Journal Européen des Systèmes Automatisés
Vol. 57, No. 5, October, 2024, pp. 1259-1272
Journal homepage: http://iieta.org/journals/jesa

Advancements and Applications of Adaptive Filters in Signal Processing

C.T. Somefun, S.A. Daramola, T.E. Somefun*

Department of Electrical and Information Engineering, Covenant University, Ota 112104, Nigeria

Corresponding Author Email: tobi.somefun@covenantuniversity.edu.ng

https://doi.org/10.18280/jesa.570502

Received: 10 April 2024
Revised: 22 July 2024
Accepted: 8 August 2024
Available online: 28 October 2024

Keywords:
adaptive filters, signal processing, convergence, computational efficiency

ABSTRACT

This paper provides an in-depth exploration of adaptive filters, indispensable tools in signal processing for their ability to dynamically adjust parameters and optimize performance in varying environments. Delving into prominent methodologies such as the Least Mean Squares (LMS), Normalized Least Mean Squares (NLMS), Recursive Least Squares (RLS), Gradient Adaptive Lattice (GAL), and Fractional Tap-Length (FTL) algorithms, the study elucidates their unique characteristics, advantages, and considerations. LMS is known for its simplicity and computational efficiency, while NLMS improves convergence speed and robustness to input signal power variations. RLS offers rapid convergence and robustness through recursive estimation, distinct from the iterative approaches of LMS and NLMS. GAL employs a lattice structure for efficient parameter estimation and numerical stability, and FTL dynamically adjusts tap-lengths for enhanced performance. The paper underscores the significance of adaptive filters in diverse applications, from telecommunications and audio processing to biomedical signal analysis and control systems, highlighting their role in driving innovation and advancement in signal processing technology.

1. INTRODUCTION

Adaptive filters are essential tools in signal processing, capable of adjusting their parameters automatically to optimize performance in varying environments [1]. They find extensive applications in diverse fields such as telecommunications, audio processing, biomedical signal processing, and control systems. Unlike fixed filters with predetermined coefficients, adaptive filters possess the ability to continuously update their coefficients based on the input signal and a defined criterion, typically to minimize error or achieve desired performance metrics [2].

Fixed filters operate with static coefficients, designed for specific, unchanging environments. These filters are effective in scenarios where the signal characteristics are known and constant. However, they lack the flexibility to adapt to varying signal conditions, making them less suitable for dynamic environments [3].

In contrast, adaptive filters embody a dynamic paradigm, enabling automated parameter adjustment to optimize performance across a spectrum of environments [4]. With a profound impact spanning telecommunication, audio processing, biomedical signal processing, and control systems, their versatility underscores their indispensability in modern technological landscapes. Adaptive filters iteratively refine their parameters in response to evolving input signals and predefined criteria, whether it be error minimization or the attainment of specific performance metrics [5].

The significance of adaptive filters lies in their ability to navigate through the inherent variability and complexity present in real-world signals and systems [6]. In telecommunications, for instance, where the quality of transmitted signals can be compromised by channel distortions and noise, adaptive filters serve as guardians of fidelity, dynamically adjusting their coefficients to mitigate such impairments and ensure optimal signal reception. Likewise, in audio processing applications, where the acoustic environment can vary drastically, adaptive filters offer a means to tailor audio output in real-time, compensating for reverberations, echoes, and other distortions to deliver a pristine auditory experience [7-11]. Their utility extends further into the domain of biomedical signal processing [12, 13], where the precise extraction of physiological information from noisy bio-signals is paramount for accurate diagnosis and treatment. Here, adaptive filters play a pivotal role in enhancing signal clarity and fidelity, facilitating the extraction of meaningful insights amidst the noise.

The inherent adaptability of these filters not only confers them with resilience in the face of environmental variability but also imbues them with the capacity for continuous improvement and optimization. Through iterative adjustments guided by error signals or predefined performance criteria, adaptive filters can refine their operation over time, honing in on the most effective parameter configurations for a given context. This adaptative learning capability endows them with a form of intelligence, allowing them to autonomously adapt to changing conditions and maintain peak performance levels. Moreover, their adaptability extends beyond mere parameter adjustment, with some adaptive filters capable of dynamically modifying their filter structures in response to varying signal characteristics. Such flexibility enables them to effectively accommodate transient phenomena, non-stationary signals,

and other dynamic aspects inherent in real-world applications, further enhancing their utility and relevance [14-16].

The evolution of adaptive filters is intricately intertwined with advances in signal processing algorithms, computational techniques, and hardware architectures. As research endeavors push the boundaries of optimization and efficiency, novel adaptive filtering methodologies continue to emerge, offering enhanced performance, reduced computational overhead, and broader applicability. Machine learning techniques, in particular, have begun to intersect with adaptive filtering, offering avenues for harnessing vast datasets to inform filter adaptation and decision-making processes [17]. Deep learning architectures, reinforcement learning frameworks, and other data-driven approaches hold promise for unlocking new frontiers in adaptive filtering, empowering filters with greater adaptability, robustness, and intelligence.

In summary, adaptive filters stand as pillars of adaptability and resilience in the domain of signal processing, offering a dynamic framework for optimizing performance across diverse and dynamic environments. Their ability to autonomously adjust parameters, learn from experience, and adapt to changing conditions underscores their significance in applications ranging from telecommunications and audio processing to biomedical signal processing and beyond. As research continues to push the boundaries of innovation, the future holds promise for further advancements in adaptive filtering methodologies, ushering in an era of heightened efficiency, intelligence, and versatility in signal processing applications.

2. METHODS OF ADAPTIVE FILTERS

Before delving into the intricacies of adaptive filtering methods, it is crucial to understand the fundamental role these techniques play in modern signal processing. Adaptive filters represent a dynamic paradigm within the realm of signal processing, offering the capability to autonomously adjust their parameters in response to changing input signals and environmental conditions [18]. This adaptability endows them with versatility and resilience, making them indispensable tools across a myriad of applications, including telecommunications, audio processing, biomedical signal processing, and control systems [19-23]. The following are five prominent methods of adaptive filters, each offering unique advantages and characteristics in addressing the diverse challenges encountered in signal processing tasks.

2.1 Least Mean Squares (LMS) algorithm

The Least Mean Squares (LMS) algorithm stands as one of the cornerstone methodologies in the realm of adaptive filters, renowned for its simplicity, efficiency, and widespread applicability. Rooted in the principles of stochastic gradient descent, the LMS algorithm operates on the premise of iteratively adjusting filter coefficients to minimize the mean squared error between the desired response and the actual output [24]. This iterative process unfolds in a manner akin to a learning mechanism, with the filter gradually refining its parameters based on the characteristics of the input signal and the prevailing error signal.

At the heart of the LMS algorithm lies the notion of gradient descent, wherein the filter coefficients are updated in a direction opposite to the gradient of the mean squared error surface. This update direction is determined by multiplying the error signal with the input signal, thus reflecting the correlation between the input and the error. Consequently, coefficients associated with input signals exhibiting higher correlation with the error signal undergo more substantial adjustments, driving the filter towards convergence.

The adaptation mechanism in the LMS algorithm is governed by a parameter known as the step size or learning rate, denoted by μ [25]. This parameter dictates the magnitude of coefficient updates in each iteration, thereby influencing the convergence speed and stability of the algorithm. A smaller step size leads to more conservative updates, ensuring stability but potentially slowing down convergence, while a larger step size accelerates convergence but risks overshooting the optimal solution and introducing instability.

Despite its simplicity and computational efficiency, the LMS algorithm is not without its limitations [26, 27]. One prominent challenge is its susceptibility to slow convergence, particularly in scenarios with highly correlated or slowly varying input signals. Moreover, the performance of the LMS algorithm hinges significantly on the appropriate selection of the step size parameter. Choosing an inadequate step size can result in suboptimal convergence rates, oscillations, or even divergence of the algorithm.

Nevertheless, the LMS algorithm finds extensive applications across diverse domains due to its ease of implementation and robust performance in many practical scenarios. In telecommunications, for instance, it is employed for echo cancellation, channel equalization, and adaptive interference suppression. Similarly, in audio processing, the LMS algorithm is utilized for noise cancellation, adaptive equalization, and acoustic echo suppression. Its versatility extends into biomedical signal processing, where it aids in artifact removal, adaptive filtering of physiological signals, and adaptive control of medical devices [28, 29].

The Least Mean Squares (LMS) algorithm is a fundamental technique in adaptive filtering, a powerful tool used in various signal processing applications [24]. Adaptive filters are designed to adjust their characteristics (weights) automatically in response to the incoming signal. The LMS algorithm achieves this by minimizing the mean squared error (MSE) between the desired signal (the one you want) and the actual output of the filter. The following are the five steps of LMS working operation:

(1). Initialization: The filter starts with a set of initial weights, which can be random values.

(2). Signal Processing: The filter takes an input signal and processes it using the current weights.

(3). Error Calculation: The difference between the desired signal (d(k)) and the filter output (y(k)) is calculated as the error signal (e(k)) at each time step (k). This error represents how well the filter is performing.

(4). Weight Update: The LMS algorithm uses the current error to update the filter weights. The update is proportional to the negative of the instantaneous gradient of the mean squared error with respect to the weights. In simpler terms, the weights are adjusted in a direction that reduces the error. The update rule is:

$$w(k+1) = w(k) + \eta * e(k) * x(k)$$

where:

w(k) is the weight vector at time step k;

w(k+1) is the updated weight vector at time step k+1;

η (eta) is the learning rate, a small positive value that controls the step size of the weight update. Choosing a proper learning rate is crucial for convergence and stability;

e(k) is the error signal at time step k;

x(k) is the input signal vector at time step k.

(5). Repeat: Steps ii-iv are repeated for each new input signal. Over time, the LMS algorithm continuously adjusts the weights, aiming to minimize the mean squared error and improve the filter's performance in approximating the desired signal.

The flowchart for LMS is given in Figure 1. It provides a visual representation of the iterative process of the LMS algorithm in an adaptive filter. You can see how the error signal drives the weight update, leading the filter to improve its performance over time.

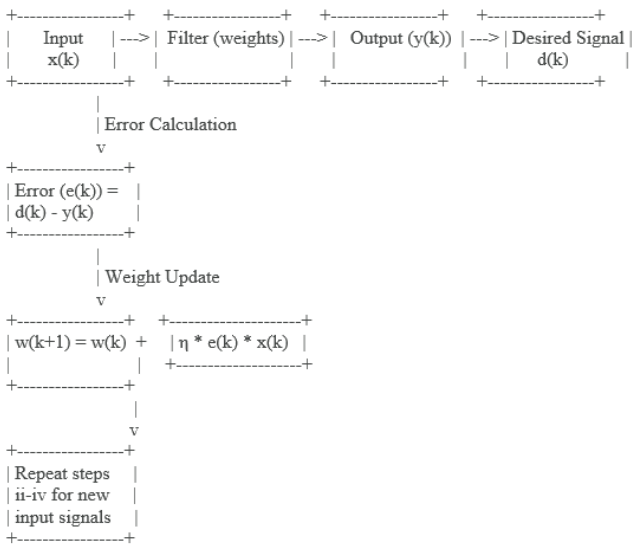

Figure 1. LMS flowchart

(1). Input: The flowchart starts with the input signal x(k).

(2). Filter: The input signal is processed by the filter with its current weights w(k).

(3). Output: The filter produces an output signal y(k).

(4). Error Calculation: The error signal e(k) is calculated as the difference between the desired signal d(k) and the filter output y(k).

(5). Weight Update: The LMS algorithm uses the error signal e(k) and the learning rate η to update the weight vector w(k). The update rule is shown in the formula on the flowchart.

(6). Repeat: Steps 2-4 are repeated for each new input signal x(k). Over time, the filter weights are continuously adjusted based on the error, aiming to minimize the difference between the desired and actual output.

2.1.1 Advantages of LMS

(1). Low computational complexity: The LMS algorithm requires minimal calculations per iteration, making it suitable for real-time applications [24].

(2). Simple implementation: The algorithm is straightforward to implement in hardware or software [24].

(3). Convergence**:** Under certain conditions, the LMS algorithm guarantees convergence to a stable state [24].

2.1.2 Disadvantages of LMS

(1). Slow convergence: The convergence rate of LMS can be slow, especially for non-stationary signals or complex filtering tasks [24].

(2). Mis-adjustment noise: The LMS algorithm introduces a small amount of noise into the filter output due to the stochastic nature of the update rule [24].

(3). Sensitive to learning rate: The choice of the learning rate significantly impacts the performance and stability of the LMS algorithm. A small learning rate can lead to slow convergence, while a large learning rate can cause instability [24].

2.1.3 Applications of LMS

(1). Echo cancellation in telecommunications [24].

(2). Noise cancellation in headphones and audio systems [24].

(3). System identification and adaptive control [24].

(4). Signal prediction and channel equalization [24].

The Least Mean Squares (LMS) algorithm serves as a foundational methodology in adaptive filtering, offering a balance between simplicity and effectiveness. Its iterative nature and gradient descent-based adaptation mechanism enable efficient parameter adjustment, making it well-suited for a wide array of signal processing applications. While challenges such as slow convergence and sensitivity to step size parameters exist, the LMS algorithm remains a valuable tool in the signal processing engineer's arsenal, driving advancements in telecommunications, audio processing, biomedical engineering, and beyond. The LMS algorithm is a basic building block for more advanced adaptive filtering algorithms that address some of its limitations, such as faster convergence or improved noise performance. There are variants of the LMS algorithm, such as the Normalized LMS (NLMS) algorithm, that address specific challenges in certain applications.

2.2 Normalized Least Mean Squares (NLMS) algorithm

The Normalized Least Mean Squares (NLMS) algorithm represents a refinement of the classic LMS algorithm, designed to address some of its inherent limitations while preserving its simplicity and computational efficiency [30]. At its core, NLMS shares the fundamental principles of its predecessor, operating on the basis of iteratively adjusting filter coefficients to minimize the mean squared error between the desired response and the actual output. However, NLMS introduces a normalization mechanism to adaptively adjust the step size parameter, offering improved convergence characteristics and robustness to variations in signal power [31].

The key innovation in the NLMS algorithm lies in its adaptive step size, which is computed based on the power of the input signal. By normalizing the step size with respect to the input signal power, NLMS effectively scales the magnitude of coefficient updates according to the amplitude of the input signal. This adaptive scaling mechanism ensures that larger updates are applied to coefficients associated with weaker input signals, facilitating faster convergence while maintaining stability and preventing overshoot [32].

2.2.1 Advantages over LMS

One of the primary advantages of the NLMS algorithm is

its faster convergence speed compared to the traditional LMS algorithm. The normalization of the step size allows NLMS to adjust more rapidly to changes in the input signal, reducing the number of iterations required to reach a steady-state solution. This characteristic is particularly beneficial in dynamic environments where signal properties can vary significantly over time.

Another significant advantage of NLMS is its robustness to variations in input signal power. In scenarios where the input signal exhibits fluctuations in amplitude, the LMS algorithm's performance can be adversely affected due to the fixed step size parameter. NLMS mitigates this issue by normalizing the step size with respect to the input signal power, ensuring consistent performance even when the signal power varies. This makes NLMS well-suited for applications with dynamic or unpredictable input signals [33].

The Normalized Least Mean Squares (NLMS) algorithm is an improvement upon the Least Mean Squares (LMS) algorithm, another cornerstone of adaptive filtering. While LMS offers simplicity and low computational cost, it can suffer from slow convergence and sensitivity to the learning rate. NLMS addresses these limitations, making it a powerful tool for various signal processing applications. Similar to LMS, NLMS aims to minimize the mean squared error (MSE) between the desired signal and the filter output [29]. However, NLMS incorporates a normalization step to address the limitations of LMS. It achieves this with the following steps:

(1). Initialization: Similar to LMS, the filter starts with a set of initial weights, which can be random values.
(2). Signal Processing: The filter takes an input signal and processes it using the current weights.
(3). Error Calculation: The difference between the desired signal (d(k)) and the filter output (y(k)) is calculated as the error signal (e(k)) at each time step (k).
(4). Normalization: Here's the key difference: NLMS calculates a normalization factor based on the energy (power) of the input signal vector (x(k)). This factor helps to prevent the learning rate from being overly influenced by large or small input signals.
(5). Weight Update: The filter weights are updated based on the error signal, but with the inclusion of the normalization factor. The update rule is:

$$w(k+1) = w(k) + [\ \eta(k) / ||x(k)||^2] * e(k) * x(k)$$

where:

w(k) is the weight vector at time step k

w(k+1) is the updated weight vector at time step k+1

η(k) is a step-size parameter that can be fixed or variable (discussed later)

$||x(k)||^2$ is the squared norm (energy) of the input signal vector at time step k

e(k) is the error signal at time step k

x(k) is the input signal vector at time step k

(6). Repeat: Steps 2-5 are repeated for each new input signal.

The flowchart for LMS is given in Figure 2. It highlights the additional step of normalization in NLMS compared to LMS. The normalization factor ensures that the learning rate is less sensitive to the input signal's power, leading to more robust and faster convergence of the filter weights.

(1). Input: The flowchart starts with the input signal x(k).
(2). Filter: The input signal is processed by the filter with its current weights w(k).
(3). Output: The filter produces an output signal y(k).
(4). Error Calculation: The error signal e(k) is calculated as the difference between the desired signal d(k) and the filter output y(k).
(5). Normalization: This is the key step in NLMS. The squared norm (energy) of the input signal vector x(k) is calculated, providing a normalization factor.
(6). Weight Update: The NLMS algorithm uses the error signal e(k), the normalization factor, and the step-size parameter η(k) to update the weight vector w(k). The update rule is shown in the formula on the flowchart.
(7). Repeat: Steps ii-vi are repeated for each new input signal x(k). The normalization helps the filter adapt effectively and achieve faster convergence compared to LMS.

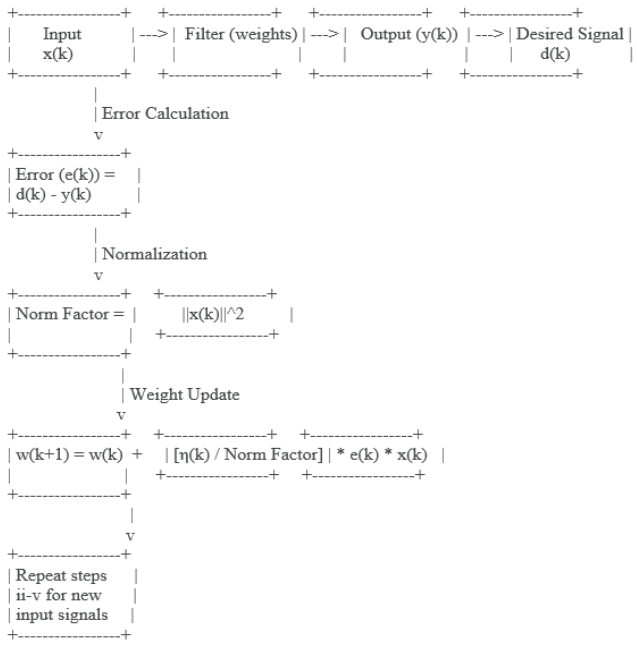

Figure 2. NLMS flowchart

One of the primary advantages of the NLMS algorithm lies in its robustness to variations in signal power, making it well-suited for applications with dynamic or unpredictable input signals [33]. Unlike the LMS algorithm, which requires manual tuning of the step size parameter, NLMS adapts its step size dynamically based on the input signal characteristics, alleviating the need for manual parameter adjustment and offering improved convergence performance.

However, despite its advantages, the NLMS algorithm is not without its limitations. In scenarios where the input signal exhibits rapid changes in power or non-stationary behavior, NLMS may suffer from slow convergence or excessive adaptation due to its reliance on instantaneous power measurements.

2.2.2 Advantages of NLMS

(1). Faster Convergence: The normalization step in NLMS helps to overcome the slow convergence issue of LMS, especially for non-stationary signals or applications with rapidly changing desired responses.
(2). Reduced Mis-adjustment Noise: NLMS can lead to less noise introduced into the filter output compared to LMS due to the normalization.
(3). Less Sensitive to Learning Rate: The normalization

factor helps to mitigate the sensitivity to the learning rate selection compared to LMS. Even with a non-optimal learning rate, NLMS can often achieve better performance than LMS.

2.2.3 NLMS variations

(1). Fixed Step-Size NLMS: A fixed value is used for η(k) throughout the adaptation process. This offers simplicity but may not be optimal for all scenarios.
(2). Variable Step-Size NLMS: η(k) is adjusted dynamically based on the input signal power or other criteria. This can lead to faster convergence and improved tracking capabilities for non-stationary signals.

2.2.4 Applications of NLMS

Similar to LMS, NLMS finds applications in various areas, including:

(1). Echo cancellation in telecommunications.
(2). Noise cancellation in headphones and audio systems.
(3). System identification and adaptive control.
(4). Signal prediction and channel equalization.

NLMS is generally computationally more expensive than LMS due to the normalization step. However, the potential benefits in convergence speed and robustness often outweigh this drawback. There are further advanced algorithms based on NLMS that address specific challenges, such as improved tracking of non-stationary signals or reduced computational complexity.

2.3 Recursive Least Squares (RLS) algorithm

The Recursive Least Squares (RLS) algorithm stands as a powerful methodology within the realm of adaptive filters, prized for its rapid convergence, robustness, and ability to handle non-stationary signals [34]. Unlike the iterative nature of the Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS) algorithms, RLS operates by recursively updating filter coefficients based on the entire history of input signals and corresponding desired responses. This recursive approach enables RLS to adapt swiftly to changing signal conditions while maintaining numerical stability and precision.

At the heart of the RLS algorithm lies the principle of recursive estimation, wherein the filter coefficients are updated based on a recursive formulation derived from the least squares criterion. Instead of relying solely on the current input signal and error, RLS considers the entire history of observations, incorporating past information into the estimation process. This recursive nature allows RLS to achieve rapid convergence by leveraging a comprehensive understanding of the signal dynamics accumulated over time.

2.3.1 Recursive estimation vs. iterative approach

The iterative approach of LMS and NLMS algorithms updates the filter coefficients incrementally, based on the current input signal and the immediate error. This method, while straightforward, can result in slower convergence, particularly in environments with highly correlated or slowly varying input signals. In contrast, RLS employs a recursive estimation technique that utilizes all past and present data to update the filter coefficients. This holistic approach results in significantly faster convergence and improved tracking of non-stationary signals.

The Recursive Least Squares (RLS) algorithm is another powerful technique in adaptive filtering, offering advantages in convergence speed and tracking capabilities compared to LMS and NLMS. However, it comes with increased computational complexity. Similar to LMS and NLMS, RLS aims to minimize the mean squared error (MSE) between the desired signal and the filter output. However, RLS takes a fundamentally different approach for weight update. It achieves this with the following steps:

(1). Initialization: The filter starts with a set of initial weights and an estimate of the inverse correlation matrix of the input signal (often an identity matrix).
(2). Signal Processing: The filter takes an input signal and processes it using the current weights.
(3). Error Calculation: The difference between the desired signal (d(k)) and the filter output (y(k)) is calculated as the error signal (e(k)) at each time step (k).
(4). Gain Calculation: RLS calculates a gain vector based on the current error and the estimated inverse correlation matrix. This gain vector determines how much the weights are adjusted based on the new error information.
(5). Weight Update: The filter weights are updated using the error signal, the gain vector, and the previously estimated inverse correlation matrix. The update ensures that the weights minimize the mean squared error over all past and present data.
(6). Correlation Matrix Update: The estimated inverse correlation matrix is updated recursively using the previous estimate and the current input signal. This update captures the time-varying nature of the input signal statistics.
(7). Repeat: Steps 2-6 are repeated for each new input signal.

The following equations provide a deeper understanding of the calculations involved in the RLS algorithm. The equations show the specific mathematical operations that drive the weight and correlation matrix updates [35].

(1). Error Calculation: Similar to LMS and NLMS, the error signal e(k) is calculated at each time step (k) as the difference between the desired signal d(k) and the filter output y(k):

$$e(k) = d(k) - y(k)$$

(2). Gain Calculation: The RLS algorithm calculates a gain vector g(k) that determines the weight update based on the current error and the estimated inverse correlation matrix P(k-1) of the input signal:

$$g(k) = P(k-1) * x(k) / (1 + x(k).T * P(k-1) * x(k))$$

(3). Weight Update: The filter weights are updated using the gain vector g(k), the error signal e(k), and the previous weights w(k-1):

$$w(k) = w(k-1) + g(k) * e(k)$$

(4). Correlation Matrix Update: The estimated inverse correlation matrix P(k) is updated recursively using the previous estimate P(k-1), the gain vector g(k), and the current input signal x(k):

$$P(k) = (I - g(k) * x(k).T) * P(k-1)$$

The flowchart for LMS is given in Figure 3. It highlights the key steps of RLS, including the gain calculation, weight update, and the crucial update of the correlation matrix. These features differentiate RLS from LMS and NLMS, leading to its superior performance in convergence and tracking, but at the cost of higher computational complexity

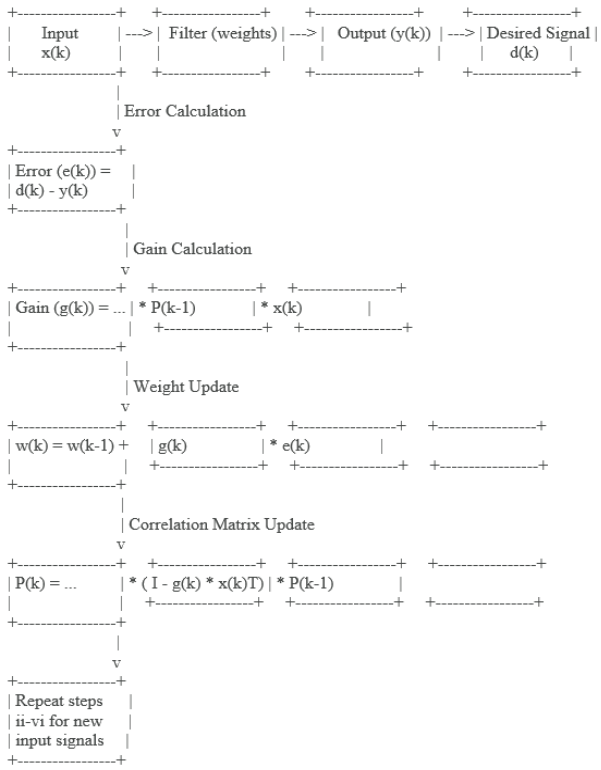

Figure 3. RLS flowchart

(1). Input: The flowchart starts with the input signal x(k).
(2). Filter: The input signal is processed by the filter with its current weights w(k-1).
(3). Output: The filter produces an output signal y(k).
(4). Error Calculation: The error signal e(k) is calculated as the difference between the desired signal d(k) and the filter output y(k).
(5). Gain Calculation: RLS calculates a gain vector g(k) based on the error e(k), the previously estimated inverse correlation matrix P(k-1), and the current input signal x(k).
(6). Weight Update: The filter weights are updated using the gain vector g(k), the error signal e(k), and the previous weights w(k-1).
(7). Correlation Matrix Update: The estimated inverse correlation matrix P(k) is updated recursively using the previous estimate P(k-1), the gain vector g(k), and the current input signal x(k). This update captures the characteristics of the time-varying input.
(8). Repeat: Steps ii-vii are repeated for each new input signal x(k). The recursive updates of weights and the correlation matrix enable RLS to achieve fast convergence and track non-stationary signals effectively.

2.3.1 Computational complexity

One of the key advantages of the RLS algorithm is its rapid convergence, facilitated by the comprehensive utilization of historical information and the adaptive adjustment of the Kalman gain. This rapid convergence makes RLS particularly well-suited for applications requiring real-time adaptation to changing signal conditions, such as adaptive equalization, system identification, and adaptive noise cancellation. Moreover, the RLS algorithm exhibits robustness to variations in signal statistics, making it suitable for handling non-stationary signals and time-varying environments.

However, the computational complexity of the RLS algorithm is higher compared to iterative methods like LMS and NLMS, primarily due to the computation of the inverse of the autocorrelation matrix. As a result, RLS may be less suitable for applications with stringent computational constraints or limited resources [34].

2.3.2 Advantages of RLS

(1). Fastest Convergence: RLS offers the fastest convergence among LMS, NLMS, and RLS, especially for stationary or slowly time-varying signals. It can achieve the minimum mean squared error in a theoretical sense.
(2). Excellent Tracking: RLS can effectively track non-stationary signals where the desired response or the signal statistics change over time, due to the recursive update of the correlation matrix.

2.3.3 Disadvantages of RLS

(1). High Computational Complexity: The RLS algorithm requires significantly more computations per iteration compared to LMS and NLMS due to matrix inversions and multiplications. This can be a limitation for real-time applications with limited processing power.
(2). Memory Intensive: RLS needs to store and update the inverse correlation matrix, which can be memory-intensive for long filters or high-dimensional input signals.
(3). Sensitive to Noise: RLS can be sensitive to noise in the input signal, as noise can corrupt the estimated correlation matrix and lead to performance degradation.

2.3.4 Applications of RLS

RLS is often used in applications where fast convergence and good tracking capabilities are crucial, despite the increased computational cost. Examples include:

(1). System identification and adaptive control
(2). Active noise cancellation with fast-changing noise sources
(3). Channel equalization for time-varying communication channels

The Recursive Least Squares (RLS) algorithm represents a sophisticated approach to adaptive filtering, offering rapid convergence, robustness, and adaptability to changing signal conditions. Its recursive formulation and utilization of historical information enable it to achieve superior performance in applications requiring real-time adaptation and handling of non-stationary signals. While the computational complexity and parameter selection present challenges, RLS remains a valuable tool in signal processing, driving advancements in telecommunications, audio processing, control systems, and beyond. There are variations of the RLS algorithm that aim to reduce the computational complexity or improve memory efficiency. These often involve approximations or simplifications in the update process. The

choice of the initial estimate for the inverse correlation matrix can impact the performance of RLS. Often, a diagonal matrix with high values on the diagonal is used as an initial guess.

2.4 Gradient Adaptive Lattice (GAL) algorithm

The Gradient Adaptive Lattice (GAL) algorithm is a sophisticated adaptive filtering technique that leverages the lattice structure to achieve efficient parameter estimation with improved numerical stability [36]. Unlike conventional algorithms such as the Least Mean Squares (LMS) or Recursive Least Squares (RLS), which operate directly on the filter coefficients, GAL operates on a lattice structure composed of reflection coefficients, offering advantages in terms of computational complexity and numerical precision [37].

At the core of the GAL algorithm lies the concept of lattice recursion, wherein the input signal is recursively processed through a lattice structure composed of adaptive weights known as reflection coefficients. These reflection coefficients capture the interactions between adjacent lattice stages, allowing for efficient parameter estimation and adaptation. The GAL algorithm operates by iteratively updating these reflection coefficients to minimize the error between the desired response and the actual output, similar to other adaptive filtering techniques.

The Gradient Adaptive Lattice (GAL) algorithm is a powerful technique for adaptive filtering, particularly well-suited for applications requiring noise cancellation or system identification [38]. It offers several advantages compared to the traditional transversal filter structure used in LMS and NLMS algorithms [39]. Unlike the transversal structure of LMS and NLMS, GAL employs a lattice structure. This structure allows for efficient implementation, reduced computational complexity, and inherent stability compared to transversal filters. The GAL algorithm aims to minimize the mean squared error (MSE) between the desired signal and the filter output, similar to other adaptive filtering techniques. The following are steps involved in GAL:

(1). Initialization: The filter starts with a set of initial reflection coefficients for each lattice stage. These coefficients control the filtering behavior.
(2). Signal Forward Propagation: The input signal is fed into the first stage of the lattice filter.
(3). Error Calculation at Each Stage: At each stage, the error signal is calculated as the difference between the forward prediction (based on previous stages' outputs) and the desired signal.
(4). Reflection Coefficient Update: Based on the error signal at each stage, the reflection coefficient for that stage is updated using a gradient descent approach. This minimizes the overall mean squared error.
(5). Backward Adaptation: The updated reflection coefficients are used to adapt the internal filter structure in a backward fashion.
(6). Output Derivation: The filter output is obtained by combining the forward propagating signal and the backward propagating error signal at the final stage.
(7). Repeat: Steps 2-6 are repeated for each new input signal.

The following equations provide a deeper understanding of the calculations within the GAL algorithm [39]. The lattice structure and the concept of forward prediction error differentiate GAL from transversal filters. GAL offers advantages in specific applications due to its inherent stability and computational efficiency.

(1). Forward Prediction Error (a priori error): At each stage p of the lattice, the forward prediction error $\varepsilon_p(k)$ represents the difference between the desired signal d(k) and the forward prediction based on the previous stages' outputs:

$$\varepsilon_p(k) = d(k) - \Sigma [a_{p,i} * \varepsilon_{(p-1)(k-i)}] \quad (i = 1 \text{ to } p)$$

where:

$a_{p,i}$ is the reflection coefficient at stage p for the ith past sample.

$\varepsilon_{(p-1)(k-i)}$ is the forward prediction error from the previous stage (p-1) at time delay k-i.

(2). Reflection Coefficient Update: The reflection coefficient $a_p(k)$ at stage p is updated using a gradient descent approach based on the forward prediction error $\varepsilon_p(k)$ and a step-size parameter μ:

$$a_p(k+1) = a_p(k) + \mu * \varepsilon_p(k) * u_p(k)$$

where:

μ is a small positive step-size parameter controlling the update speed.

$u_p(k)$ is a filtered version of the input signal at stage p.

(3). Backward Adaptation: The updated reflection coefficients are used to adjust the internal filter structure in a backward fashion, influencing future forward predictions.
(4). Output: The filter output y(k) is obtained by combining the forward propagating signal and the backward propagating error signal at the final stage.

These equations detail the calculations involved in the forward prediction error, reflection coefficient update, and the concept behind backward adaptation.

(1). Initialization: The filter starts with initial reflection coefficients for each stage.
(2). Signal Forward Propagation: The input signal is fed into the first stage.
(3). Error Calculation at Each Stage: The forward prediction error $\varepsilon_p(k)$ is calculated at each stage using the equation.
(4). Reflection Coefficient Update: The reflection coefficient $a_p(k)$ for each stage is updated using the equation based on $\varepsilon_p(k)$, μ, and $u_p(k)$.
(5). Backward Adaptation: The updated reflection coefficients are used to adapt the internal filter structure based on the concept explained.
(6). Output Derivation: The filter output y(k) is obtained by combining forward and backward signals at the final stage.
(7). Repeat: Steps ii-vi are repeated for each new input signal.

The flowchart for LMS is given in Figure 4. It highlights the key steps involved in filtering the input signal and adapting the filter based on the desired output.

Step 1: Input (Start of Each Iteration)

(1). x(k): This represents the current input signal at time step k.

Steps 2-4: Forward Processing and Error Calculation (Each Stage)

(1). $u_p(k)$: This denotes the input signal at stage p after

processing with previous reflection coefficients.

(2). v_(p-1)(k): This signifies the forward output from the previous stage (p-1) at time step k.

(3). a_i(k): This represents the reflection coefficient at stage i at time step k. (Used in multiple stages)

(4). ε_p(k): This signifies the forward prediction error calculated at stage p based on the difference between the desired signal d(k) and the forward prediction using previous stages' outputs and reflection coefficients.

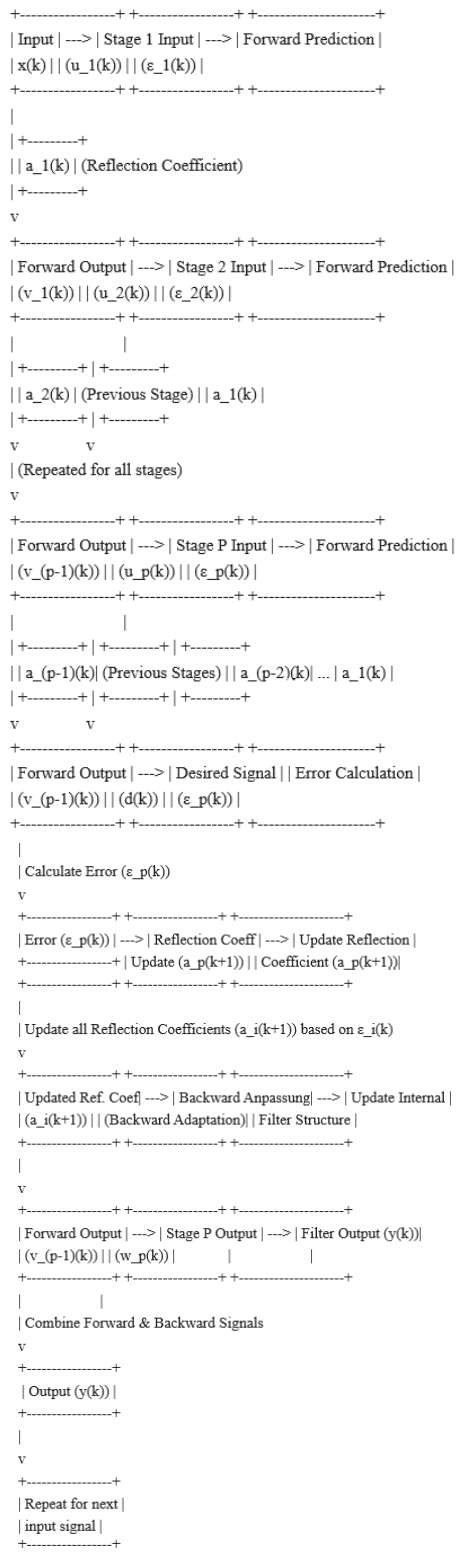

Figure 4. GAL flowchart

Step 5: Reflection Coefficient Update (Each Stage)

(1). a_p(k+1): This represents the updated reflection coefficient at stage p for the next iteration (k+1), calculated based on the error signal ε_p(k) and a step-size parameter μ (not explicitly shown).

Step 6 (Implicit): Backward Adaptation

(1). This step is not explicitly shown but is crucial. The updated reflection coefficients (a_i(k+1)) for all stages are used to adjust the internal filter structure in a backward fashion. This adaptation influences future forward predictions, impacting the filter's behavior.

Step 7: Output Derivation

(1). w_p(k): This signifies the output from stage p at time step k.

(2). y(k): This represents the final filter output y(k) obtained by combining the forward output from the final stage (v_(p-1)(k)) and the backward propagating error signal.

Step 8: Iteration

(1). The flowchart emphasizes that the entire process (Steps 1-7) repeats for each new input signal x(k).

In the GAL algorithm, the forward recursion computes the forward reflection coefficients based on the autocorrelation at each lattice stage, while the backward recursion updates the backward reflection coefficients based on the forward reflection coefficients and the lattice prediction error. These recursive updates allow the GAL algorithm to adaptively adjust the lattice structure to minimize the prediction error and achieve optimal filter performance.

One of the key advantages of the GAL algorithm is its numerical stability, which stems from the lattice structure and the recursive nature of parameter estimation. By operating on reflection coefficients rather than direct filter coefficients, GAL mitigates numerical precision issues that may arise in conventional algorithms, such as RLS. Additionally, the lattice structure offers computational advantages, as the number of computations required for parameter estimation scales linearly with the filter length, unlike direct methods that scale quadratically.

Furthermore, the GAL algorithm exhibits robust performance in non-stationary environments and can handle rapidly changing signal statistics effectively. Its ability to adaptively adjust the lattice structure allows it to track changes in signal characteristics and maintain optimal filter performance over time. Additionally, GAL offers advantages in terms of hardware implementation, making it suitable for real-time applications with limited computational resources.

However, despite its advantages, the GAL algorithm may suffer from sensitivity to initialization and parameter tuning, particularly in scenarios with highly correlated input signals or rapidly changing environments. Additionally, the computational complexity of the GAL algorithm may be higher compared to simpler adaptive filtering techniques, necessitating careful consideration of resource constraints in practical implementations.

2.4.1 Advantages of GAL

(1). Reduced Computational Complexity: Compared to transversal filters used in LMS and NLMS, the lattice structure of GAL offers lower computational complexity, making it suitable for real-time applications.

(2). Inherent Stability: The lattice structure naturally

avoids stability issues that can arise in transversal filters with large adaptation steps.

(3). Efficient Noise Cancellation: GAL is particularly effective in canceling narrowband noise due to its ability to create deep notches in the frequency response.

2.4.2 Disadvantages of GAL

(1). Higher Initial Set-up Cost: Setting up the initial reflection coefficients for the lattice filter can be more complex compared to initializing weights in transversal filters.

(2). May Not Be Optimal for All Applications: While effective for noise cancellation and system identification, GAL might not be the best choice for all adaptive filtering tasks, especially those requiring very fast convergence.

2.4.3 Applications of GAL

(1). Echo cancellation in telecommunications.

(2). Noise cancellation in headphones and audio systems.

(3). System identification and adaptive control.

(4). Signal prediction and channel equalization (particularly for narrowband interference).

2.4.4 Additional points

(1). There are variations of the GAL algorithm, such as the variable step-size GAL, which can improve convergence speed in certain scenarios.

(2). The GAL algorithm can be combined with other adaptive filtering techniques to leverage the benefits of both approaches.

2.4.5 Comparison to LMS and NLMS

(1). While LMS and NLMS offer simplicity and ease of implementation, GAL provides advantages in terms of computational complexity and inherent stability, making it a valuable alternative for specific applications.

(2). The choice between GAL, LMS, and NLMS depends on factors like computational constraints, desired convergence speed, and the nature of the filtering task (e.g., noise cancellation vs. general purpose filtering).

The Gradient Adaptive Lattice (GAL) algorithm represents a sophisticated approach to adaptive filtering, leveraging the lattice structure to achieve efficient parameter estimation with improved numerical stability. Its recursive nature, coupled with the adaptability of the lattice structure, enables robust performance in non-stationary environments and real-time applications. While challenges such as sensitivity to initialization and computational complexity exist, GAL remains a valuable tool in signal processing, driving advancements in telecommunications, audio processing, and other domains.

2.5 Fractional Tap-Length (FTL) adaptive filter

The Fractional Tap-Length (FTL) adaptive filter represents an innovative approach to adaptive filtering that offers a balance between computational efficiency and performance adaptability [40]. Unlike traditional adaptive filters with fixed tap-lengths, which rely on a predetermined number of taps to capture signal dynamics, FTL adaptively selects the tap-length based on the characteristics of the input signal. This dynamic adjustment enables FTL to efficiently adapt to varying signal conditions while minimizing computational complexity.

2.5.1 Motivation for using Fractional Tap-Lengths

Traditional fixed-length filters can face limitations when the optimal filter length is not known a priori or when the signal characteristics change over time. A fixed tap-length may either be too short to capture all the relevant signal features or too long, resulting in unnecessary computational overhead and potential overfitting. The motivation behind using fractional tap-lengths is to create a more flexible and efficient filtering mechanism that can dynamically adjust to the optimal filter length based on the input signal's characteristics.

Fractional tap-lengths allow the filter to fine-tune the number of active taps, providing a more granular and precise control over the filter's response. This flexibility improves the filter's performance in scenarios with varying signal complexities and noise levels, enabling better signal representation and noise reduction.

At the heart of the FTL algorithm lies the concept of fractional tap-lengths, wherein the filter dynamically adjusts the number of taps used for parameter estimation based on the signal characteristics. This dynamic adjustment is typically guided by a criterion such as the signal-to-noise ratio (SNR) or the level of signal complexity, allowing FTL to allocate resources more effectively and focus computational efforts on regions of the signal that are most relevant for estimation [40].

The Fractional Tap-Length (FTL) adaptive filter is a technique that addresses a limitation in traditional fixed-length transversal filters used in adaptive filtering algorithms like LMS and NLMS [41]. These algorithms require a pre-defined number of taps (weights) for the filter, which might not be optimal for all scenarios. The FTL filter overcomes this limitation by introducing the concept of fractional tap weights. The FTL filter utilizes a fixed number of physical taps but allows the weights associated with these taps to take on fractional values between 0 and 1, effectively increasing the filter's flexibility and adaptability. This fractional representation helps the filter achieve a response that might be difficult or impossible with only integer-valued weights. The following steps are required in FTL operation:

(1). Initialization: The filter starts with a set of initial weight values, which can be fractional.

(2). Signal Processing: The input signal is processed by the filter using the current fractional weights.

(3). Error Calculation: The difference between the desired signal and the filter output is calculated as the error signal.

(4). Weight Update: The fractional weights are updated based on an adaptive algorithm like LMS or NLMS, but the update step size is scaled by a factor related to the fractional weight itself. This scaling ensures stability and proper convergence.

(5). Repeat: Steps 2-4 are repeated for each new input signal.

2.5.2 Mathematical formulation

(1). Initialization: $w_i(0)$

where, w_i are the initial fractional weights for each tap i.

(2). Signal Processing: $y(k) = \sum^{L-1}_{i=0} wi(k) \cdot x(k-i)$

where, y(k) is the filter output at time step k, $w_i(k)$ are the fractional weights, and $x(k-i)$ are the input signal samples.

(3). Error Calculation: $e(k) = d(k) - y(k)$

where, e(k) is the error signal, and d(k) is the desired signal.

(4). Weight Update: $w_i(k+1) = w_i(k) + \mu \cdot e(k) \cdot x(k-i) \cdot \phi(w_i(k))$

where, μ is the step size, and $\phi(w_i(k))$ is a scaling function related to the fractional weight $w_i(k)$.

Figure 5 shows the flowchart of FTL. This flowchart, along with the explanation, highlights how FTL filters operate. By utilizing fractional weights and the weight scaling factor during updates, FTL filters achieve a more adaptable and precise filtering response compared to traditional fixed-length integer-weight filters.

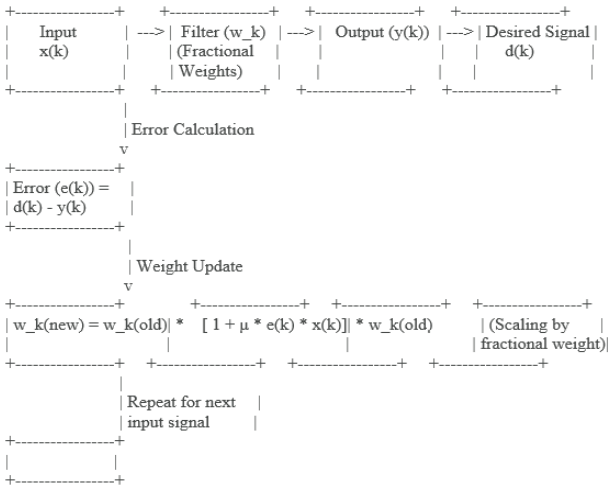

Figure 5. FTL flowchart

(1). Input: The process begins with the input signal `x(k)`.
(2). Filter: The input signal is processed by the filter using the current fractional weights `w_k`.
(3). Output: The filter produces an output signal `y(k)`.
(4). Error Calculation: The error signal `e(k)` is calculated as the difference between the desired signal `d(k)` and the filter output `y(k)`.
(5). Weight Update: The fractional weights `w_k` are updated based on an adaptive algorithm like LMS or NLMS. However, the update step size (`μ`) is scaled by the current fractional weight `w_k(old)` itself. This scaling factor ensures stability and proper convergence during the update process.

2.5.3 Advantages of FTL

(1). Improved Approximation Capability: By allowing fractional weights, the FTL filter can approximate a wider range of desired filter responses compared to fixed-length integer-weight filters.
(2). Reduced Computational Complexity: Compared to increasing the number of taps in a traditional filter, FTL can achieve similar performance with a lower number of physical taps, potentially reducing computational complexity.
(3). Flexibility: FTL offers more flexibility in filter design, allowing for a finer adjustment of the filter's frequency response.

2.5.4 Disadvantages of FTL

(1). Increased Implementation Complexity: While potentially reducing computational complexity compared to many taps, implementing the fractional weight scaling during updates can add slight complexity compared to a standard LMS or NLMS implementation.
(2). Potential for Numerical Issues: Depending on the specific implementation and data types used, there might be a small risk of numerical errors due to the fractional weight representation.

2.5.5 Applications of FTL

(1). System identification and adaptive control
(2). Channel equalization, particularly for channels with rapid variations
(3). Noise cancellation, especially for applications requiring precise filtering characteristics

2.5.6 Comparison to Traditional Fixed-Length Filters

(1). FTL filters offer a balance between the simplicity of fixed-length filters and the potential performance gains of using a large number of taps.
(2). The choice between FTL and traditional filters depends on factors like the desired filter response complexity, computational constraints, and the specific application requirements.

The Fractional Tap-Length (FTL) adaptive filter represents a versatile approach to adaptive filtering that offers a balance between computational efficiency and performance adaptability. Its ability to dynamically adjust the tap-length based on signal characteristics enables efficient resource allocation and effective adaptation to changing signal conditions. While challenges such as parameter selection and convergence issues exist, FTL remains a valuable tool in signal processing, driving advancements in telecommunications, audio processing, and other domains. There are variations of the FTL algorithm, such as the convex combination of adaptive filters for variable width and fractional tap-length LMS, which can further enhance the flexibility and performance. FTL can be combined with other adaptive filtering techniques like variable step-size LMS to improve convergence speed.

3. STRUCTURE OF ADAPTIVE FILTERS

The structure of adaptive filters encompasses the architecture and components that enable these filters to dynamically adjust their parameters in response to changing input signals and system conditions. While the specific structure may vary depending on the algorithm and application, adaptive filters typically share common elements designed to facilitate parameter estimation, error minimization, and adaptation. The general structure of an adaptive filter consists of an input signal, a filter, an adaptation algorithm, a reference signal, and an error calculation block. The input signal is processed through the filter, and the output is compared with the desired response or a reference signal to compute the error. The adaptation algorithm then updates the filter coefficients based on this error to minimize it over time.

(a). Input Signal

At the core of every adaptive filter is the input signal, which serves as the primary source of information for parameter estimation and adaptation. The input signal may originate from various sources, including sensors, communication channels, or recorded data streams. It forms the basis for the filtering operation, with the adaptive filter tasked with processing and modifying the input signal to achieve specific objectives such as noise reduction, signal enhancement, or system

identification.

(b). Filter Structure

The filter structure defines the arrangement and connectivity of filter coefficients, which are adjusted adaptively to achieve desired filtering characteristics. Depending on the application and algorithm, adaptive filters may employ different filter structures such as finite impulse response (FIR) filters, infinite impulse response (IIR) filters, lattice filters, or frequency-domain filters. Each filter structure offers unique advantages and trade-offs in terms of computational complexity, stability, and performance.

(c). Adaptation Algorithm

The adaptation algorithm governs the update mechanism for adjusting filter coefficients based on input signals and error signals. Common adaptation algorithms include the Least Mean Squares (LMS) algorithm, Recursive Least Squares (RLS) algorithm, Normalized Least Mean Squares (NLMS) algorithm, and variants thereof. These algorithms employ iterative or recursive procedures to minimize the error between the desired response and the actual output, driving the filter towards convergence and optimal performance.

(d). Error Calculation

Error calculation is a critical component of adaptive filters, providing feedback on the discrepancy between the desired response and the actual output. The error signal is typically computed by comparing the output of the adaptive filter with a reference signal or desired response. This error signal serves as the driving force for parameter adaptation, guiding the filter towards minimizing the difference between the desired and actual outputs.

(e). Adaptation Control Parameters

Adaptive filters often incorporate control parameters that govern the adaptation process, including the step size (or learning rate), forgetting factor, regularization parameter, and convergence criteria. These parameters influence the rate of adaptation, stability, and convergence behavior of the adaptive filter. Proper selection and tuning of these parameters are essential for achieving optimal filter performance and stability.

(f). Output Signal

The output signal of the adaptive filter represents the processed version of the input signal, modified according to the adaptive filter's parameters and characteristics. The output signal reflects the filter's ability to mitigate noise, enhance signal features, or perform system identification, depending on the application and objectives.

Overall, the structure of adaptive filters embodies a dynamic framework for parameter estimation and adaptation, enabling them to effectively respond to changing signal conditions and achieve desired filtering objectives. By leveraging input signals, adaptation algorithms, error feedback, and control parameters, adaptive filters can adaptively adjust their parameters to optimize performance across a wide range of applications, including telecommunications, audio processing, biomedical signal processing, and control systems.

4. CHARACTERISTICS OF ADAPTIVE FILTER METHODS

As we delve deeper into the realm of adaptive filters, it becomes essential to examine the distinct characteristics that define each method. These characteristics shape the performance, applicability, and adaptability of adaptive filters across various signal processing tasks. Understanding these attributes provides valuable insights into the strengths and limitations of each method, enabling informed decision-making in selecting the most suitable approach for specific applications. Let us now explore the key characteristics of five prominent adaptive filter methods, shedding light on their respective advantages and disadvantages as shown in Table 1.

(1). Convergence Rate: Refers to how quickly the adaptive filter reaches a steady-state solution or converges to a desired response.

(2). Stability: Ensures that the filter remains stable during operation, avoiding issues such as oscillations or divergence.

(3). Computational Complexity: Indicates the computational resources required to implement the adaptive filter algorithm, including memory and processing power.

(4). Robustness: Describes the ability of the adaptive filter to maintain performance in the presence of variations in input signals or environmental conditions.

(5). Adaptability: Reflects how well the adaptive filter adjusts its parameters to changes in the input signal or system characteristics.

Table 1. Advantages and disadvantages of each method

Methods	Advantages	Disadvantages
LMS Algorithm	Simple implementation, low computational complexity.	Slow convergence, sensitivity to step size parameters.
NLMS Algorithm	Faster convergence, robustness to variations in signal power.	Slow convergence in non-stationary environments.
RLS Algorithm	Fast convergence, high performance in non-stationary environments.	High computational complexity, memory requirements.
GAL Algorithm	Numerical stability, low computational complexity.	Sensitivity to initialization, numerical precision issues.
FTL Adaptive Filter	Dynamic tap-length adjustment, reduced computational complexity.	Requires careful design considerations, may sacrifice performance for complexity reduction.

5. FUTURE DIRECTION OF ADAPTIVE FILTERS

The future of adaptive filters lies in addressing emerging challenges such as handling massive data streams, adapting to dynamic environments in real-time, and integrating with machine learning techniques for enhanced adaptability and intelligence. Research efforts will likely focus on developing more efficient adaptation algorithms, exploring novel architectures, and leveraging advancements in hardware technology for faster and more energy-efficient

implementations. Additionally, adaptive filters will continue to find applications in emerging fields such as Internet of Things (IoT), autonomous systems, and cognitive radio networks, driving innovation and evolution in adaptive filtering techniques.

5.1 Potential areas for further exploration

1. Machine Learning Integration: One of the most promising directions for adaptive filters is the integration of machine learning techniques. By leveraging vast datasets, machine learning algorithms can enhance the adaptability and intelligence of adaptive filters. Techniques such as deep learning, reinforcement learning, and neural networks can be used to develop adaptive filters that can learn complex patterns and improve performance in dynamic environments.
2. Real-Time Adaptation: Real-time applications require adaptive filters to process and respond to signals with minimal delay. Future research can focus on developing ultra-fast adaptive algorithms that can keep up with high-speed data streams. This involves optimizing the computational efficiency of existing algorithms and exploring new architectures that can handle real-time adaptation without compromising performance.
3. Energy-Efficient Implementations: As adaptive filters are deployed in portable and battery-powered devices, energy efficiency becomes a critical concern. Research can explore ways to reduce the power consumption of adaptive filters by optimizing their algorithms and leveraging energy-efficient hardware architectures such as FPGA and ASIC implementations.
4. Robustness to Non-Stationary Environments: Adaptive filters need to maintain high performance in non-stationary environments where signal characteristics change over time. Future research can focus on developing algorithms that can quickly adapt to these changes and maintain robust performance. This includes exploring adaptive filters with variable step sizes, dynamic tap-length adjustments, and other mechanisms that can respond to changing conditions.
5. Hybrid Adaptive Filtering Techniques: Combining multiple adaptive filtering techniques can leverage the strengths of each method and mitigate their individual limitations. Research can explore hybrid approaches that integrate different algorithms, such as combining LMS with RLS or FTL with NLMS, to achieve superior performance in various applications.
6. Applications in Emerging Technologies: Adaptive filters will play a crucial role in emerging technologies such as the Internet of Things (IoT), autonomous systems, and cognitive radio networks. These applications require adaptive filters that can handle large-scale, heterogeneous, and dynamic data. Research can focus on developing adaptive filters tailored for these applications, addressing their specific challenges and requirements.
7. Enhanced Noise Cancellation: As noise cancellation technology evolves, there is a growing demand for adaptive filters that can provide superior noise suppression in various environments. Future research can explore advanced algorithms and architectures for adaptive noise cancellation, aiming to improve the accuracy and efficiency of these systems.
8. Biomedical Signal Processing: Adaptive filters are essential in biomedical applications, where they help in extracting meaningful physiological information from noisy bio-signals. Future research can focus on developing adaptive filters that can handle the unique challenges of biomedical signal processing, such as non-stationary noise and artifact removal, and enhance the accuracy of medical diagnostics and treatments.

5.2 Future trends

1. Data-Driven Adaptive Filtering: The trend towards data-driven approaches in adaptive filtering is expected to continue, with a focus on algorithms that can learn from large datasets and improve their performance over time. This includes the use of big data analytics, machine learning, and artificial intelligence to enhance the adaptability and intelligence of adaptive filters.
2. Hardware Acceleration: The use of hardware acceleration, such as GPUs and specialized hardware, is becoming increasingly important for implementing adaptive filters in real-time applications. Research will likely explore new hardware architectures and techniques to accelerate adaptive filtering algorithms and improve their performance.
3. Self-Learning Adaptive Filters: The development of self-learning adaptive filters that can autonomously adjust their parameters and improve their performance without human intervention is a promising area of research. These filters can use techniques such as reinforcement learning and online learning to continuously adapt and optimize their performance.
4. Multi-Sensor Adaptive Filtering: As the number of sensors and the complexity of sensor networks increase, there is a need for adaptive filters that can process and integrate data from multiple sensors. Future research can focus on developing multi-sensor adaptive filters that can handle heterogeneous data and provide accurate and reliable signal processing.

In summary, the future of adaptive filters is bright, with numerous opportunities for innovation and improvement. By addressing the challenges of real-time adaptation, energy efficiency, robustness, and integration with machine learning, adaptive filters can continue to evolve and play a crucial role in various applications. As technology advances, adaptive filters will become even more versatile and intelligent, driving progress in telecommunications, audio processing, biomedical signal processing, and beyond.

6. CONCLUSION

In conclusion, the realm of adaptive filters stands as a dynamic and versatile domain within signal processing, offering sophisticated methodologies for parameter estimation, error minimization, and adaptation in response to changing signal conditions. Throughout this exploration, we have examined several prominent adaptive filtering techniques,

each with its unique characteristics, advantages, and considerations.

The Least Mean Squares (LMS) algorithm, known for its simplicity and computational efficiency, serves as a foundational methodology in adaptive filtering. While offering ease of implementation, LMS may encounter challenges such as slow convergence and sensitivity to step size parameters. The Normalized Least Mean Squares (NLMS) algorithm addresses some of these limitations by normalizing the adaptation step size based on the input signal power, resulting in improved convergence and robustness to signal variations.

The Recursive Least Squares (RLS) algorithm represents a sophisticated approach to adaptive filtering, leveraging recursive estimation and the entire history of observations to achieve rapid convergence and robust performance in non-stationary environments. Despite its computational complexity, RLS offers unparalleled adaptability and numerical stability, making it suitable for real-time applications requiring dynamic parameter adjustment.

The Gradient Adaptive Lattice (GAL) algorithm introduces a novel approach to adaptive filtering, utilizing the lattice structure to achieve efficient parameter estimation with improved numerical stability. By operating on reflection coefficients, GAL mitigates numerical precision issues and offers robust performance in non-stationary environments. Similarly, the Fractional Tap-Length (FTL) adaptive filter dynamically adjusts the tap-length based on signal characteristics, offering a balance between computational efficiency and adaptability.

The structure of adaptive filters embodies a dynamic framework for parameter estimation and adaptation, encompassing input signals, filter structures, adaptation algorithms, error feedback, and control parameters. Through iterative or recursive procedures, adaptive filters adaptively adjust their parameters to minimize errors and optimize performance across diverse applications, including telecommunications, audio processing, biomedical signal processing, and control systems.

In essence, adaptive filters play a crucial role in modern signal processing, offering intelligent solutions for mitigating noise, enhancing signal features, and performing system identification in real-world environments. As technology continues to evolve, adaptive filters will continue to drive innovation and advancements, enabling more efficient and effective processing of signals in a wide range of applications. By leveraging the principles of adaptability, resilience, and optimization, adaptive filters pave the way for enhanced performance and intelligence in signal processing systems of the future.

ACKNOWLEDGMENT

The authors wish to acknowledge Covenant University for her financial support.

REFERENCES

[1] Yazdanpanah, H., Diniz, P.S., Lima, M.V. (2020). Feature adaptive filtering: Exploiting hidden sparsity. IEEE Transactions on Circuits and Systems I: Regular Papers, 67(7): 2358-2371. https://doi.org/10.1109/TCSI.2020.2976882

[2] Kumar, K., Pandey, R., Karthik, M.L.S., Bhattacharjee, S.S., George, N.V. (2021). Robust and sparsity-aware adaptive filters: A review. Signal Processing, 189: 108276. https://doi.org/10.1016/j.sigpro.2021.108276

[3] Zhang, X.D. (2022). Modern Signal Processing. Walter de Gruyter GmbH & Co KG.

[4] Cheng, P., Chen, Z., Ding, M., Li, Y., Vucetic, B., Niyato, D. (2020). Spectrum intelligent radio: Technology, development, and future trends. IEEE Communications Magazine, 58(1): 12-18. https://doi.org/10.1109/MCOM.001.1900200

[5] Amali, S.M.J., Baskar, S. (2013). Fuzzy logic-based diversity-controlled self-adaptive differential evolution. Engineering Optimization, 45(8): 899-915. https://doi.org/10.1080/0305215X.2012.713356

[6] Zhao, J., Zhao, W., Deng, B., et al. (2023). Autonomous driving system: A comprehensive survey. Expert Systems with Applications, 122836. https://doi.org/10.1016/j.eswa.2023.122836

[7] Henderson, J.J. (2021). Mande music in the black Atlantic: Migration and the affordances of world music record production (Doctoral dissertation, Duke University).

[8] Puglisi, V.F. (2023). Audio Analysis via Deep Learning for Forensics and Investigation Purposes. Università Degli Studi di Catania.

[9] Rapp, M.C.L. (2023). Retroreflection and waterfalls: Speaking and singing in extreme acoustic environments (Doctoral dissertation).

[10] Nayak, N.R., Bhuvaneshwari, S. (2017). A survey on task scheduling algorithm and quality of service for resource allocation in cloud environment. Indian Journal of Engineering, 14(35): 63-70.

[11] Okokpujie, K., Okokpujie, I.P., Subair, R.E., Simonyan, E.O., Akingunsoye, A.V. (2023). Designing an adaptive age-invariant face recognition system for enhanced security in smart urban environments. Ingénierie des Systèmes d'Information, 28(4): 815-822. https://doi.org/10.18280/isi.280402

[12] Kumar, M., Mishra, S.K. (2020). A comprehensive review on nature inspired neural network based adaptive filter for eliminating noise in medical images. Current Medical Imaging, 16(4): 278-287. https://doi.org/10.2174/1573405614666180801113345

[13] Mohammed Kaleem, A., Kokate, R.D. (2021). A survey on FECG extraction using neural network and adaptive filter. Soft Computing, 25: 4379-4392. https://doi.org/10.1007/s00500-020-05447-w

[14] Alam, A. (2022). Employing adaptive learning and intelligent tutoring robots for virtual classrooms and smart campuses: Reforming education in the age of artificial intelligence. In Advanced Computing and Intelligent Technologies: Proceedings of ICACIT 2022, Singapore: Springer Nature Singapore, pp. 395-406. https://doi.org/10.1007/978-981-19-2980-9_32

[15] Awelewa, A., Omiloli, K., Samuel, I., Olajube, A., Popoola, O. (2023). Robust hybrid estimator for the state of charge of a lithium-ion battery. Frontiers in Energy Research, 10: 1069364. https://doi.org/10.3389/fenrg.2022.1069364

[16] Omiloli, K., Awelewa, A., Samuel, I., Obiazi, O., Katende, J. (2023). State of charge estimation based on a modified extended Kalman filter. International Journal of

Electrical & Computer Engineering, 13(5): 5054-5065. https://doi.org/10.11591/ijece.v13i5.pp5054-5065
[17] Kumar, R., Sharma, N.V.K., Chaurasiya, V.K. (2024). Adaptive traffic light control using deep reinforcement learning technique. Multimedia Tools and Applications, 83(5): 13851-13872. https://doi.org/10.1007/s11042-023-16112-3
[18] Zhang, Y., Carballo, A., Yang, H., Takeda, K. (2023). Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. ISPRS Journal of Photogrammetry and Remote Sensing, 196: 146-177. https://doi.org/10.1016/j.isprsjprs.2022.12.021
[19] Kadhim, N.O., Khudhair, K.T., Najjar, F.H., Al-Jawahry, H.M. (2021). Digital filters windowing for data transmission enhancement in communication channel. Indonesian Journal of Electrical Engineering and Computer Science, 24(3): 1454-1468. https://doi.org/10.11591/ijeecs.v24.i3.pp1454-1468
[20] Rangayyan, R.M., Krishnan, S. (2024). Biomedical Signal Analysis. John Wiley & Sons.
[21] Górriz, J.M., Ramírez, J., Ortiz, A., et al. (2020). Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications. Neurocomputing, 410: 237-270. https://doi.org/10.1016/j.neucom.2020.05.078
[22] Ma, L., Huang, H., Zhao, P., Su, T. (2020). Acoustic echo cancellation by combining adaptive digital filter and recurrent neural network. arXiv preprint arXiv:2005.09237. https://doi.org/10.48550/arXiv.2005.09237
[23] Sankar, S., Kar, A., Burra, S., Swamy, M.N.S., Mladenovic, V. (2020). Nonlinear acoustic echo cancellation with kernelized adaptive filters. Applied Acoustics, 166: 107329. https://doi.org/10.1016/j.apacoust.2020.107329
[24] Diniz, P.S., Diniz, P.S. (2020). The least-mean-square (LMS) algorithm. Adaptive Filtering: Algorithms and Practical Implementation, 61-102. https://doi.org/10.1007/978-3-030-29057-3_3
[25] Zerguine, A., Ahmad, J., Moinuddin, M., Al-Saggaf, U. M., Zoubir, A.M. (2022). An efficient normalized LMS algorithm. Nonlinear Dynamics, 110(4): 3561-3579. https://doi.org/10.1007/s11071-022-07773-0
[26] Dogariu, L.M., Paleologu, C., Benesty, J., Oprea, C., Ciochină, S. (2020). LMS algorithms for multilinear forms. In 2020 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, pp. 1-4. https://doi.org/10.1109/ISETC50328.2020.9301133
[27] Ling, Q., Ikbal, M. A., Kumar, P. (2021). Optimized LMS algorithm for system identification and noise cancellation. Journal of Intelligent Systems, 30(1): 487-498. https://doi.org/10.1515/jisys-2020-0081
[28] Bismor, D. (2023). Leaky partial update LMS algorithms in application to structural active noise control. Sensors, 23(3): 1169. https://doi.org/10.3390/s23031169
[29] Khan, A.A., Shah, S.M., Raja, M.A.Z., Chaudhary, N.I., He, Y., Machado, J.T. (2021). Fractional LMS and NLMS algorithms for line echo cancellation. Arabian Journal for Science and Engineering, 1-14. https://doi.org/10.1007/s13369-020-05264-1
[30] Becker, A.C., Kuhn, E.V., Matsuo, M.V., Benesty, J. (2024). On the NP-VSS-NLMS algorithm: Model, design guidelines, and numerical results. Circuits, Systems, and Signal Processing, 43(4): 2409-2427. https://doi.org/10.1007/s00034-023-02565-2
[31] Lee, H.S., Jin, C., Shin, C., Kim, S.E. (2023). Sparse diffusion least mean-square algorithm with hard thresholding over networks. Mathematics, 11(22): 4638. https://doi.org/10.3390/math11224638
[32] Lee, M., Park, T., Park, P. (2022). Variable step-size ℓ_0 -Norm constraint NLMS algorithms based on novel mean square deviation analyses. IEEE Transactions on Signal Processing, 70: 5926-5939. https://doi.org/10.1109/TSP.2022.3231187
[33] La Rosa, A.B., Pereira, P.T., Ücker, P., Paim, G., da Costa, E.A., Bampi, S., Almeida, S. (2021). Exploring NLMS-based adaptive filter hardware architectures for eliminating power line interference in EEG signals. Circuits, Systems, and Signal Processing, 40: 3305-3337. https://doi.org/10.1007/s00034-020-01620-6
[34] Eweda, E., Bershad, N.J., Bermudez, J.C. (2020). Stochastic analysis of the recursive least squares algorithm for cyclostationary colored inputs. IEEE Transactions on Signal Processing, 68: 676-686. https://doi.org/10.1109/TSP.2020.2964225
[35] Qin, Z., Tao, J., Xia, Y. (2020). A proportionate recursive least squares algorithm and its performance analysis. IEEE Transactions on Circuits and Systems II: Express Briefs, 68(1): 506-510. https://doi.org/10.1109/TCSII.2020.3004466
[36] Batabyal, T., Weller, D., Kapur, J., Acton, S.T. (2022). Efficient learning of transform-domain LMS filter using graph Laplacian. IEEE Transactions on Neural Networks and Learning Systems, 34(10): 7608-7620. https://doi.org/10.1109/TNNLS.2022.3144637
[37] Wung, J., Jukić, A., Malik, S., Souden, M., Pichevar, R., Atkins, J., Naik, D. Acero, A. (2020). Robust multichannel linear prediction for online speech dereverberation using weighted householder least squares lattice adaptive filter. IEEE Transactions on Signal Processing, 68: 3559-3574. https://doi.org/10.1109/TSP.2020.2997201
[38] Lu, L., Yin, K. L., de Lamare, R. C., Zheng, Z., Yu, Y., Yang, X., Chen, B. (2021). A survey on active noise control techniques--Part I: Linear systems. arXiv preprint arXiv:2110.00531. https://doi.org/10.48550/arXiv.2110.00531
[39] Sowmya, K.B., Chandana, G., Daigond, A.M. (2021). Adaptive filters and neural net. In Machine Learning in Signal Processing, Chapman and Hall/CRC, pp. 159-184.
[40] Azarnia, G., Sharifi, A.A. (2021). Steady-state analysis of distributed incremental variable fractional tap-length LMS adaptive networks. Wireless Networks, 27(7): 4603-4614. https://doi.org/10.1007/s11276-021-02754-4
[41] Pauline, S.H., Samiappan, D., Kumar, R., Anand, A., Kar, A. (2020). Variable tap-length non-parametric variable step-size NLMS adaptive filtering algorithm for acoustic echo cancellation. Applied Acoustics, 159: 107074. https://doi.org/10.1016/j.apacoust.2019.107074

NOMENCLATURE

IoT	Internet of Things
NLMS	Normalized Least Mean Squares
RLS	Recursive Least Squares
GAL	Gradient Adaptive Lattice
FTL	Fractional Tap-Length