with an opposite signal to achieve noise reduction. Therefore, the active noise control system requires high real-time performance. In a multi-channel active noise system, a large amount of calculation is a key factor that causes the real-time performance of the system to decrease. This paper simplifies the system structure based on the existing fast AP algorithm, further reduces the system overhead and achieves the purpose of improving real-time performance. And through the DSP chip, an experimental platform for the active noise control system was built, and the proposed algorithm was validated for practical results.

Key words: Active noise control, affine projection algorithms, fast affine projection algorithms, ANC structure.

INTRODUCTION

The AP algorithm utilizes the strong correlation of input signals and uses data reuse methods to effectively improve the convergence speed of the algorithm [1]. This enables the ANC system to track signals more quickly and has less computational load than the RLS algorithm [2]. The AP algorithm is more often used in echo cancellation systems [3], and it seems to be too computationally expensive to use in an ANC system, especially in multi-channel systems. Therefore, it is of great practical significance to study the fast AP algorithm.

The conjugate iterative fast affine projection algorithm is another FAP that replaces the inverse matrix of the input vector autocorrelation matrix by iteratively solving a set of linear equations with conjugates. The algorithm complexity of CGFAP is 2L+2N2+9N+1, is equivalent to the basic affine projection algorithm. Its complexity is only higher than NLMS [4]. It can be used to solve application problems related to sound processing, especially when combined with DSP design. CGFAP uses a feedback mode [5], which does not accumulate quantization errors in the calculation process. The Gauss-Side Iterative Fast Affine Projection Algorithm (GSFAP) was proposed by Felix Albu et al. in 2002. Because this algorithm is based on the research of CGFAP, it inherits the low complexity and good convergence performance of CGFAP [6]. Stable, and can be well applied to echo cancellation and other applications. In this paper, based on the use of three efficient calculation methods, a simplified structure of the AP algorithm is proposed, further reducing the amount of calculation of the AP algorithm and improving the system's real-time performance.

AP ALGORITHM

The core goal of the AP algorithm is that during each iteration, the coefficient vector $\mathbf{w}(\mathrm{n}+1)$ is infinitely close to the previous time coefficient $\mathbf{w}(\mathrm{n})$, and the error signal is guaranteed to be closest to zero. In order to achieve this goal, using the Lagrange multiplier method, the method converts the constraint problem into a function with no constraints, which is equivalent to collapsing two parallel objects into one function. A specific function facilitates the derivation of the algorithm and has a theoretical basis.

The weight coefficient update formula of the AP algorithm is as follows:

$$\mathbf{w}(\mathrm{n}+1)=\mathbf{w}(\mathrm{n})+\mu\mathbf{A}^{T}(n)[\boldsymbol{A}(n)\mathbf{A}^{T}(n)+\delta\boldsymbol{I}]^{-1}\boldsymbol{e}(n) \tag{1}$$

The $\boldsymbol{A}(n)$ is an N×L matrix, N is the projection depth, and L is the adaptive filter order, defined as:

$$\mathbf{A}^{T}(n)=[\mathbf{x}(n),\mathbf{x}(n-1),\dots,\mathbf{x}(n-N+1)] \tag{2}$$

Then $\boldsymbol{e}(n)$ is an N×1 vector, defined as follows:

$$\boldsymbol{e}(n)=\mathbf{d}(\mathrm{n})-\boldsymbol{A}(n)\mathbf{w}(\mathrm{n}) \tag{3}$$

Where $\mathbf{d}(\mathrm{n})$ is an N×1 vector that consists of the desired signal.

The above is the AP algorithm, where δ is a small normal number and I is an N×N identity matrix. δ is introduced to prevent $\boldsymbol{A}(n)\mathbf{A}^{T}(n)$ from being irreversible.

Fast AP Algorithm

The existing fast AP algorithm is a combination of three efficient strategies. The three methods are respectively an efficient error vector calculation method, an efficient inverse matrix calculation method, and an efficient weight coefficient iteration method. All three methods can further reduce the computational complexity of the algorithm. The formula for each iteration of the multi-channel fast AP algorithm is as follows:

$$v_{i,j,k}(n)=h_{j,k}^{T}\boldsymbol{x}_{i}'(n) \tag{4}$$

The $v_{i,j,k}(n)$ is reference signal $x_i(n)$ filtered by the plant model $\boldsymbol{h}_{j,k}^{T}$ at time n. I J K are respectively the number of reference sensors, actuators and error sensors.

$$\widehat{\boldsymbol{d}}_{k}(n)=e_{k}(n)-\textstyle\sum_{j=1}^{J}\boldsymbol{h}_{j,k}^{T}\boldsymbol{y}_{j}(n) \tag{5}$$

Where $e_k(n)$ is kth errors signal at time n.

$$\widehat{\boldsymbol{e}}_{k}(n)=\begin{bmatrix}\hat{d}_{k}(n)-\boldsymbol{v}_{i,j,k}^{T}(n)\mathbf{w}(n)\\(1-\mu)\bar{\mathbf{e}}(\mathrm{n}-1)\end{bmatrix} \tag{6}$$

The $\widehat{\boldsymbol{e}}_{k}(n)$ is error vector.

$$\boldsymbol{\epsilon}_{i,j,k}(n)=[\boldsymbol{V}_{i,j,k}^{T}(\mathrm{n}-1)\boldsymbol{V}_{i,j,k}(\mathrm{n}-1)+\delta\mathrm{I}]^{-1}\widehat{\boldsymbol{e}}_{k}(n) \tag{7}$$

Where $\boldsymbol{\epsilon}_{i,j,k}(n)$ is the normalized error vector $\boldsymbol{V}_{i,j,k}(n)$ is defined as $[\boldsymbol{v}_{i,j,k}(n)\dots\boldsymbol{v}_{i,j,k}(n-N+1)]^{T}$.

Weight coefficient iteration formula:

$$\widehat{\boldsymbol{w}}_{i,j}(n)=\widehat{\boldsymbol{w}}_{i,j}(n-1)+\mu\textstyle\sum_{k=1}^{K}\boldsymbol{v}_{i,j,k}(\mathrm{n}-\mathrm{N})\mathbf{e}_{i,j,k(N-1)}(n) \tag{8}$$

Output $y_i(n)$ is:

$$y_i(n) = \sum_{i=1}^{I}\left[\widehat{\boldsymbol{w}}_{i,j}^{T}(n)\boldsymbol{x}_{\boldsymbol{i}}(n) - \sum_{k=1}^{K}\mu\bar{\boldsymbol{r}}_{i,j,k}^{T}(n)\bar{\boldsymbol{e}}_{i,j,k}'(n-1)\right] \tag{9}$$

The $\bar{\boldsymbol{r}}_{i,j,k}^{T}(n)$ is the auxiliary (N-1) × 1 vector to iteratively calculate the product $\boldsymbol{v}_{i,j,k}^{T}(n)\bar{\boldsymbol{v}}_{i,j,k}^{T}(n)$.

$$\bar{\boldsymbol{r}}_{i,j,k}(n) = \bar{\boldsymbol{r}}_{i,j,k}(n-1) + v_{i,j,k}(n)\bar{\boldsymbol{v}}_{i,j,k}^{T}(n) - v_{i,j,k}(n-L)\bar{\boldsymbol{v}}_{i,j,k}^{T}(n-L) \tag{10}$$

Where $\bar{\boldsymbol{v}}_{i,j,k}^{T}(n)$ is the (N-1) × 1 vector with the last (N-1) samples of the signal $v_{i,j,k}(n)$.

$$\boldsymbol{e}_{i,j,k}'(n) = \boldsymbol{\epsilon}_{i,j,k}(n) + \begin{bmatrix} 0 \\ \bar{\boldsymbol{e}}_{i,j,k}'(n) \end{bmatrix} \tag{11}$$

Where $\bar{\boldsymbol{e}}_{i,j,k}'(n)$ is the first (N-1) elements of the vertor $\boldsymbol{e}_{i,j,k}'(n)$.
The total number of multiplication is IJK$(7N^2 + 15N + M + L - 2) + IJ(L+1) + MJK + L + N - 1$.
The structure of ANC system using AP algorithm as follows:

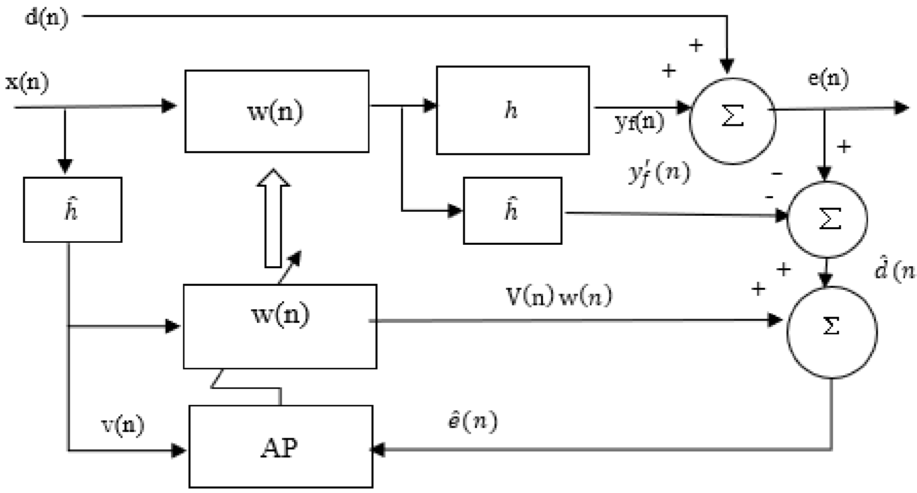

FIG. 1. The structure of ANC system using AP algorithm

AP Algorithm of Simplified Structure

The ANC system using the AP algorithm is more complicated because it involves estimating the expected signal [7].

Considering that the purpose of estimating the desired signal is to find the error vector, if we take the historical signal collected by the error sensor as N elements in the error vector, then we can avoid estimating the expected signal, and change the following as follows:

$$\mathbf{e}(\mathrm{n}) = \begin{bmatrix} d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \\ \bar{\mathbf{d}}(n-1) - \bar{\mathbf{A}}(n-1)\mathbf{w}(n) \end{bmatrix} \approx [e(n)\; e(n-1) \ldots e(n-N+1)]^T \tag{12}$$

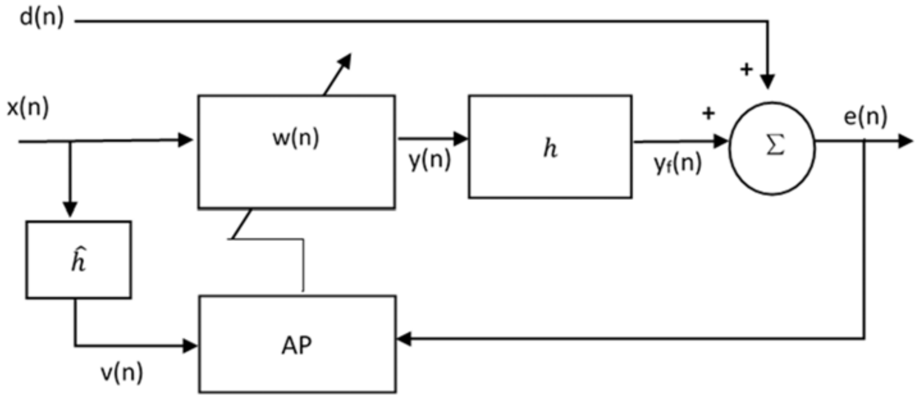

FIG. 2. Simplified Structure of AP algorithm

In this way, the algorithm will further reduce the amount of computation, and compared with the AP algorithm using the efficient error vector calculation method, M+N-1 multiplications are reduced. The simplified structure of the AP algorithm does not require any calculations to calculate the error vector. And its performance and traditional AP algorithm compared to the use of efficient error vector calculation method is almost the same, and later will be displayed in the simulation results. The simplified structure FXAP algorithm is now called the SFXAP algorithm (simplified FXAP). The formula for each iteration of the Fast SFXAP algorithm is as follows:

$$\boldsymbol{\epsilon}_{i,j,k}(n) = [\boldsymbol{V}^{T}_{i,j,k}(\mathrm{n}-1)\boldsymbol{V}_{i,j,k}(\mathrm{n}-1) + \delta\mathrm{I}]^{-1}\boldsymbol{e}_k(n) \tag{13}$$

Where $\boldsymbol{\epsilon}_{i,j,k}(n)$ is the normalized error vector. $\boldsymbol{V}_{i,j,k}(n)$ is defined as $[\boldsymbol{v}_{i,j,k}(n) \dots \boldsymbol{v}_{i,j,k}(n-N+1)]^T$. Weight coefficient iteration formula:

$$\widehat{\boldsymbol{w}}_{i,j}(n) = \widehat{\boldsymbol{w}}_{i,j}(n-1) + \mu \sum_{k=1}^{K} \boldsymbol{v}_{i,j,k}(\mathrm{n}-\mathrm{N})\mathbf{e}_{i,j,k(N-1)}(n) \tag{14}$$

Output $y_i(n)$ is:

$$y_i(n) = \sum_{i=1}^{I}\left[\widehat{\boldsymbol{w}}^{T}_{i,j}(n)\boldsymbol{x_i}(n) - \sum_{k=1}^{K} \mu\bar{\boldsymbol{r}}^{T}_{i,j,k}(n)\bar{\boldsymbol{e}}'_{i,j,k}(n-1)\right] \tag{15}$$

$$\bar{\boldsymbol{r}}_{i,j,k}(n) = \bar{\boldsymbol{r}}_{i,j,k}(n-1) + v_{i,j,k}(n)\bar{\boldsymbol{v}}^{T}_{i,j,k}(n) - v_{i,j,k}(n-L)\bar{\boldsymbol{v}}^{T}_{i,j,k}(n-L) \tag{16}$$

Where $\bar{\boldsymbol{r}}^{T}_{i,j,k}(n)$ is the auxiliary (N-1) × 1 vector to iteratively calculate the product $\boldsymbol{v}^{T}_{i,j,k}(n)\bar{\boldsymbol{v}}^{T}_{i,j,k}(n)$.

$$\boldsymbol{e}'_{i,j,k}(n) = \boldsymbol{\epsilon}_{i,j,k}(n) + \begin{bmatrix} 0 \\ \bar{\boldsymbol{e}}'_{i,j,k}(n) \end{bmatrix} \tag{17}$$

Where $\bar{\boldsymbol{e}}'_{i,j,k}(n)$ is the first (N-1) elements of the vertor $\boldsymbol{e}'_{i,j,k}(n)$.
The total number of multiplication is $\mathrm{IJK}(7N^2 + 15N + L - 2) + IJ(L+1)$.

Now we assume that our system is a 1:2:2 multichannel ANC system with M=128, L=64, and N=5. We can calculate the number of multiplication needed by every iteration, and then form a table as follows:

TABLE 1. Three Scheme comparing

Algorithm	Multiplication per iteration	times
FXAP	$\mathrm{KIJ[M{+}2LN{+}L{+}LN^2{+}N^2{+}O(N^3/2)]}$ $\mathrm{+LIJ+JKM}$	11218
Fast FXAP	$\mathrm{IJK(7N^2+15N+M+L-2)}$ $\mathrm{+IJ(L+1)+MJK+L+N-1}$	2470
SFXAP	$\mathrm{KIJ[M{+}LN{+}L{+}LN^2{+}N^2{+}O(N^3/2)]}$ $\mathrm{+LIJ}$	9426
Fast SFXAP	$\mathrm{IJK(7N^2+15N+L-2)+IJ(L+1)}$	1378

As can be seen from the above table, the simplified structure of the AP algorithm presented in this paper reduces the computational complexity by 16% compared to the traditional structure of the AP algorithm. After applying three efficient calculation methods, the calculation amount of the SFXAP algorithm is further reduced. The Fast SFXAP algorithm reduces the computational complexity by 44.2% over the Fast FXAP algorithm. The Fast SFXAP algorithm has a 87.7% reduction in computation compared to the FXAP algorithm.

SIMULATION RESULTS

To test the performance of the proposed FXAP and SFXAP algorithms, we simulated with a 1:2:2 ANC system. The primary noise was Gaussian white noise, the secondary modeling filter was M = 128, and the adaptive filter was L. = 64 steps. In order to observe the influence of projection depth on the performance of the algorithm, the projection depth N is equal to 5 and 10 respectively.

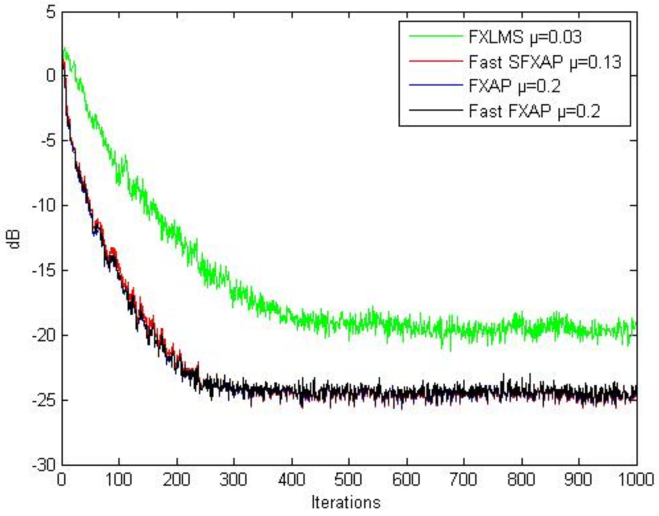

FIG. 3. The first error sensors when N=5

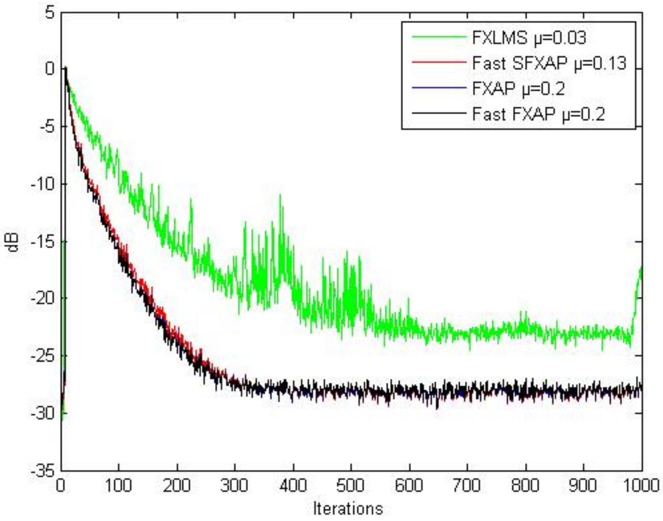

FIG. 4. The second error sensors when N=5

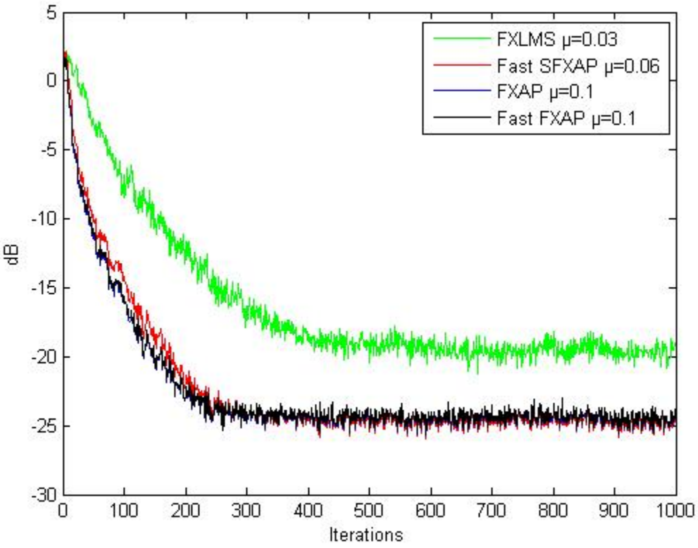

FIG. 5. The first error sensors when N=5

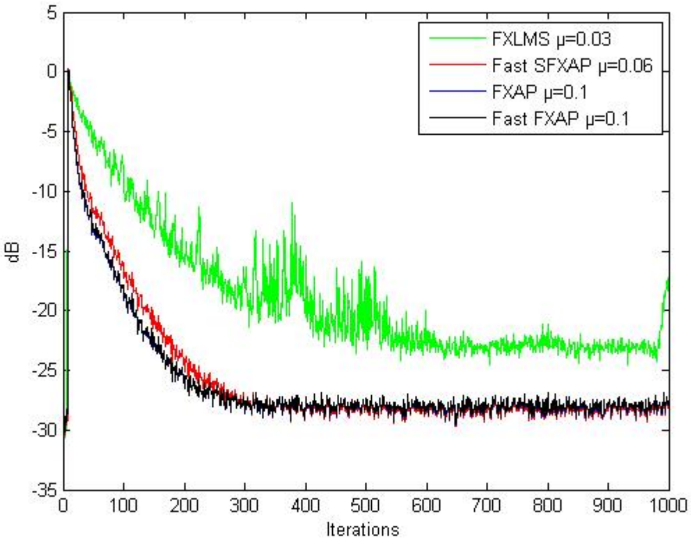

FIG. 6. The second error sensors when N=5

The above is a simulation graph when N is equal to 5 and 10 respectively. The step size μ of each algorithm is different, and the optimal μ in the current condition is selected. The specific values are shown in the figure. It can be seen that the convergence speed of the FXLMS algorithm is the slowest and the error after convergence is the largest, which is not as good as the AP algorithm in two aspects. The two AP algorithms Fast FXAP and Fast SFXAP are not much different when N=5. When N increases, it can be seen that the convergence speed of the FXAP algorithm is faster, because the larger the N, the error vector estimation in the SFXAP algorithm the value will show greater deviations. In general, however, N=10 did not bring greater benefits to the final effect, and N=10 would impose a multiple computational burden. For example, when N=5, the calculation of SFXAP was 1378. When N=10, the SFXAP calculation increased to 3778. The increase in the amount of calculation does not bring corresponding benefits, so a comprehensive view of N=5 is the best choice.

SUMMARY

The purpose of this article is to propose a fast AP algorithm to increase the amount of computation of AP algorithms in practical applications. Currently, there are three kinds of efficient calculation methods to improve the speed of the AP algorithm, but each method is inevitably required to take an estimate of the desired signal. In this paper, the fast AP algorithm, which simplifies the structure, avoids obtaining the desired signal. On the basis of ensuring the performance of the algorithm, it further improves the real-time performance of the AP algorithm and reduces the amount of calculation. In the final simulation experiment, it can be seen that the Fast SXFAP algorithm is not significantly different in performance from the FXAP algorithm, but it is reduced by 87.7% in computation.

REFERENCES

1. Ramesh Chandra Mishra and R Bhattacharjee, Adaptive decision feedback equalizer with set-membership affine projection algorithm, ICIIS. 2016, p. 437 - 442.
2. Y Li and Y Wang, Low-complexity non-uniform penalized affine projection algorithms for active noise control. EUSIPCO. 2017, p. 1275 – 1279.
3. A Gonzalez, M Diego and G Piñero. Steady-state performance of multichannel affine projection algorithms for active noise control. EUSIPCO, 2008, p. 1 – 5.
4. Y Zakharov, Vítor H. Nascimento, Sliding-Window RLS Low-Cost Implementation of Proportionate Affine Projection Algorithms, IEEE/ACM, 2014, Vol: 22, Issue: 12, P.1815 – 1824.

5. Y Xiao, K Hasegawa, An Ikuta and K Noguchi, Statistical properties of the FXLMS-based narrowband active noise control system, EUSIPCO, 2006, p. 1 – 5.
6. Zayyani H. Continuous mixed p-norm adaptive algorithm for system identification[J]. IEEE Signal Processing Letters, 2014, 21(9): 1108-1110.
7. Fernando Gonzalez, Roberto Rossi, German R. Molina and Gustavo, FxLMS and MFxLMS Stability Constrains when used in Active Noise Control, IEEE, 2013, Vol.11, p. 213-217.