MATHEMATICS AND COMPUTATIONAL SCIENCES

Vol 6(1), 2025, pp:89-103

DOI:10.30511/mcs.2025.2043219.1242

ISSN:2717-2708

http://mcs.qut.ac.ir/

Qom University of Technology

Affine projection LMS adaptive algorithm with variable smoothing of weight update matrix

Mehdi Bekrani [1], **Hadi Zayyani** [2]

[1,2]Department of Electrical and Computer Engineering, Qom University of Technology, Qom, Iran

Abstract: Improving the convergence speed of adaptive filters is crucial for enhancing performance in applications involving highly correlated input signals. In this paper, we propose a novel method to improve the convergence performance of the affine projection LMS (AP-LMS) algorithm by incorporating a variable smoothing approach for the weight update matrix. The smoothing parameter is dynamically assigned based on the difference between the instantaneous and smoothed values of the weight update matrix. Simulation results for FIR system modeling demonstrate that the proposed algorithm achieves superior convergence performance in estimating system cœfficients compared to competing adaptive algorithms for both stationary and non-stationary input signals.

Keywords: Adaptive filters; Convergence speed; Affine projection LMS; Variable smoothing; Computational complexity

2020 Mathematics Subject Classification: 62F35; 93D21.

Received: 11 October 2024, **Accepted:** 06 January 2025

1 Introduction

Adaptive filters play a crucial role in various signal processing applications, including echo cancellation, channel equalization, noise reduction, and adaptive beamforming [6]. Their ability to adapt filter parameters in real-time makes them ideal for dynamic environments where signal or noise characteristics frequently change. In scenarios where fixed filters fail to provide optimal performance due to input signal variations, adaptive filters, through real-time algorithmic updates, continuously adjust their weights by comparing the filter's output with a desired signal until the output converges to the target [9].

Among adaptive filtering techniques, the Least Mean Squares (LMS) and Normalized LMS (NLMS) algorithms are popular due to their simplicity, low computational cost, and efficient tracking. However, these algorithms often struggle with poor convergence in the presence of highly correlated input signals and are sensitive to the signal's power spectrum variations [6]. This limitation reduces their practicality for real-time applications.

To address these challenges, the Affine Projection LMS (AP-LMS) algorithm was developed, extending NLMS to improve convergence performance and reduce sensitivity to input signal correlations [4, 11, 14]. AP-LMS is known for its robustness against noise and its ability to handle eigenvalue spreads more effectively [16, 25]. Over the years, several improvements have been proposed for AP-LMS. For example, the Three-Level Clipped AP-LMS (CAP-LMS) offers similar performance to AP-LMS with reduced complexity [4], while the Fast AP-LMS shows improved tracking and convergence comparable to the Recursive Least Squares (RLS) algorithm [7, 5].

[1]Corresponding author: bekrani@qut.ac.ir

[2]zayyani@qut.ac.ir

Various strategies have been introduced to further enhance AP-LMS. For instance, a bias-compensated AP-LMS was proposed to reduce steady-state mean-square error (MSE) in noisy conditions [24]. Moreover, a variable step-size AP-LMS has been developed to increase resilience against impulsive noise [13].

In more advanced approaches, such as [8], both step size and projection order are adjusted dynamically in each iteration to improve the algorithm's performance, although this introduces higher computational demands. Similarly, a recent approach [10] combines outlier detection with affine projection, improving resistance to impulsive noise and reducing complexity through simplified step-size adjustments.

The evolving order-based affine projection sign algorithm [25] introduces an adaptive strategy to adjust projection order based on error power, significantly enhancing both convergence speed and filtering accuracy compared to fixed-order algorithms.

Additionally, subband versions of AP-LMS, which reduce input signal correlation by dividing the signal into frequency subbands, have been proposed to further accelerate convergence in highly correlated input environments [15, 17]. This subband approach decreases spectral variations within each subband, leading to faster convergence [12, 17].

In this paper, a novel AP-LMS algorithm is proposed that uses a variable smoothing function to compute the input correlation matrix. This modification reduces the excess error caused by instantaneous input values during convergence, resulting in faster convergence and improved tracking performance. Simulations demonstrate that the proposed algorithm achieves quicker steady-state convergence and superior tracking compared to existing approaches.

2 Adaptive Filtering Algorithms

Adaptive filtering algorithms can be categorized into various types, including first-order and second-order algorithms, as well as other specialized forms designed for specific applications. First-order adaptive algorithms, like LMS, use the gradient of the cost function to update filter cœfficients, offering low computational complexity but slower convergence, particularly for correlated inputs. In contrast, second-order algorithms, such as LMS-Newton adaptive algorithm and the AP-LMS algorithm, use the second derivative or the inverse of the input autocorrelation matrix, resulting in faster convergence but higher computational complexity and memory requirements. This work focuses on second-order algorithms, which excel in environments with highly correlated inputs. The key feature of second-order adaptive filtering algorithms is their high convergence rate. In the following, we discuss two well-known second-order adaptive filtering algorithms in detail.

2.1 LMS-Newton Adaptive Algorithm

The LMS-Newton algorithm is a sophisticated adaptive algorithm that reduces reliance on eigenvalue spread, making it particularly effective for adaptive filtering applications [2, 20, 21]. By integrating the LMS algorithm with Newton's optimization approach, this algorithm enhances both convergence speed and accuracy. Consequently, it is well-suited for environments characterized by rapidly changing signals or high correlation among input data, where rapid adaptation is crucial. However, the necessity to calculate the Hessian matrix introduces added complexity, rendering it more computationally intensive compared to simpler adaptive algorithms.

To understand its operation, consider an adaptive filter of length N, where the input signal at time n is represented by $\mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-N+1}]^T$. The weight update equation for the LMS-Newton algorithm is given by [6]:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e_n \mathbf{R}_x^{-1} \mathbf{x}_n \tag{2.1}$$

In this equation, μ is the step size that controls the convergence speed, while $\mathbf{R}_x$ serves as an estimate of the input correlation matrix, defined as

$$\mathbf{R}_x = E\{\mathbf{x}_n \mathbf{x}_n^T\}. \tag{2.2}$$

The error signal e_n is calculated as the difference between the desired signal (d_n) and the filter output:

$$e_n = d_n - \mathbf{w}_n^T \mathbf{x}_n. \tag{2.3}$$

In practical scenarios, the inverse of the correlation matrix is not readily available, necessitating estimation methods. Various techniques exist for this purpose, including recursive approaches based on the Matrix Inversion Lemma (MIL) [1, 21], auto-regressive model-based approximations, matrix diagonalization [2], and methods that utilize the inverse of the input power spectrum [18, 19]. Other approaches involve approximating the autocorrelation matrix inverse through Fourier transforms and iterative algorithms [20, 26].

A particularly effective recursive method for estimating $= \mathbf{R}_x^{-1}$ employs the Matrix Inversion Lemma (MIL), expressed as follows [9, 21]:

$$_n = \frac{1}{1-\lambda} \left\{ _{n-1} - \frac{ _{n-1}\mathbf{x}_n\mathbf{x}_n^T \ _{n-1}}{\mathbf{x}_n^T \ _{n-1}\mathbf{x}_n + \frac{1-\lambda}{\lambda}} \right\} \tag{2.4}$$

Here, λ is a small positive constant that regulates the convergence step for the inverse calculation.

The LMS-Newton algorithm's ability to rapidly adapt and maintain accuracy makes it a valuable tool in real-time signal processing applications. By effectively addressing the challenges associated with correlated input signals and computational complexity, it paves the way for improved performance in dynamic environments.

2.2 The AP-LMS Algorithm

The AP-LMS algorithm is a powerful adaptive filtering technique that leverages the correlation of the input signal to enhance the convergence performance. In this algorithm, the input matrix is constructed using M sequential vectors from the input signal, which can be expressed as follows [4, 6]:

$$\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, \ldots, \mathbf{x}_{n-M+1}]. \tag{2.5}$$

This formulation creates an $N \times M$ input matrix that captures the recent history of the input signal, allowing the algorithm to utilize more contextual information for better performance. The corresponding desired signal vector is represented as:

$$\mathbf{d}_n = [d_n, d_{n-1}, \ldots, d_{n-M+1}]^T. \tag{2.6}$$

Based on these definitions, the adaptive filter cœfficients in the AP-LMS algorithm are updated using the following equation:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \mathbf{X}_n \left(\mathbf{X}_n^T \mathbf{X}_n + \psi \mathbf{I}_M \right)^{-1} \mathbf{e}_n, \tag{2.7}$$

where μ is the step size that controls the convergence speed, ψ is a small positive constant that prevents the input matrix from becoming singular, the term $\mathbf{I}_M$ denotes an identity matrix of dimension $M \times M$, and the error vector $\mathbf{e}_n$ is defined as the difference between the desired signal and the output of the adaptive filter, given by:

$$\mathbf{e}_n = \mathbf{d}_n - \mathbf{X}_n^T \mathbf{w}_n. \tag{2.8}$$

A noteworthy feature of the AP-LMS algorithm is its adaptability to the size of M. In the special case where $M = 1$, the AP-LMS algorithm simplifies to the NLMS algorithm. However, as M increases, the algorithm incorporates more input information in the update process, leading to improved convergence speed and enhanced performance [6]. This capability makes the AP-LMS algorithm particularly effective in dynamic environments where input signal characteristics can vary rapidly.

3 The Proposed AP-LMS

The update equation for the AP-LMS algorithm aims to implement a stochastic version of Newton's method [6] (page 176), similar to the LMS-Newton adaptive algorithm. In this context, we introduce a more accurate approximation of Newton's method, termed the AP-LMS Newton (AP-LMS-N) algorithm. In this algorithm, we replace the instantaneous matrix $\mathbf{X}_n^T\mathbf{X}_n + \psi\mathbf{I}_M$ in the cœfficient update equation (2.7) with a new $M \times M$ correlation matrix $\widetilde{\mathbf{R}} = E\{\mathbf{X}_n^T\mathbf{X}_n\}$. This leads to the following update equation:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\mathbf{X}_n\widetilde{\mathbf{R}}^{-1}\mathbf{e}_n. \tag{3.1}$$

Since obtaining a statistical realization of $\widetilde{\mathbf{R}}$ is impractical, we assume that the signals are ergodic and use an estimate for $\widetilde{\mathbf{R}}$ based on time averaging, defined as:

$$\widetilde{\mathbf{R}} \approx \mathbf{R}_n = \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k^T\mathbf{X}_k. \tag{3.2}$$

This estimation can be reformulated as:

$$\mathbf{R}_{n+1} = \lambda_n\mathbf{R}_n + (1-\lambda_n)\mathbf{X}_n^T\mathbf{X}_n, \tag{3.3}$$

where $\lambda_n = \frac{n}{n+1}$ serves as a forgetting factor. Initially, λ_n is small for early input samples, but as n increases, it approaches one.

A primary drawback of using (3.3) to estimate $E\{\mathbf{X}_n^T\mathbf{X}_n\}$ is the delay in adaptation. Since $\mathbf{R}_n$ averages over all past inputs, it may not quickly reflect changes in the input signal's statistics. This lag can lead to slower adaptation in scenarios where the characteristics of the input signal vary over time, thereby reducing the adaptive algorithm's tracking ability. Additionally, as n becomes very large, the sensitivity of $\mathbf{R}_n$ to new values of $\mathbf{X}_n^T\mathbf{X}_n$ approaches zero. This leads to a biased estimate of the correlation matrix $\mathbf{R}_n$ for non-stationary signals. Such bias can degrade the performance of the adaptive filter, especially in dynamic environments.

To address this limitation, we extend the recursive computation in (3.3) by employing different values for $0 < \lambda_n < 1$. Notably, when $\lambda_n = 0$, (3.3) yields instantaneous values of $\mathbf{X}_n^T\mathbf{X}_n$, causing (3.1) to revert to the traditional AP-LMS algorithm introduced in (2.7). As a foundational step toward this extension, we will demonstrate that $\mathbf{R}_n$ provides a more accurate approximation of $E\{\mathbf{X}_n^T\mathbf{X}_n\}$ when $\lambda_n > 0$. We first demonstrate that the variance of the elements in $\mathbf{R}_n$ depends on λ_n and is less than that of the elements in $\mathbf{X}_n^T\mathbf{X}_n$.

Let r_n^{ij} and $\widehat{r}_n^{ij}$ denote the (i,j)th elements of $\mathbf{R}_n$ and $\mathbf{X}_n^T\mathbf{X}_n$, respectively. For improved readability, we will omit the superscripts i and j in the following discussion. According to (3.3), we can express r_{n+1} as follows:

$$r_{n+1} = \lambda_n r_n + (1-\lambda_n)\widehat{r}_n. \tag{3.4}$$

We can rewrite this smoothing equation (3.4) for r_n using a recursive formula:

$$r_n = \lambda_n\widehat{r}_n + \lambda_n(1-\lambda_n)\widehat{r}_{n-1} + \lambda_n(1-\lambda_n)^2\widehat{r}_{n-2} + \cdots. \tag{3.5}$$

This representation indicates that r_n is a weighted sum of the current and past values of $\widehat{r}_n$, with weights that decay exponentially. Assuming for simplicity that the variance of $\widehat{r}_n$ is independent over time, we can express the

variance of r_n as follows:

$$\begin{aligned}
\mathsf{Var}(r_n) &= \lambda_n^2 \mathsf{Var}(\widehat{r}_n) + \lambda_n^2(1-\lambda_n)^2 \mathsf{Var}(\widehat{r}_{n-1}) + \lambda_n^2(1-\lambda_n)^4 \mathsf{Var}(\widehat{r}_{n-2}) + \cdots \\
&= \mathsf{Var}(\widehat{r}_n) \cdot \lambda_n^2 \left(1 + (1-\lambda_n)^2 + (1-\lambda_n)^4 + \cdots\right) \\
&= \mathsf{Var}(\widehat{r}_n) \lambda_n^2 \sum_{k=0}^{\infty} (1-\lambda_n)^{2k} \\
&= \mathsf{Var}(\widehat{r}_n) \cdot \frac{\lambda_n^2}{1-(1-\lambda_n)^2} \\
&= \mathsf{Var}(\widehat{r}_n) \cdot \frac{\lambda_n}{2-\lambda_n}. \qquad (3.6)
\end{aligned}$$

Since $\frac{\lambda_n}{2-\lambda_n} \leq 1$ for $0 < \lambda_n \leq 1$, it follows that:

$$\mathsf{Var}(r_n) \leq \mathsf{Var}(\widehat{r}_n). \qquad (3.7)$$

This indicates that a larger value of λ_n results in less variation for r_n.

Next, we can calculate the expectation of r_n as follows:

$$\mathbb{E}[r_n] = \lambda_n \mathbb{E}[\widehat{r}_n] + \lambda_n(1-\lambda_n)\mathbb{E}[\widehat{r}_{n-1}] + \lambda_n(1-\lambda_n)^2 \mathbb{E}[\widehat{r}_{n-2}] + \cdots . \qquad (3.8)$$

Assuming $\mathbb{E}[\widehat{r}_n] = \eta$ for some constant η, we can substitute this into (3.8):

$$\begin{aligned}
\mathbb{E}[r_n] &= \lambda_n \eta + \lambda_n(1-\lambda_n)\eta + \lambda_n(1-\lambda_n)^2 \eta + \cdots \\
&= \eta \left(\lambda_n + \lambda_n(1-\lambda_n) + \lambda_n(1-\lambda_n)^2 + \cdots\right) \\
&= \eta \lambda_n \left(1 + (1-\lambda_n) + (1-\lambda_n)^2 + \cdots\right) \\
&= \eta \cdot \frac{\lambda_n}{1-(1-\lambda_n)} = \eta \cdot \frac{\lambda_n}{\lambda_n} = \eta. \qquad (3.9)
\end{aligned}$$

Thus, we conclude:

$$\mathbb{E}[r_n] = \mathbb{E}[\widehat{r}_n]. \qquad (3.10)$$

According to (3.10), both r_n and $\widehat{r}_n$ are centered around the same expected value. Furthermore, as noted in (3.7), we have $\mathsf{Var}(r_n) < \mathsf{Var}(\widehat{r}_n)$. This implies that r_n exhibits less variability than $\widehat{r}_n$, resulting in r_n showing less fluctuation around this mean compared to $\widehat{r}_n$.

Therefore, although r_n shares the same expectation as $\widehat{r}_n$, it is more likely to remain closer to this mean value due to its lower variance. This indicates that $\mathbf{R}_n$ is more stable and less susceptible to fluctuations than $\mathbf{X}_n^T \mathbf{X}_n$.

Building on the principles of moving average filtering, we initially propose a smoothed matrix using a constant value for λ_n in the weight update process, such that $0 < \lambda_n < 1$. In this smoothing-based AP-LMS (SAP-LMS) algorithm, the value of λ_n is independent of n. Using a constant λ_n allows the estimate to adapt more quickly to changes in the input signal. The forgetting factor emphasizes recent input data, enabling the correlation matrix estimate to respond effectively to variations in signal characteristics. Additionally, by giving more weight to recent inputs, this approach reduces the bias associated with non-stationary signals, thereby improving the accuracy and performance of the adaptive filter in dynamic environments.

The SAP-LMS algorithm offers a straightforward implementation, as the recursion remains stable and predictable without the need to dynamically adjust λ_n. However, careful tuning of the forgetting factor is essential. A high value of λ_n close to 1 retains more information from the past, which is beneficial for stationary signals but may slow down adaptation. Conversely, a low forgetting factor, closer to 0, increases responsiveness to changes in signal statistics but may lead to less stable performance if the input signal varies rapidly.

To address this limitation, we introduce a novel time-varying parameter λ_n. To maintain tractability, we suggest determining λ_n based on the difference between the instantaneous values of $\mathbf{X}_n^T\mathbf{X}_n$ and the smoothed values $\mathbf{R}_n$. We first define:

$$\delta_n = \|\mathbf{R}_n - \mathbf{X}_n^T\mathbf{X}_n\|. \tag{3.11}$$

where $\|\cdot\|$ denotes the Frobenius norm. When δ_n is large, λ_n should be small and close to zero. Conversely, when δ_n is small, λ_n should be large and close to one. Based on this reasoning, we propose the following relationship for λ_n:

$$\lambda_n = \left(\frac{\delta_n}{1+\delta_n}\right)^p, \tag{3.12}$$

where p indicates the sensitivity of λ_n to δ_n. In the proposed variable smoothing-based AP-LMS (VSAP-LMS) algorithm, for stationary input signals at the beginning of convergence, λ_n is small, while towards the end of convergence, λ_n approaches one. For non-stationary input signals, however, λ_n remains less than one to preserve the algorithm's tractability.

Considering the modified smoothing equation (3.12), this adjustment introduces a dynamic feedback mechanism into the smoothing process, where the weight λ_n depends on how close the smoothed matrix $\mathbf{R}_n$ is to the current observation $\mathbf{X}_n^T\mathbf{X}_n$.

By substituting (3.12) into (3.4) and taking the variance, the recursive equation becomes:

$$\mathrm{Var}(r_{n+1}) = \left(\frac{\delta_n}{1+\delta_n}\right)^{2p}\mathrm{Var}(r_n) + \left(\frac{1}{1+\delta_n}\right)^{2p}\mathrm{Var}(\widehat{r}_n). \tag{3.13}$$

This equation describes how the variance of r_{n+1} evolves over time based on the difference in variances between r_n and $\widehat{r}_n$:

- When $\delta_n \approx 0$: In this scenario, $\lambda_n \to 0$. Consequently, the variance of r_n approaches Var($\widehat{r}_n$, making the smoothed variable more sensitive to the current value $\widehat{r}_n$ for modification.

- When δ_n is significantly different from zero: Here, λ_n approaches 1, causing r_{n+1} to rely more on r_n. This conservative smoothing process results in slower adaptation to unusual data and impulsive noise. Thus, the variance of r_{n+1} remains closer to Var(r_n), preserving the historical smoothing.

Figure 1 illustrates the variation of λ_n concerning δ_n for different values of p. Based on this figure, we can observe that the value of p determines the degree of dependency of λ_n on δ_n. As p increases, the sensitivity of λ_n to δ_n decreases, meaning that λ_n approaches 1 for larger values of δ_n. Conversely, as p decreases towards zero, the sensitivity of λ_n to small values of δ_n significantly increases. Consequently, even a slight increase in δ_n causes λ_n to quickly approach 1.

With this adjustment of λ_n, the proposed VSAP-LMS algorithm emphasizes past data during changes. When δ_n is large, using a larger λ_n close to one means that the update places more emphasis on past data, effectively smoothing out changes. This approach helps maintain stability and prevents overreaction to sudden changes or noise.

In contrast, the algorithm increases sensitivity during steady-state conditions. When δ_n is small, a smaller λ_n close to zero gives more weight to the most recent data, making the filter more responsive to minor fluctuations in the input. This can improve accuracy in tracking steady-state signals.

It is noteworthy that, to calculate the inverse of the matrix $\mathbf{R}_n$ in (3.1), we can update $_n = \mathbf{R}_n^{-1}$ using the matrix inverse lemma. This leads to the following update equation:

$$_{n+1} = \frac{_n}{\lambda_n} - \frac{_n\mathbf{X}\mathbf{X}^T\ _n}{\frac{\lambda_n^2}{1-\lambda_n} + \lambda_n\mathbf{X}^T\ _n\mathbf{X}}. \tag{3.14}$$

This equation provides the inverse of the matrix $\mathbf{R}_n$ based on the matrix inverse lemma and the established definitions.

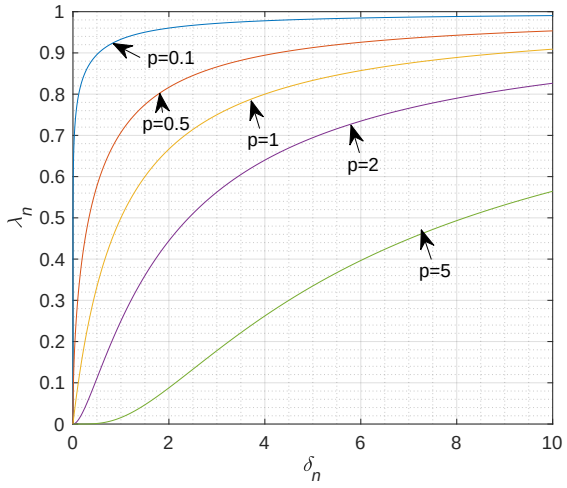

Figure 1: λ_n as a function of δ_n for various values of p.

4 Complexity Analysis

To assess the computational complexity of the algorithms, we examine the number of multiplications and additions required per iteration. In (2.1), the LMS-Newton algorithm necessitates $\frac{1}{3}N^3 + N^2 + 2N$ multiplications and $\frac{1}{3}N^3 + N^2 + N$ additions for each iteration, based on the Gaussian elimination method for matrix inversion.

In (2.7), the AP-LMS algorithm requires $M^2N + M + \frac{1}{3}M^3 + NM + M^2 + N$ multiplications and $M^2(N-1) + M + \frac{1}{3}M^3 + (M-1) + N(M-1) + N$ additions per iteration, again considering Gaussian elimination for matrix inversion.

For the proposed AP-LMS algorithms in (3.1), including AP-LMS-N, SAP-LMS, and VSAP-LMS, the number of operations is significantly lower than LMS-Newton, requiring $\frac{1}{3}M^3 + M^2 + M(N+1)$ multiplications and $\frac{1}{3}M^3 + M^2 + M(N-1)$ addition, assuming Gaussian elimination for matrix inversion.

In smoothing methods, the operational count varies depending on the specific algorithm. In (3.3), the total operations, excluding those needed for λ_n, are $M^2(N+2)$ multiplications and M^2N additions per iteration. Similarly for LMS-Newton, the total operations for recursive computation of $\mathbf{R}_n$, are $N^2(M+2)$ multiplications and N^2M additions per iteration. The computations for λ_n depend on the algorithm type. In AP-LMS-N and LMS-Newton, it requires 1 division and 1 addition per iteration. In SAP-LMS, no operation is required as λ_n is constant. For VSAP-LMS, it necessitates $M^2 + 1$ additions, M^2 multiplications and 1 division per iteration. Although these additional computations make VSAP-LMS slightly more computationally intensive than standard AP-LMS, it remains considerably more efficient than LMS-Newton in terms of computational cost.

Lastly, for computing the error signal in (2.3), we require N multiplications and N additions/subtractions. Additionally, in (2.8), MN multiplications and MN additions/subtractions are necessary. Table 1 summarizes the computational complexity for each algorithm.

Fig. 2 shows the computational complexity of adaptive algorithms for $N = 16$ for values of M from 1 to 10, in terms of the number of additions and multiplications in each iteration. The left side of the figure displays the number of addition operations per iteration, while the right side shows the number of multiplication operations per iteration. It is observed that the AP-LMS method has the lowest computational burden. The proposed methods AP-LMS-N, SAP-LMS, and VSAP-LMS have very similar computational loads, and ultimately, the LMS-Newton method has the highest computational load. For example, for $M = 4$, the number of multiplication operations for the LMS-Newton method

Table 1: Computational Complexity of Algorithms

Algorithm	No. of Additions	No. of Multiplications
AP-LMS	$\frac{1}{3}M^3 + M^2 + M(2N+1)$	$\frac{1}{3}M^3 + M^2 + M(2N-1)$
SAP-LMS	$\frac{1}{3}M^3 + M^2(N+3) + M(2N+1)$	$\frac{1}{3}M^3 + M^2(N+1) + M(2N-1)$
VSAP-LMS	$\frac{1}{3}M^3 + M^2(N+3) + M(2N+1) + 1$	$\frac{1}{3}M^3 + M^2(N+3) + M(2N-1) + 1$
AP-LMS-N	$\frac{1}{3}M^3 + M^2(N+3) + M(2N+1) + 1$	$\frac{1}{3}M^3 + M^2(N+1) + M(2N-1) + 1$
LMS-Newton	$\frac{1}{3}N^3 + N^2(M+3) + 3N + 1$	$\frac{1}{3}N^3 + N^2(M+1) + 2N + 1$

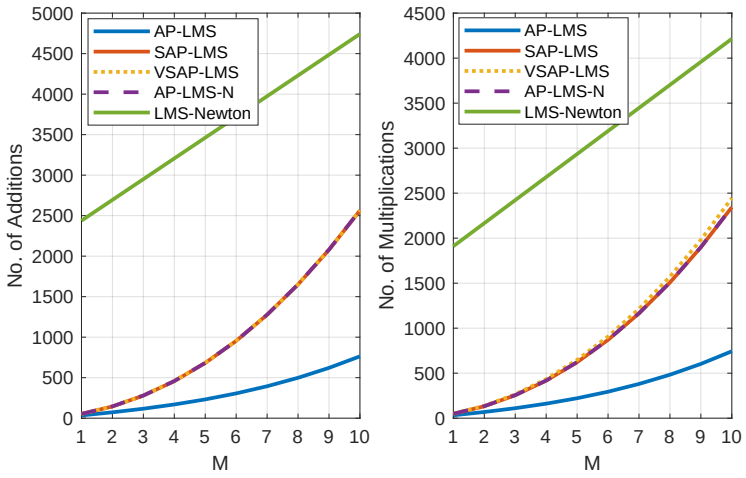

Figure 2: Computational complexity of adaptive algorithms for $N = 16$ in terms of M.

is 2678, while for the algorithms AP-LMS-N, SAP-LMS, VSAP-LMS, and AP-LMS, the counts are 418, 417, 450, and 161, respectively. The increased computational burden of the proposed methods is achieved for improved convergence speed in both steady-state and non-steady-state conditions.

5 Simulation Results

For the simulation, we consider three types of white and colored signals as input signals. In the first case, a white Gaussian noise (WGN) signal is applied for evaluations. In the second case, an autoregressive (AR) signal is considered, which is generated by passing a WGN signal through an AR filter with the following function. Finally, in the third case, a colored signal with a speech-like spectrum is considered, which is created by passing a white Gaussian signal through a colorizing filter with the coefficients [0.3574, 0.9, 0.3574] [3].

The system under consideration for modeling is assumed to have an impulse response of length $N = 16$, with coefficients distributed according to a Gaussian distribution with a mean of zero and a variance of one.

In all simulations, the algorithm was independently executed 100 times, and the average of the results was used. Additionally, WGN with specific Signal-to-Noise Ratio (SNR$= 30$ dB) was added to the desired signal. In all simulations, we assume $M = 4$ unless explicitly stated otherwise. Additionally, we set the value of lambda for SAP-LMS to 0.5, unless explicitly mentioned otherwise.

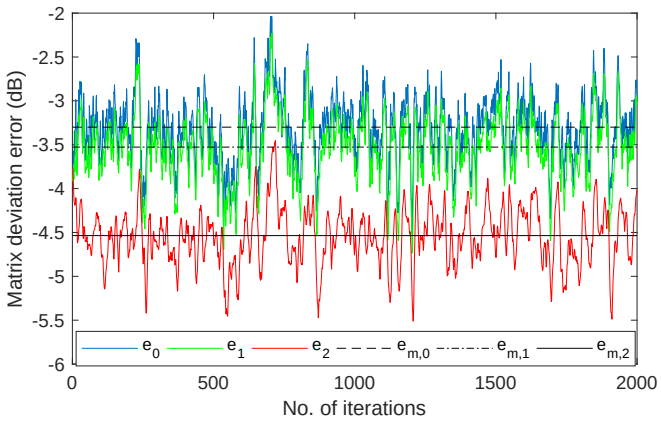

Figure 3: Comparison of Matrix deviation error for AP-LMS, SAP-LMS, and VSAP-LMS.

In the first simulation, we plot the variation of the error between the estimated input matrix $\widehat{\mathbf{R}}_n$ and $\mathbf{R}_x$ in two cases: one where λ_n is constant and the other where it is time-varying. To quantify the error, we define the matrix deviation error as $e_0 = \mathbf{X}_n^T\mathbf{X}_n - \widetilde{\mathbf{R}}$ for AP-LMS, $e_1 = \mathbf{R}_n - \widetilde{\mathbf{R}}$ for SAP-LMS where $\lambda_n = 0.5$, and $e_2 = \mathbf{R}_n - \widetilde{\mathbf{R}}$ for VSAP-LMS where λ_n is time-varying using (3.12). These quantities represents the difference between the estimated input matrix in the weight update process and the matrix $\widetilde{\mathbf{R}}$ derived using (3.2).

Fig. 3 illustrates the errors e_0, e_1, and e_2, over the first 2000 iterations with stationary WGN input. For the VSAP-LMS method, the value of $p = 1$ was used. The average error values over all iterations are also depicted as $e_{m,0}$, $e_{m,1}$, and $e_{m,2}$, for e_0, e_1, and e_2, respectively. It is observed that the error in the proposed method is lower than in the other two methods, demonstrating that the variable smoothing estimation approach yields results closer to $\widetilde{\mathbf{R}}$.

In the next simulation, we examine the effect of p on the convergence behavior of the proposed VSAP-LMS algorithm. The input signal is considered to be a colored speech-like signal. Fig. 4 shows the MSE variation with respect to the number of iterations for different values of p ranging from 1 to 12, with a step size of 0.1 for the algorithm.

As can be observed, increasing p slightly improves the convergence speed. For instance, after 1000 iterations, the MSE reaches -24 dB for $p = 1$, while for $p = 2$, it reaches -24.5 dB, for $p = 4$, it reaches -24.7 dB, and finally, for $p = 12$, it reaches -25 dB.

However, as p increases, the computational complexity of calculating λ also rises. Therefore, for the rest of the simulations, we will consider $p = 1$. It is clear that increasing p would lead to further improvement in the performance of the proposed algorithm.

Fig. 5 illustrates the variations of λ in the VSAP-LMS algorithm for each iteration. As before, the step size is set to 0.1, and the input signal is a colored speech-like signal. It can be observed that λ is variable, fluctuating between 0 and 1.

The next simulation examines the effect of these variations on the convergence speed of the algorithm. In this simulation, we compare the performance of the proposed time-varying λ with a fixed λ. The step size for both the SAP-LMS and VSAP-LMS algorithms is set to 0.1, with the input signal being the same as in the previous experiment.

Fig. 6 compares the convergence behavior of the SAP-LMS algorithm for different values of λ with that of the VSAP-LMS algorithm. It is evident that the value of λ has a significant impact on the convergence speed of the SAP-LMS algorithm. The relationship between convergence speed and λ is complex: during the early stages of convergence, λ values close to 1 yield better results and lead to lower MSE, while in the later stages, λ values closer

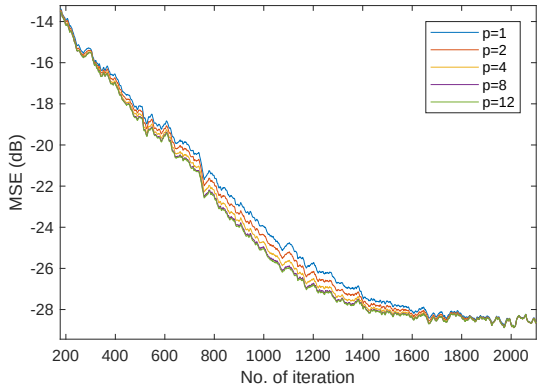

Figure 4: variation of MSE with $M = 4$ for stationary inputs for various amounts of p.

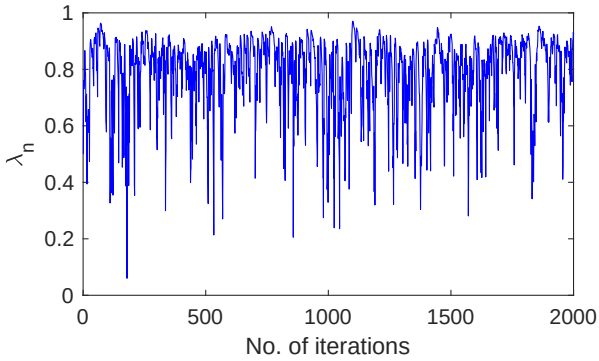

Figure 5: Variation of λ_n in terms of iteration number.

to 0 result in faster convergence and lower MSE.

In the same figure, the effect of the variable λ in the VSAP-LMS algorithm is also shown. Based on this figure, the variable λ approach results in a lower MSE compared to the fixed-λ method in SAP-LMS.

Fig. 7 compares the convergence behavior of the proposed VSAP-LMS algorithm with the AP-LMS, SAP-LMS, and LMS-Newton algorithms for the three mentioned input signals: white noise, AR input, and colored speech-like input. The step size for each algorithm has been adjusted so that all algorithms converge to the same steady-state MSE value, allowing for a fair comparison of their convergence speeds.

It is observed that the SAP-LMS algorithm with $\lambda = 0.5$ shows a convergence behavior similar to AP-LMS, without significant improvement in convergence speed. However, the proposed VSAP-LMS algorithm exhibits a considerable improvement in convergence speed, with performance very close to that of AP-LMS-N. At the same time, the computational complexity of VSAP-LMS is significantly lower than that of AP-LMS-N.

On the other hand, it is clear that the LMS-Newton algorithm achieves much faster convergence across all three input types. However, this comes at the cost of significantly higher computational complexity, which makes the LMS-Newton algorithm impractical for applications with long filter lengths due to its computational demands.

Finally, Fig. 8 compares the convergence speed of the proposed VSAP-LMS algorithm with AP-LMS for different values of M ranging from 1 to 10. We note that, when M =1, the algorithm reduces to the NLMS algorithm. For

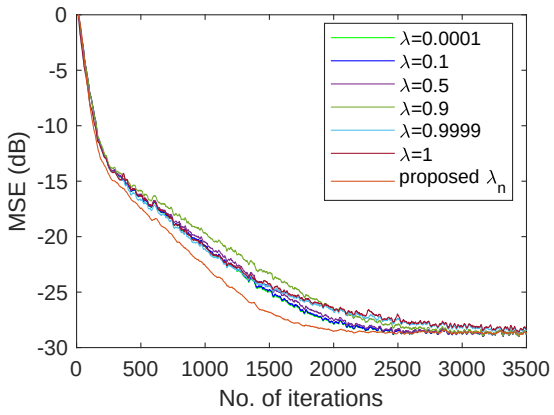

Figure 6: Variation of MSE with various constant λ in terms of iteration number.

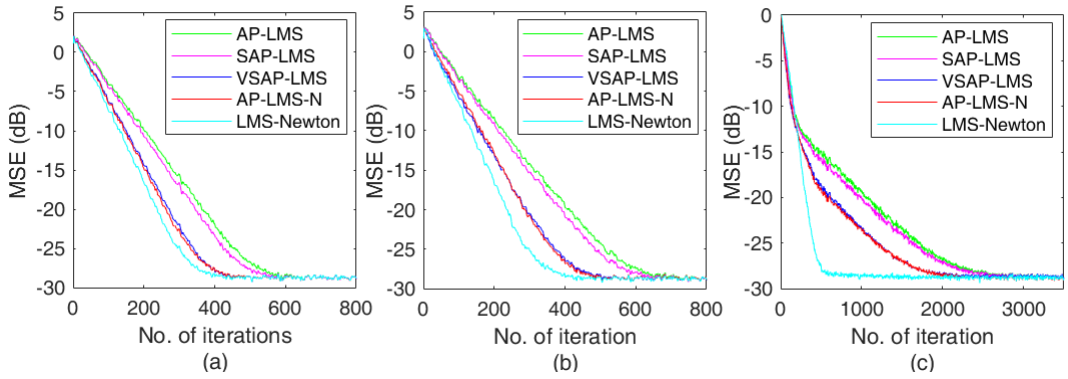

Figure 7: variation of MSE with $M = 4$ for stationary inputs: (a) WGN input, (b) AR input, (c) colored speech-like input..

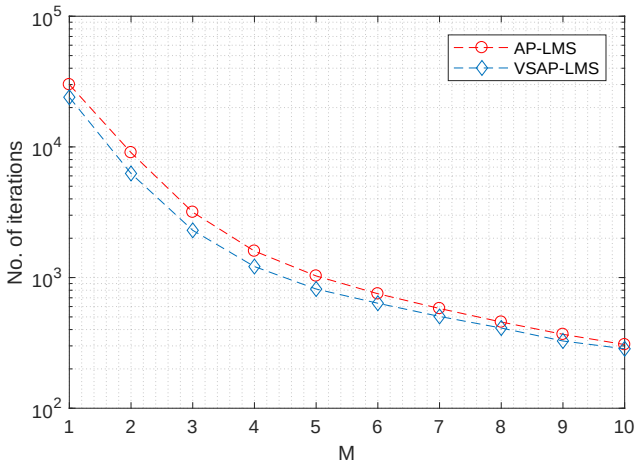

Figure 8: Convergence speed of AP-LMS and VSAP-LMS using number of iterations required to achieve the MSE of $-25\ dB$ for stationary inputs.

comparison, the number of iterations required for each algorithm to reach an MSE level of -25 dB is taken as the benchmark.

It can be observed that the number of iterations required for the VSAP-LMS algorithm is significantly lower than for the AP-LMS algorithm. For example, with $M = 3$, the proposed method requires 2300 iterations, whereas the AP-LMS method requires 3170 iterations, meaning the AP-LMS algorithm takes 870 more iterations to reach the same MSE level.

Here, the signals are considered non-stationary with a length of 4000 samples, in three types: white, AR, and colored with a speech-like spectrum. The variance of these signals starts at 0.5 at the beginning and reaches 14 at the end. An example of a non-stationary colored signal with a speech-like spectrum used in this experiment is shown in Fig. 9. The results of running the algorithm with previous settings are displayed in Fig. 10. Figures a, b, and c show the MSE variations for white, AR, and colored inputs with a speech-like spectrum, respectively. In all these figures, it is observed that the AP-LMS-N method, which uses the approach mentioned in equation 3.2 to calculate

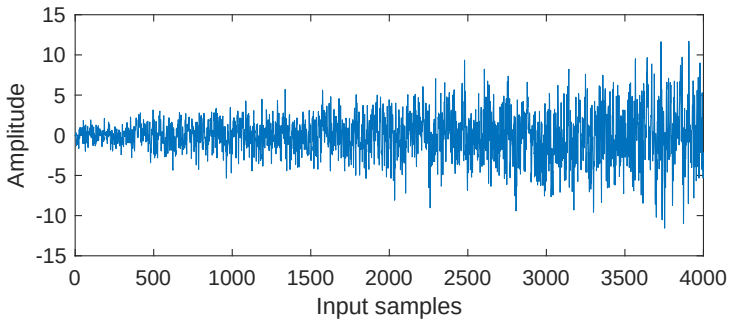

Figure 9: Non-stationary colored input.

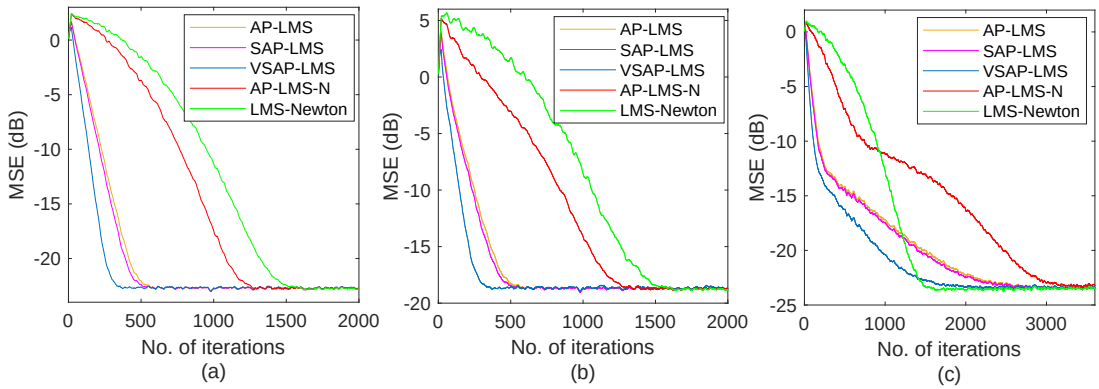

Figure 10: variation of MSE with $M = 4$ for non-stationary inputs: (a) WGN input, (b) AR input, (c) colored speech-like input.

$\mathbf{R}_n$, exhibits a slow convergence rate due to the ineffective updating of $\mathbf{R}_n$. The LMS-Newton method also shows a slow convergence rate for the same reason, as it uses $\mathbf{R}_n$ calculated similarly to equation 3.2. The traditional AP-LMS method yields better results because it utilizes instantaneous values, effectively tracking changes in the input signal statistics. The SAP-LMS method also demonstrates similar performance to AP-LMS. However, the proposed VSAP-LMS method shows superior capability in tracking signal statistics due to the real-time adjustment of the lambda value, resulting in a higher convergence speed. To compare convergence speeds, if we consider the number of iterations required to reach an MSE level of -20 dB in Fig. c, where the input has a speech-like spectrum, it can be seen that the proposed VSAP-LMS method reaches this level after 938 iterations. In comparison, LMS-Newton takes 1266 iterations, SAP-LMS takes 1298 iterations, AP-LMS takes 1508 iterations, and AP-LMS-N takes 2460 iterations. This indicates the superior convergence rate of the proposed VSAP-LMS method in tracking non-stationary signals.

6 Conclusion

This paper presented a novel approach to enhancing the convergence speed and tracking ability of the affine projection LMS (AP-LMS) algorithm through the introduction of variable smoothing in the weight update matrix. The dependence of the smoothing parameter on the difference between the instantaneous and smoothed values is crucial for adapting the algorithm's sensitivity to variations in the input signal, especially in high-correlation scenarios.

Simulation results demonstrated that the proposed method accelerates the convergence rate compared to traditional adaptive algorithms for both stationary and non-stationary input signals. These findings underscore the effectiveness of variable smoothing in optimizing the AP-LMS algorithm, paving the way for its application in real-time adaptive filtering scenarios where rapid convergence is essential. Future work could focus on refining the smoothing strategy further and exploring its implications for other adaptive filtering techniques.

References

[1] Ahmad, M. S., Kukrer, O., Hocanin, A. (2011). Recursive inverse adaptive filtering algorithm. Digital Signal Processing, 21(4), 491–496.

[2] Bekrani, M., Lotfizad, M., Khong, A. W. H. (2010). An efficient quasi-LMS/Newton adaptive algorithm for stereophonic acoustic echo cancellation. In Proc. IEEE Asia Pac. Conf. Circuits Syst. (pp. 684–687).

[3] Bekrani, M., Khong, A. W. H., Lotfizad, M. (2011). A linear neural network based approach to stereophonic acoustic echo cancellation. IEEE Transactions on Audio, Speech, and Language Processing, 19(6), 1743–1753.

[4] Bekrani, M., Bibak, R., Lotfizad, M. (2019). Improved clipped affine projection adaptive algorithm. IET Signal Processing, 13(1), 103–111.

[5] Douglas, S. (1995). The fast affine projection algorithm for active noise control. In Asilomar Conference on Signals, Systems and Computers (pp. 1245–1249).

[6] Farhang-Boroujeny, B. (2013). Adaptive filters: Theory and applications (2nd ed.). John Wiley and Sons.

[7] Gay, S. L. (1993). A fast converging, low complexity adaptive filtering algorithm. In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (pp. 1245–1249).

[8] Gonzalez, A., Ferrer, M., Albu, F., Diego, M. (2012). Affine projection algorithms: Evolution to smart and fast algorithms and applications. In Proc. 20th European Signal Processing Conference (pp. 1965–1969).

[9] Haykin, S. (2014). Adaptive filter theory (5th ed.). Pearson. https://www.pearson.com/

[10] Hou, Y., Li, G., Zhang, H., Zhang, H., Zhao, J. (2024). Affine projection algorithm with outlier detection for robust filtering. IEEE Transactions on Circuits and Systems II: Express Briefs.

[11] Huang, F., Zhang, J., Zhang, S. (2016). Combined-step-size affine projection sign algorithm for robust adaptive filtering in impulsive interference environments. IEEE Transactions on Circuits and Systems, 63(5), 493–497.

[12] Husøy, J. H., Abadi, M. S. E. (2017). On the convergence speed of the normalized subband adaptive filter: Some new insights and interpretations. In International Symposium on Signals, Circuits and Systems.

[13] Koike, S. I. (2016). Analysis of affine projection normalized correlation algorithm. In International Symposium on Intelligent Signal Processing and Communication Systems.

[14] Lee, K., Bæk, Y., Park, Y. (2015). Nonlinear acoustic echo cancellation using a nonlinear postprocessor with a linearly constrained affine projection algorithm. IEEE Transactions on Circuits and Systems II: Express Briefs, 62(9), 881–885.

[15] Liu, D., Zhao, H. (2023). Affine projection sign subband adaptive filter algorithm with unbiased estimation under system identification. IEEE Transactions on Circuits and Systems II: Express Briefs, 70 (3), 1209–1213.

[16] Nita, V. A., Dobre, R. A., Ciochina, S., Paleologu, C. (2017). Improved convergence model of the affine projection algorithm for system identification. In International Symposium on Signals, Circuits and Systems.

[17] Petraglia, M. R., Haddad, D. B., Marques, E. L. (2016). Affine projection subband adaptive filter with low computational complexity. IEEE Transactions on Circuits and Systems II: Express Briefs, 63(10), 989–993.

[18] Poletti, M. A., Teal, P. D. (2021). A superfast Toeplitz matrix inversion method for single- and multi-channel inverse filters and its application to room equalization. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29(6), 3144–3157.

[19] Salman, M. S., Kukrer, O., Hocanin, A. (2012). A fast implementation of quasi-Newton LMS algorithm using FFT. In Proc. Inf. Commun. Technol. Appl. (pp. 510–513).

[20] Salman, M. S., Kukrer, O., Hocanin, A. (2020). A fast quasi-Newton adaptive algorithm based on approximate inversion of the autocorrelation matrix. IEEE Access, 8, 47877–47887.

[21] Tsinos, C. G., Diniz, P. S. R. (2019). Data-selective LMS-Newton and LMS-Quasi-Newton algorithms. In Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (pp. 4848–4852).

[22] Wu, M., Xiong, N., Vasilakos, A. V., Leung, V. C. M., Chen, C. L. P. (2022). RNN-K: A reinforced Newton method for consensus-based distributed optimization and control over multiagent systems. IEEE Transactions on Cybernetics, 52(5), 4012–4026.

[23] Yuan, W. Y., Zhou, Y., Huang, Z. Y., Liu, H. Q. (2015). A VAD-based switch fast LMS/Newton algorithm for acoustic echo cancellation. In Proc. IEEE Int. Conf. Digit. Signal Process. (pp. 967–970).

[24] Zhao, H., Zheng, Z. (2016). Bias-compensated affine-projection-like algorithms with noisy input. Electronics Letters, 52(9), 712–714.

[25] Zhao, J., Ni, X., Li, Q., Tang, L., Zhang, H. (2024). Evolving order based affine projection sign algorithm for enhanced adaptive filtering. IEEE Signal Processing Letters, 31, 1530–1534.

[26] Zorkun, A. E., Salas-Natera, M. A., Martínez Rodriguez-Osorio, R. (2022). Improved iterative inverse matrix approximation algorithm for zero forcing precoding in large antenna arrays. IEEE Access, 10, 100964–100975.