WEP
Warwick
Evans
Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 1, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n1.14
Journal homepage: https://wepub.org/index.php/IJCSIT/index

# Research on Filtering Algorithms for LMS-based Pressure Testing Systems

Yuanhao Chang [1], Kun Wang [2,*]

[1] School of Mechanical Engineering, Tianjin University of Technology and Education, Tianjin 300222, China
[2] Tianjin Institute of Aerospace Mechanical and Electrical Equipment, Tianjin 300458, China

## ABSTRACT

Pressure testing systems are widely used in industrial, aerospace, and other fields, where environmental noise is a key factor affecting the accuracy of testing systems. To address the problems of noise interference in pressure testing systems and the drawbacks of the classical noise filtering algorithm LMS, such as slow convergence speed and insufficient time-varying signal tracking capability, this paper proposes a denoising method based on an improved adaptive filtering algorithm. Specifically, the Root Mean Square Propagation adaptive learning rate adjustment mechanism is introduced into the fixed step-size module of the LMS algorithm to optimize the weight update process of the algorithm. By leveraging the ability of RMSProp to adaptively assign learning rates to different parameters during the iteration process, the problem of convergence stagnation caused by premature learning rate attenuation is effectively avoided. Meanwhile, an attenuation coefficient is incorporated into the improved algorithm to suppress oscillations during gradient updates, thereby significantly enhancing the stability and robustness in time-varying noise environments. Experimental results show that the improved algorithm increases the convergence speed by 69.5% and improves the signal-to-noise ratio by 17.7%, which enhances the signal processing accuracy and anti-interference capability, and satisfies the denoising requirements of ground tests for pressure testing systems.

## KEYWORDS

Adaptive Algorithm; LMS; RMSProp; Convergence Speed

## 1. INTRODUCTION

In numerous technical fields such as modern industry and aerospace, force measuring systems play an indispensable core role. With the accelerated global industrialization and intelligentization process, traditional manual force measurement modes have been difficult to meet the requirements of high precision, high efficiency and high reliability in modern scenarios. Manual force measurement is plagued by problems including large data recording errors, high labor costs and susceptibility to human interference. Against this backdrop, force measuring systems have emerged as an integrated solution that incorporates sensor technology, data processing technology and automatic control technology. By enabling automatic collection, accurate measurement, real-time transmission and intelligent analysis of weight data, these systems have become an indispensable key infrastructure in the upgrading of modern industries, and are widely applied in sectors like iron and steel metallurgy, food processing and aerospace. At present, although force measuring systems have achieved remarkable development, many problems still persist in complex application scenarios. Among these, dynamic force measurement accuracy control remains a great challenge; under conditions such as high-speed movement of goods, vibration interference and uneven load distribution, how to balance measurement speed and accuracy continues to be a core issue.

The traditional PID dynamic weighing compensation method can achieve an accuracy of ±0.1% under low-velocity conditions through parameter adjustment. However, its fixed parameters make it difficult to adapt to complex working conditions such as acceleration/deceleration and sudden load changes, leading to compensation inaccuracies [1]. Yan W et al. [2] adopted the EKF algorithm to suppress sensor noise via state equation modeling, reducing the static weighing error by 0.08%. Nevertheless, this method relies on an accurate system model; model mismatch in complex environments will result in accuracy degradation, and it also shows poor adaptability to non-Gaussian noise. The multi-factor compensation algorithm based on the BP neural network proposed by A K R [3] integrates environmental parameters, which reduces the environmental interference error by 1.2%. Yet, it requires a large amount of labeled data for training, exhibits limited generalization capability under extreme working conditions, and features slow parameter updating, making it difficult to meet the requirements of real-time weighing. Zhang Haochen et al. [4] applied an adaptive FIR filter based on the LMS algorithm to dynamic vehicle weighing. This method can adjust parameters in real time to track signal changes, but there is still room for optimization. Liu X [5] employed a wavelet threshold denoising method to address vibration interference, cutting down the vibration-induced error by 0.5%. However, the threshold selection depends on empirical values, leading to poor adaptability to multi-frequency vibrations, and the coupling effect between vibration and load is not taken into consideration. In summary, the existing methods still have deficiencies in terms of dynamic adaptability, model robustness and complex working condition handling capability.

To address the issue of complex noise interference encountered by pressure testing systems in practical applications, this paper proposes an improved denoising method based on the adaptive filtering algorithm and verifies the superiority of the improved algorithm. Aiming at the problems of slow convergence speed and insufficient tracking capability for time-varying signals in non-stationary noise environments, the RMSProp adaptive learning rate adjustment mechanism is introduced to optimize the weight updating process of the algorithm. By assigning adaptive learning rates to different parameters during the iteration process, the problem of convergence stagnation caused by premature learning rate attenuation is effectively avoided. By suppressing oscillations in the gradient updating process, the stability and robustness of the algorithm in time-varying noise environments are significantly improved.

## 2. LMS ALGORITHM MODEL

The adaptive filtering theory, proposed by Widrow B et al., is an optimal filtering method developed on the basis of linear filtering techniques such as Wiener filtering and Kalman filtering [6]. Owing to its superior adaptive performance and enhanced filtering capability, it has been widely applied in numerous fields including communications, system identification, noise cancellation, adaptive line enhancement, adaptive channel equalization, speech linear prediction, and adaptive antenna arrays.

The core of an adaptive filter lies in its adaptive filtering algorithm. From the perspective of real-time control, the algorithm is expected to be simple with low computational complexity. Since the mean square error $\xi(n)$ is the filter weight $W(n)$ is a quadratic function, the method of steepest gradient descent can thus be adopted for recursive solution. According to the principle of the steepest gradient descent method, the iterative update of weight coefficients can be expressed as follows:

$$W(n+1) = W(n) - \frac{\mu}{2}\nabla\xi(n) \tag{1}$$

In the formula, μ denotes the convergence factor, also referred to as the convergence coefficient or step size, which exerts a crucial influence on the convergence speed of the iterative process. In general, a larger convergence step size leads to a relatively faster convergence speed [7]. $\nabla\xi(n)$ represents the gradient of the mean square error function with respect to the weight coefficient vector $W(n)$ the gradient, The negative sign indicates that the weight update proceeds along the direction of the gradient. Substituting Equation $RW^* = P$ into Equation (1), we obtain:

$$W(n+1) = W(n) + \mu\left[P - RW(n)\right] \tag{2}$$

Based on the optimization theory of steepest gradient descent, $W(n)$ will eventually converge to $W^*$, Meanwhile, $\xi(n)$ will attain its minimum value. However, in numerous practical applications, The statistical characteristics of signals d(n) and X(n) are unknown; therefore, iterative calculation cannot be directly performed using Equation (2). The square of the instantaneous error $e^2(n)$ is generally used to estimate the mean square $\xi(n)$, namely:

$$\hat{\xi}(n) = e^2(n) \tag{3}$$

The gradient of the corresponding mean square error estimate is expressed as follows:

$$\nabla\hat{\xi}(n) = 2\left[\nabla e(n)\right]e(n) \tag{4}$$

From the output error signal $e(n) = d(n) - y(n) = d(n) - W^T(n)X(n)$ of the adaptive filter, we obtain:

$$\nabla e(n) = -X(n) \tag{5}$$

Substituting into Equation (3), we obtain:

$$\nabla\hat{\xi}(n) = -2X(n)e(n) \tag{6}$$

Then Equation (1) is transformed into:

$$W(n+1) = W(n) + \mu X(n)e(n) \tag{7}$$

This is the well-known LMS algorithm. This method is relatively simple and effective for solving $W^*$, as it does not require prior calculation of the correlation matrix or matrix inversion operations.

The LMS algorithm is an adaptive filtering algorithm suitable for scenarios where prior statistical characteristics are unknown. Its core function lies in iteratively solving the weight coefficients of the filter to achieve effective filtering of mixed noise. The system operation principle of this algorithm is illustrated in Figure 1. where $u(n)$ denotes the input signal of the filter, and $y(n)$ denotes the output signal of the filter. The error signal $e(n)$ is obtained from the difference between the desired signal $d(n)$ and the output signal $y(n)$. Based on $e(n)$, the adaptive filtering algorithm is able to update the weight coefficients of the filter, and ultimately make the output signal $y(n)$ approximate the desired signal $d(n)$ as closely as possible [8].

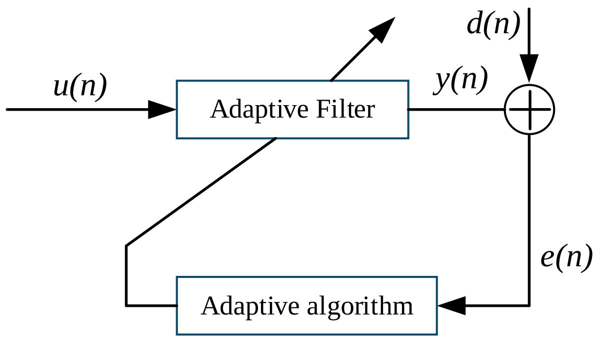


**Figure 1.** Schematic Diagram of the Adaptive Filter

For the adaptive filter illustrated in Figure 1, when the LMS adaptive filtering algorithm is applied to it, the calculation steps are as follows:

(1) Select the parameters and initial conditions, including the filter order L, the convergence coefficient μ, and the initial weight vector W(0).

(2) Calculate the output of the adaptive filter:

$$y(n)=\sum_{i=0}^{L-1} w_i(n)x(n-i) \tag{8}$$

(3) Calculate the error signal:

$$e(n)=d(n)-y(n) \tag{9}$$

(4) Update the weight $W(n)$ of the filter:

$$w_i(n+1)=w_i(n)+\mu x(n-l)e(n) \tag{10}$$

## 3. IMPROVEMENT AND OPTIMIZATION OF THE LMS ALGORITHM

### 3.1. Algorithm Improvement

The adaptive gradient algorithm (AdaGrad) is a type of algorithm capable of adaptively adjusting the model's learning rate parameters. It can yield favorable results in deep learning models. The optimizer of this algorithm first sets a global learning rate, then takes the ratio of the global learning rate to the square root of the accumulated historical gradients as the actual learning rate. Meanwhile, it calculates the update amount for each parameter separately, thereby realizing the dynamic adjustment of the learning rate for each parameter.

The most prominent advantage of the AdaGrad algorithm is that it does not require manual adjustment of the learning rate and exhibits excellent handling capability for sparse gradients, with the initial value usually only needing to be set to 0.001 [9]. This algorithm can accelerate the update speed of parameters with low update frequencies while slowing down the update pace of parameters with high update frequencies. In addition, AdaGrad can enhance the robustness of stochastic gradient descent (SGD). The formula for its gradient accumulation update step is as follows:

$$\theta_{t+1}=\theta_t-\frac{\eta}{\sqrt{G_t+\varepsilon}}\nabla_\theta J(\theta_t) \tag{11}$$

ε is usually set to 10−8 to avoid division by zero, while the step size η is set to 0.01. The advantage of this algorithm is its ability to automatically adjust the learning step size. However, it also has a drawback in that the learning rate update is highly dependent on the squared gradient. During each update process, the gradient vector remains positive, causing the denominator of the learning rate to accumulate continuously and gradually approach zero, which ultimately leads to the premature termination of algorithm iteration.

The improvement of the LMS adaptive filtering algorithm refers to the automatic adjustment of factors such as acquired data-dependent parameters, sequences, and conditions during data processing and analysis, so as to ensure the accuracy and efficiency of the algorithm [10]. Based on the previous analysis of the AdaGrad adaptive weight algorithm, this study integrates the AdaGrad intelligent algorithm with the LMS adaptive filtering algorithm, and realizes parameter optimization by adjusting the learning rate and parameter frequency of the algorithm at different stages. This embodies an adaptive learning rate concept:

(1) At the initial stage of iteration, the current position is far from the optimal point, so a large learning rate μ is adopted.

(2) As the number of iteration steps increases and the model gradually approaches the optimal point where the optimal solution is located, the learning rate μ can be reduced accordingly.

Such a design enables the learning rate to automatically adapt to the gradient changes during the optimization process, and the parameter iteration formula is as follows:

$$\theta_s = \theta_{s-1} - \frac{\mu}{\sqrt{\sum_{i=0}^{s-1} (g_i)^2}} g_{s-1} \tag{12}$$

The learning rate of this method is obtained by multiplying the current learning rate by the reciprocal of the sum of the squared gradients from the previous iteration, thereby enabling the automatic adjustment of the learning rate according to gradient changes.

In the improved LMS algorithm, this study draws on this idea and incorporates an adaptive learning rate mechanism based on historical gradient information, allowing the algorithm to independently update the step size for each weight parameter. The algorithm flow after the preliminary improvement is shown in the table below:

**Table 1.** Steps of the AdaGrad-LMS Algorithm

| Algorithm Adagrad-LMS | |
|---|---|
| 1 | Initialize: isr is the length of input xn, rps is for numerical stability, and M is the order |
| 2 | for k=M:isr |
| 3 | Obtain M sampling points of the current input xn and assign them to x |
| 4 | Current output $y = (W^{'}) * x$ |
| 5 | Calculation error $e(n) = d(n) - y(n)$ |
| 6 | Compute gradient $grad = e(n) * x$ |
| 7 | Update accumulated squared gradients $r = 0.96 * r + grad^{'} * grad;$ |
| 8 | Update Weights $W(:,k) = W(:,k-1) + mu/(rps + sqrt(r)) * grad$ |
| 9 | Calculate filtered output $yn(k) = W(:,k).^{'} * x$ |
| 10 | end |

It can be seen from Table 1 that the gradient descent algorithm takes the square root of the accumulated historical gradients as the denominator during the iteration process. This causes the learning rate to continuously decrease and eventually approach zero, leading to the premature termination of the algorithm before it reaches the optimal solution. To address this problem, the RMSProp algorithm is now introduced to improve the gradient descent-based LMS algorithm.

The improvement of the LMS algorithm using the AdaGrad gradient descent algorithm represents a fundamental optimization approach [11]. This method lacks an iterative decay mechanism and only dynamically adjusts the learning rate by accumulating the sum of squared historical gradients. Such an adjustment tends to cause the learning rate to prematurely enter a monotonically decreasing state as the iteration progresses, thereby affecting the subsequent optimization performance of the algorithm. Therefore, a decay coefficient ρ is introduced to adjust the update method of the accumulated squared gradients, which is modified to root mean square propagation. The iteration formula is as follows:

$$\theta_s = \theta_{s-1} - \frac{\mu}{\sqrt{v_s}} g_{s-1} \tag{13}$$

$$v_1 = g_0^{\ 2} \tag{14}$$

$$v_s = \alpha v_{s-1} + (1-\alpha)(g_{s-1})^2 \tag{15}$$

During the optimization of the LMS algorithm, the term root mean square refers to the square root of the mean of the squared gradients, while propagation means incorporating historical gradient data into the current gradient update process. Based on this logic, the core of the optimization process lies in adjusting the 7th step in Table 1 and redesigning the update method of the coefficient $w(n)$ in accordance with Equation (8). After the above improvements, the steps of the new algorithm are presented in Table 2.

**Table 2.** Steps of the RMSProp-LMS Algorithm

| Algorithm RMSProp-LMS | |
|---|---|
| 1 | Initialization: isr is the length of the input xn, error e(n), numerical stability constant rps, order M, and decay factor p |
| 2 | for $k{=}M : isr$ |
| 3 | Obtain M sample points of the current input xn and assign them to x |
| 4 | Current output $y = (W^{'}) * x$ |
| 5 | Calculation error $e(n) = d(n) - y(n)$ |
| 6 | Compute the gradient $grad = en * x$ |
| 7 | Update accumulated squared gradients $r = p * r + p(1\text{-}\ p) * grad^{'} * grad$ |
| 8 | Update Weights $W(:,k) = W(:,k-1) + mu/(rps + sqrt(r)) * grad$ |
| 9 | Calculate filtered output $yn(k) = W(:,k)^{'} * x$ |
| 10 | end |

It can be seen from Table 2 that, compared with Table 1, the core adjustment of the improved algorithm is reflected in the calculation logic of the accumulated squared gradients in the 7th step, with a decay coefficient ρ added. Specifically, the new root mean square obtained by processing the squared gradient values through exponentially weighted moving average can form a smoothed estimate of the magnitude of the parameter gradients, thereby providing a reliable basis for the learning rate adjustment of each parameter. This calculation method of historical squared gradients based on exponentially weighted moving average can gradually weaken the impact of early gradients on the current step size adjustment as the iteration progresses. It can not only maintain the effective learning rate required by the algorithm and avoid the learning rate from falling into an extremely small state prematurely, but also specifically address the core problem of the rapid decline of the learning rate in gradient descent algorithms.

## 3.2. Algorithm Optimization

Now, parameter optimization is performed on the improved algorithm. According to the literature investigation, the performance optimization of the RMSProp-LMS algorithm can be achieved through parameter adjustment, involving four key parameters: algorithm order, initial weight, learning rate, and maximum number of iterations. This study first conducts a parameter optimization research on the RMSProp-LMS algorithm to explore its optimal performance.

(1) Adjustment of Algorithm Order

The algorithm order is a key factor affecting prediction accuracy, and a reasonable order setting is an important prerequisite for the algorithm to achieve excellent performance [12]. In this study, 20% of the total data volume is initially selected as the algorithm order, corresponding to a specific value of 150. Figure 2 presents the algorithm error distribution when the order is set to 100, from which it can be observed that the error values of multiple sampling points have exceeded 0.2000. Considering that the amplitude of the original signal is 0.16, this error level is beyond the acceptable range. Simulation results show that the average error of the algorithm under this order is approximately 0.2135, with a time complexity of 0.0083 s. The overall performance fails to meet expectations and thus requires further optimization. Two approaches will be adopted subsequently to adjust the order parameter: one is to increase the order gradually, and the other is to decrease the order appropriately, so as to explore the optimal parameter configuration.

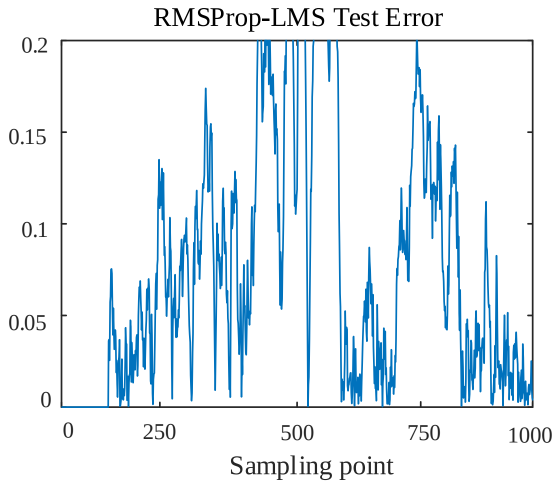


**Figure 2.** Algorithm Error Diagram at Order 100

The first optimization approach is to increase the algorithm order. Simulation tests are carried out with different orders including 100, 130, 150, 170, 190 and 200. The results indicate that increasing the order does not improve the algorithm performance; on the contrary, it leads to a continuous deterioration of the performance. Taking the test results with an order of 200 as an example, in the error distribution shown in Figure 3, the error values at multiple sampling positions have exceeded 0.25. Simulation data show that when the order is 200, the average error of the algorithm is approximately 0.2678, with a time complexity of 0.0065 s. This error level has caused severe interference to the original signal. Therefore, the first optimization approach is not feasible, and it is necessary to switch to the second approach for further research.

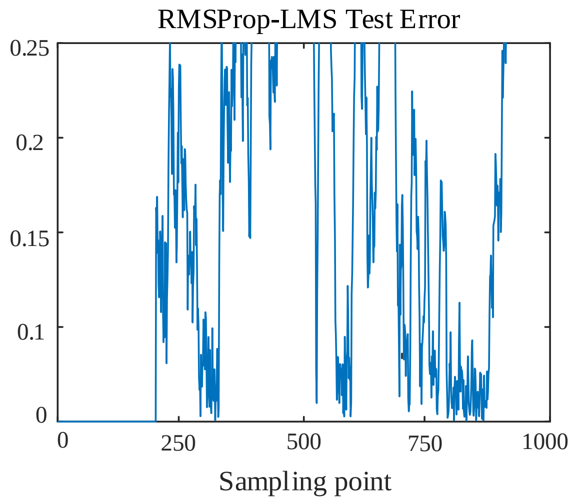


**Figure 3.** Algorithm Error Diagram at Order 200

The second optimization approach is to reduce the filter order. Simulation analyses are conducted with orders including 40, 30, 20, 10, 5 and 4. Simulation results show that with the decrease of the order, the average error of the algorithm presents a downward trend, but this trend is not continuous; instead, there exists an obvious critical interval. When the order ranges from 4 to 10, the average error reaches the minimum value; once the order is lower than 4, the average error starts to increase inversely. Based on this, 4 is determined as the final order of the algorithm. Figure 4 shows the error characteristic diagram when the order is 4, where the errors of the vast majority of sampling points are controlled within 0.1800, with only a few sampling points exceeding this range. The corresponding simulation data are as follows: the average error is approximately 0.1309, and the time complexity is 0.0064 s, indicating a favorable overall optimization effect.

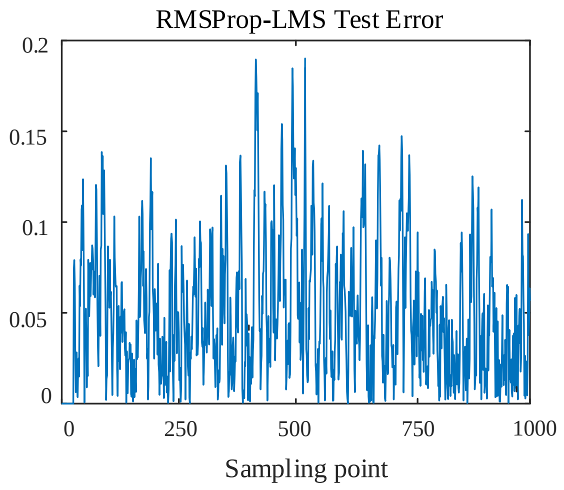


**Figure 4.** Algorithm Error Diagram at Order 4

(2) Adjustment of Initial Weight

The initial weight has a significant impact on algorithm accuracy, and an appropriate value needs to be selected. Considering that the initial weight usually needs to be set to a small value, combined with the results of literature investigation, 0.4 is selected as the initial weight for testing. Figure 5 shows the error distribution under this initial weight, where the errors of most sampling points have exceeded 0.6, and the errors of some points have even reached 1.4. Simulation data indicate that the average error is approximately 0.5632 with a time complexity of 0.0065 s. The performance is unsatisfactory and requires further adjustment. In view of the fact that the current value of the initial weight is still relatively large, it is necessary to reduce the initial weight to achieve the expected performance.

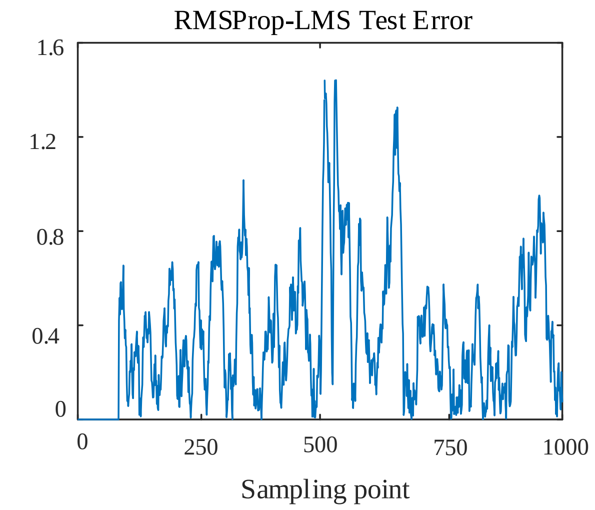


**Figure 5.** Algorithm Error Diagram at Initial Weight of 0.4

Simulation analyses are carried out with different initial weight parameters including 0.2, 0.15, 0.1 and 0.05. The results show that with the decrease of the initial weight, the algorithm performance shows a gradual optimization trend, but there exists a critical value characteristic: when the initial weight is lower than 0.15, the algorithm will experience an abrupt increase in error when processing part of the data. Based on the simulation results of multiple groups of data, 0.15 is finally determined as the optimal initial weight. Figure 6 shows the error distribution diagram under this initial weight, where the errors of most sampling points are controlled within 0.09. The corresponding simulation data are as follows: the average error is approximately 0.0732, with a time complexity of 0.0063 s.

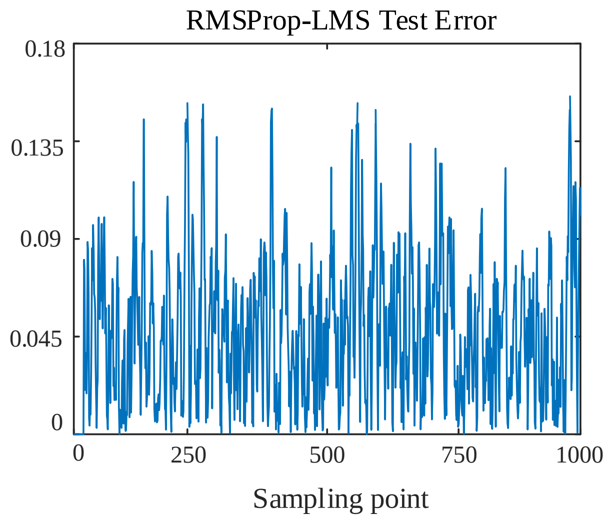


**Figure 6.** Algorithm Error Diagram at Initial Weight of 0.15

(3) Adjustment of Learning Rate

Simulation tests are conducted with different learning rate parameters including 0.008, 0.006, 0.004, 0.002 and 0.001. The results show that the algorithm error follows a trend of first decreasing and then increasing with the variation of learning rate, with the minimum error achieved when the learning rate is set to 0.002. Figure 7 presents the error distribution diagram under this learning rate, where the errors of most sampling points are controlled below 0.1. Simulation data indicate that the corresponding average error is approximately 0.0674 with a time complexity of 0.0063 s, which fully meets the expected design requirements.

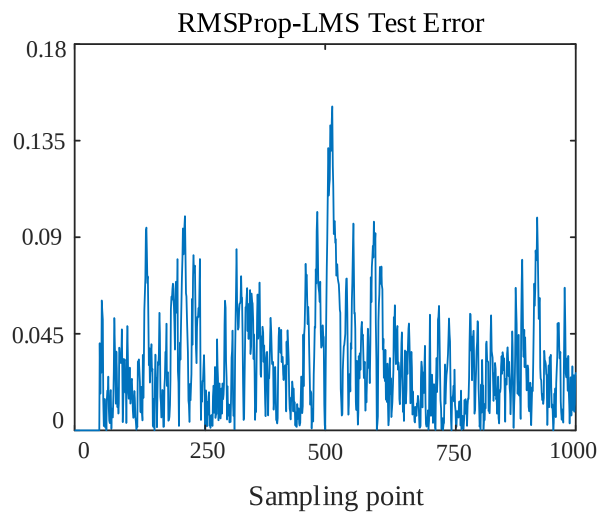


**Figure 7.** Algorithm Error Diagram at Learning Rate of 0.002

(4) Adjustment of Iteration Number

The iteration number is not only correlated with the algorithm error level, but also has an impact on its time complexity. Therefore, when adjusting the iteration number, it is necessary to balance the dual requirements of low error and low time complexity. Figure 8 presents the simulation characteristic diagram of the iteration number, and the results show that: when the iteration number exceeds 5, the error improvement effect tends to be stable, while the time complexity increases significantly; when the iteration number is less than 5, although the time complexity decreases to a certain extent, the error increases sharply. Based on a comprehensive trade-off between the two factors, 5 is finally determined as the optimal iteration number.

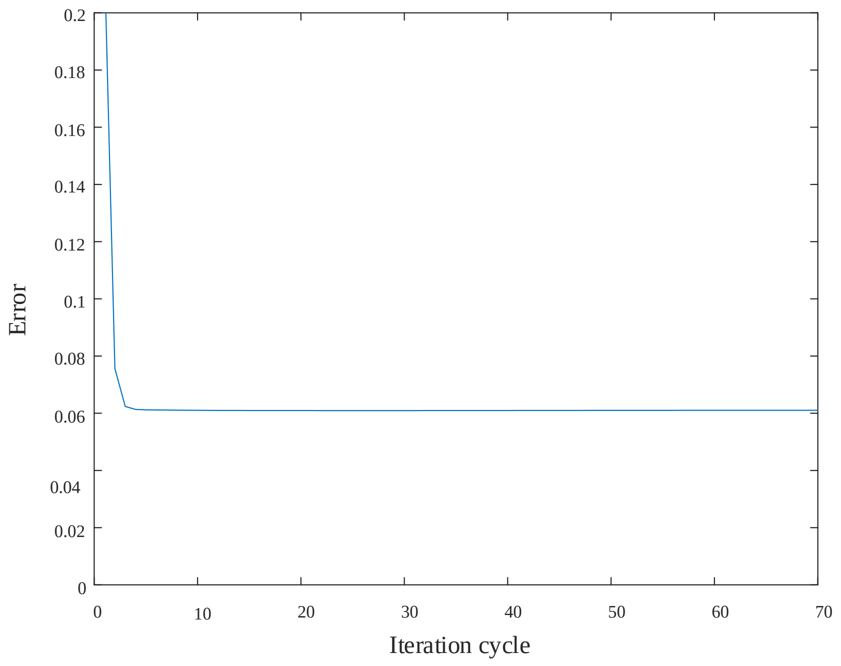


**Figure 8.** Simulation Diagram of Iteration Cycles

After parameter optimization, the final parameter configuration of the RMSProp-LMS algorithm is as follows: order, initial weight, learning rate, and maximum iteration number. Figure 9 shows the algorithm error distribution diagram after parameter debugging. The errors of the vast majority of sampling points are controlled within 0.0900. Simulation results indicate that the average error of the algorithm under this parameter configuration is 0.0473.

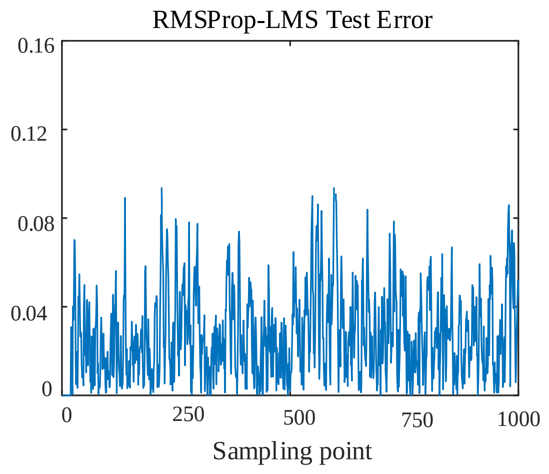


**Figure 9.** Error Diagram of the RMSProp-LMS Algorithm

## 4. ALGORITHM SIMULATION AND ANALYSIS

After parameter optimization of the RMSProp-LMS algorithm, an experimental framework is constructed by invoking the LMS filtering module and the RMSProp-LMS filtering module via the model, and corresponding processing results are generated [13]. Thus, a simulation comparison between the improved algorithm and the original algorithm is conducted under consistent experimental conditions. The preliminary arrangement results of the time-domain waveforms before and after filtering obtained from the experiment are shown in Figure 10. To simulate the complex characteristics of data in real-world scenarios, the study introduces the AR noise model and the MA noise model to perform secondary processing on additive white Gaussian noise (AWGN) for injecting uncertainties. This processing method makes the experimental model more consistent with the dynamic variation laws of the real world, thereby enhancing the authenticity and scene simulation fidelity of the experiment.

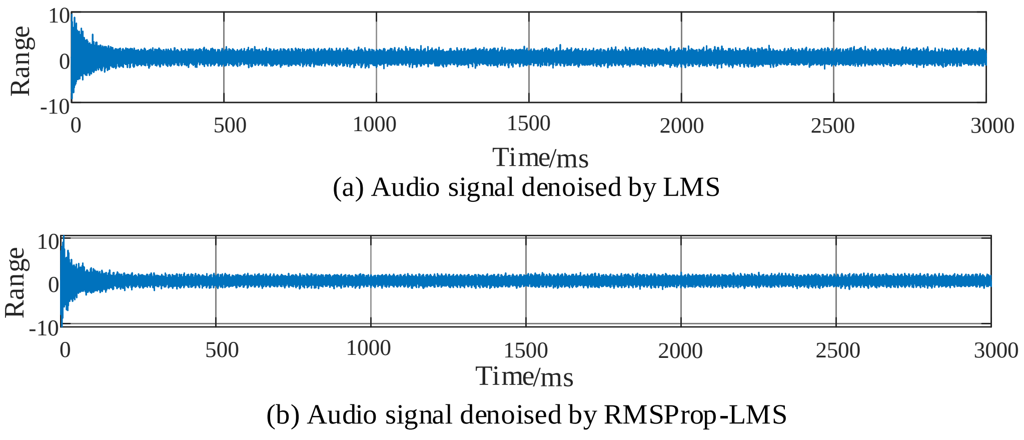
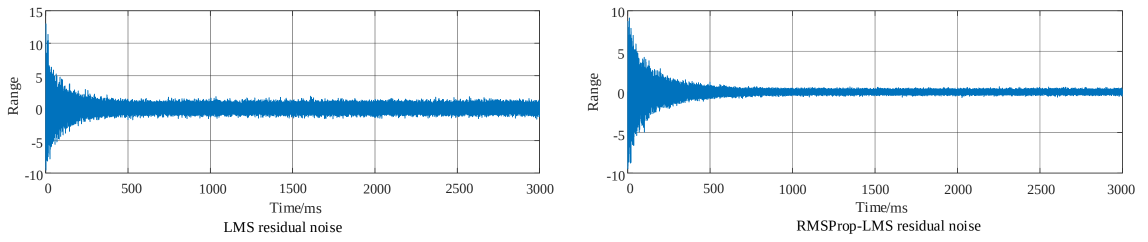


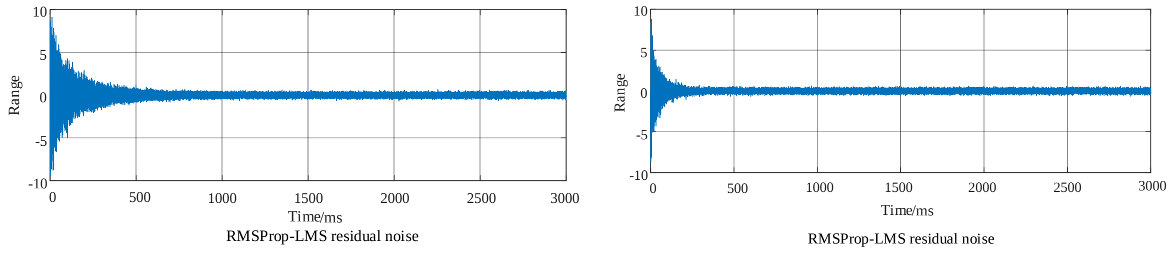
**Figure 10.** Time-Domain Comparison Diagram of Denoising Performance between RMSProp-LMS and LMS Algorithms

It can be concluded from the results in Figure 10 that the improved algorithm can still effectively complete the filtering task in harsh channel environments. From the perspective of time-domain waveforms, although the waveforms after denoising exhibit obvious distortion in the initial stage, they can quickly approximate the waveforms of the original audio signals and gradually achieve signal restoration. In addition, according to the parameter conditions set in the experiment, the initial step size is determined as η0. The filtering performance can be further optimized by adjusting the step size in subsequent research. Based on this, a more detailed comparative analysis of the filtering effects of the improved algorithm will be carried out in the following content, so as to further verify the effectiveness and superiority of the improved algorithm.

This experiment is designed on the basis of the block diagram of the equalizer principle. First, the signal-to-noise ratio (SNR) of the input signal is set to -6 dB. Then, by adjusting the specific value of the filter order M in the initial parameters, independent simulation experiments are conducted under two parameter configurations of M1and M2, focusing on observing the time-domain waveform characteristics of the residual noise after filtering. The experimental results show that after the algorithm enters the convergence state, the residual noise in the latter half of the time-domain waveform shows a consistent variation trend, and the noise amplitude is maintained at a low level. To more clearly demonstrate the dynamic variation law of noise in the pre-convergence stage, the time-domain waveform of the first 3 seconds is selected for analysis, and the specific results are shown in Figure 11(a), (b), (c) and (d).



(a) LMSResidual Noise Waveform at M=32 (b) RMSProp-LMSResidual Noise Waveform at M=32



(c) LMS Residual Noise Waveform at M=4 (d) RMSProp-LMSResidual Noise Waveform at M=4

**Figure 11.** Comparison of residual noise waveforms between RMSProp-LMS and LMS

Under the condition of filter order M=32, the time required for the noise level of the two algorithms to reach a steady state after filtering was observed. The results show that the convergence time of the RMSProp-LMS algorithm is approximately 0.5 s, while that of the traditional LMS algorithm is 0.65 s, representing a convergence speed improvement of about 23%. In terms of noise amplitude, the RMSProp-LMS algorithm exhibits a lower steady-state amplitude level, indicating a more favorable filtering performance. When the order is reduced to M=4, the difference in convergence speed between the two algorithms further widens: the RMSProp-LMS algorithm can achieve convergence in only about 0.23s, whereas the traditional LMS algorithm requires 0.4s, corresponding to a convergence speed improvement of approximately 60%. This demonstrates that the RMSProp-LMS algorithm has a more significant convergence performance advantage under the low-order filter configuration.

In terms of mean squared error (MSE), different filter orders have a significant impact on algorithm performance. At a filter order of M=32, the filtered output MSE of the traditional LMS algorithm is 1.433×10−3 dB, while the MSE of the improved RMSProp-LMS algorithm after filtering is reduced to 1.398×10−3dB, representing a 2.67% reduction in error amplitude compared with the traditional algorithm. When the filter order is M=4, the steady-state error advantage of the improved algorithm is further highlighted: the filtered MSE of the traditional LMS algorithm is 1.304×10−3dB, while that of the RMSProp-LMS algorithm is only 0.795×10 −3dB, with a 42% reduction in error amplitude.

To eliminate the accidental error that may exist in a single experiment and ensure the reliability of the experimental results, further Monte Carlo simulation experiments were conducted. A total of 200 independent simulation tests were completed by gradually increasing the input signal-to-noise ratio (SNR). The relevant steady-state error data of the algorithms under the above different filter orders were statistically organized, and the average value of each index was calculated. The final summarized results are presented in Table 3.

**Table 3.** Monte Carlo Simulation Data of the LMS Algorithm

| Algorithm complexity | Algorithm Name | Convergence Time / s | MSE/dB |
|---|---|---|---|
| M=32 | LMS | 100 | 1.564 |
| | RMSProp-LMS | 89 | 1.359 |
| | Average Improvement Rate | 11% | 13.1% |
| M=4 | LMS | 105 | 1.556 |
| | RMSProp-LMS | 32 | 1.225 |
| | Average Improvement Rate | 69.5% | 21.27% |

It can be concluded from the statistical data in Table 3 that, verified by multiple repeated simulation tests, the RMSProp-LMS algorithm maintains superior convergence accuracy and convergence speed compared with the traditional LMS algorithm under both filter order configurations of M=4 and M=32. In particular, such performance advantages are more prominent when the filter order is at a low level. This result is highly consistent with the core characteristics of the RMSProp-LMS algorithm. Moreover, with the increase in the input signal-to-noise ratio (SNR), the improvement of the output SNR achieved by the improved algorithm over the original algorithm becomes increasingly significant. Simulation results show that in harsh channels with an input SNR above -6 dB, the RMSProp-LMS algorithm achieves an average convergence speed improvement of 69.5% and an average output SNR improvement of 17.7% compared with the original LMS algorithm.

## 5. CONCLUSION

This paper focuses on the goal of improving the anti-interference capability of the pressure test system, and completes the improvement and verification of the adaptive filtering algorithm. Aiming at the problem of complex noise interference existing in the experimental environment of the pressure test system, a denoising method based on the improvement of the adaptive filtering algorithm is proposed, and the superiority of the improved algorithm is verified. To address the problems of slow convergence speed and insufficient time-varying signal tracking capability in non-stationary noise environments, the RMSProp adaptive learning rate adjustment mechanism is introduced to optimize the algorithm weight update process. By assigning adaptive learning rates to different parameters during the iteration process, the problem of convergence stagnation caused by premature learning rate attenuation is effectively avoided. By suppressing the oscillation in the gradient update process, the stability and robustness in time-varying noise environments are significantly improved. Experimental results show that the improved algorithm achieves a 69.5% increase in convergence speed and a 17.7% improvement in signal-to-noise ratio (SNR), which enhances the accuracy of signal processing and the anti-interference capability of the system.

Although the RMSProp-LMS algorithm exhibits excellent filtering performance in environments with an SNR of -6 dB, its performance degrades significantly in strong interference scenarios with lower SNRs. On the one hand, gradient calculation is susceptible to outliers in strong noise environments, leading to an increase in the estimation deviation of accumulated squared gradients and thus causing distortion in learning rate adjustment. On the other hand, the algorithm has a strong dependence on the differences in statistical characteristics between the input signal and noise. When the noise exhibits non-Gaussian and impulsive characteristics, the optimization objective based on mean squared error (MSE) minimization cannot effectively separate signal and noise components. As a result, residual interference remains in the filtered signal, making it difficult to meet the high-precision testing requirements in extremely harsh environments.

## REFERENCES

[1] Vandermark R L, Brennan R J, Ehlert A K, et al. Estimating net energy for activity for grazing beef cattle by integrating GPS tracking data, in-pasture weighing technology, and animal nutrition models [J]. Frontiers in Veterinary Science, 2025, 121620584-1620584.

[2] Yan W, Ren H, Luo X, et al. Hybrid-data-driven bridge weigh-in-motion technology using a two-level sequential artificial neural network [J]. Computer-Aided Civil and Infrastructure Engineering, 2025, 40(20): 2992-3012.

[3] A K R, M R K. Measuring weight with electronic scales in clinical and research settings during the coronavirus disease 2019 pandemic [J]. Obesity (Silver Spring, Md), 2020, 28(7): 1182-1183.

[4] Ma W, Li Q, Li J, et al. A method for weighing broiler chickens using improved amplitude-limiting filterin algorithm and BP neural networks [J]. Information Processing in Agriculture, 2020, (prepublish).

[5] Liu X, Yang Z, Shi B. Weigh-in-Motion Method Based on Modular Sensor System and Axle Recognition with Neural Networks [J]. Applied Sciences, 2025, 15(2): 614-614.

[6] Socha A, Izydorczyk J. Strain Gauge Calibration for High Speed Weight-in-Motion Station [J]. Sensors (Basel, Switzerland), 2024, 24(15): 4845-4845.

[7] Researchers at University of Science and Technology Beijing Target Machine Learning (Investigation of the Temperature Compensation of Piezoelectric Weigh-In-Motion Sensors Using a Machine Learning Approach) [J]. 2022(Apr. 13): 32-33.

[8] Yuan L, Wang Z, Li H, et al. Piezoresistive Pressure Sensors: Synergistic Resistance Modulation toward Ultrahighly Sensitive Piezoresistive Pressure Sensors (Adv. Mater. Technol. 4/2020) [J]. Advanced Materials Technologies, 2020, 5(4): n/a-n/a.

[9] SLKOR Semiconductor Launches New Products: RS485 Chips, CAN Chips, and Half-Bridge Drivers [J]. M2 Presswire, 2025.

[10] Ma T, Lu G, Qin Z, et al. STM32-Based Control System Design for Orbiting Binocular Vision Mobile Platforms: Regular Papers [J]. Journal of Robotics and Mechatronics, 2025, 37(5): 1230-1245.

[11] Turek P. Comparison of PID Controller Settings for an Active Bearing Support Controlled Using the LabVIEW Software Environment [J]. Applied Sciences, 2025, 15(23): 12826-12826.

[12] Oh J A, Stoddard J C, Queenan C, et al. Efficient and affordable thermoelectric measurement setup using Arduino and LabVIEW for education and research [J]. American Journal of Physics, 2025, 93(12): 991-999.

[13] Wu R, Yang Z. Research on an Industrial IoT-Enabled Pipeline Micro-Vibration Detection System Based on LabVIEW and Proteus [J]. Internet Technology Letters, 2025, 8(6): e70149-e70149.