PAPER Special Section on Cryptography and Information Security

A Combined Fast Adaptive Filter Algorithm with an Automatic Switching Method

Youhua WANG† and Kenji NAKAYAMA†, Members

SUMMARY This paper proposes a new combined fast algorithm for transversal adaptive filters. The fast transversal filter (FTF) algorithm and the normalized LMS (NLMS) are combined in the following way. In the initialization period, the FTF is used to obtain fast convergence. After converging, the algorithm is switched to the NLMS algorithm because the FTF cannot be used for a long time due to its numerical instability. Nonstationary environment, that is, time varying unknown system for instance, is classified into three categories: slow time varying, fast time varying and sudden time varying systems. The NLMS algorithm is applied to the first situation. In the latter two cases, however, the NLMS algorithm cannot provide a good performance. So, the FTF algorithm is selected. Switching between the two algorithms is automatically controlled by using the difference of the MSE sequence. If the difference exceeds a threshold, then the FTF is selected. Other wise, the NLMS is selected. Compared with the RLS algorithm, the proposed combined algorithm needs less computation, while maintaining the same performance. Furthermore, compared with the FTF algorithm, it provides numerically stable operation.

key words: *adaptive filters, fast RLS, normalized LMS, tracking speed, stability*

1. Introduction

The LMS algorithm is highly popular for its simplicity. A main drawback of the LMS algorithm, however, is its low convergence rate. In order to solve this problem with a reasonable computational load, various fast RLS algorithms have been proposed [1]. Unfortunately, the fast RLS algorithms suffer from the numerical instability problem. Much research has been done to improve these two classes of adaptive filter algorithms.

In the class of the LMS algorithm, for example, varying the step size according to the sign of the gradient or the moving average of the misadjustment level is proposed in [2] and [3] to improve the convergence rate. Conjugate gradient technique is another method used to speed up convergence [4]. These efforts can improve the convergence rate to some extent, but are still uncomparable to that of the RLS algorithm.

On the other hand, various rescue methods have been proposed to solve the numerical instability problem in the fast RLS algorithms. They include resetting the internal parameters just before some rescue variables become negative [5], reintroducing some error feedbacks into the fast RLS algorithms [6], and using alternately two fast RLS algorithms in parallel [7]. These rescue methods can elongate the stable implementation of the fast RLS algorithms. However, completely stable implementation can not be obtained.

Another advantage of the RLS algorithm over the LMS algorithm seems to be the high tracking speed for a time varying system. Bershed et al. have shown, however, that in some applications, the LMS algorithm has a superior tracking performance [8]. Eleftherion et al. have also discussed the tracking performances of the RLS and the LMS algorithms. They gave a boundary condition based on the weight lag error. Under this boundary condition, there is no advantage in using the more complex RLS algorithm [9]. In this paper, a boundary condition based on both the weight noise and the weight lag errors is introduced. It can be easily shown that the tracking performance of the NLMS algorithm is superior to that of the RLS algorithm under certain conditions.

In order to best utilize the advantages of the fast RLS and the NLMS algorithms for achieving computational simplicity, numerical stability, fast convergence rate, and fast tracking, we propose a new method in which the NLMS and the FTF algorithms are combined. An automatic switching method between the two algorithms is also proposed. The proposed method is investigated in stationary and nonstationary environments and compared with the NLMS and the RLS algorithms.

This paper is organized as follows. In Sect. 2, the performances of the fast RLS and the NLMS algorithms are discussed. A modified NLMS is proposed to reduce misadjustment. In Sect. 3, we first describe the combination method of the two algorithms. Then, we compare the proposed method with the periodic restarting method. Finally, we discuss the boundary condition of tracking ability between the NLMS and the RLS algorithms. Description of the simulation and discussions of the results are given in Sect. 4.

Manuscript received May 6, 1993.
Manuscript revised May 27, 1993.

†The authors are with the Graduate School of Natural Sci. & Tech., Kanazawa University, Kanazawa-shi, 920 Japan.

2. Fast RLS and the Normalized LMS

2.1 Performance of the Fast RLS Algorithm

There are several kinds of fast RLS algorithms. A representative one is the FTF algorithm [5]. An important feature of this algorithm is that it can provide not only an exact solution to the RLS problem but also a computational cost that increases linearly with the number of taps contained in the adaptive filter, as in the LMS algorithm.

However, the numerical instability problem encountered in the FTF algorithm greatly impairs its practical applications. The numerical instability caused in transient and in steady-state has been discussed in several papers [5],[6].

In a transient period, such as initialization and reinitialization, the instability is caused mainly by finite-precision and large-order effects [5]. An effective method to overcome this problem is to introduce a so-called soft constraint initialization. That is to suppose that the tap input $u(-M)$ is equal to $(\lambda^{-M}\delta)^{1/2}$ rather than zero. Like in the RLS algorithm, using the soft constraint initialization will introduce a bias into the estimate of $\hat{\mathbf{w}}(n)$. This bias approches zero when the iteration n increases. Appropriate choice of the value of δ can stabilize the FTF algorithm during the transient period, and introduce a slightly suboptimal RLS solution. Stability during the transient period is very important, since initialization and reinitialization are repeatedly used in the proposed algorithm.

In a steady-state period, the instability is caused mainly by the accumulation of roundoff errors due to finite-precision effects. The structure of the FTF algorithm further increases the trend of this accumulation, since all the redundant information contained in the RLS algorithm is removed. By reintroducing some error feedbacks into the FTF algorithm, its stability performance is improved [6]. However, the stability can be maintained only when certain conditions are satisfied. In practical applications, the rescue is still essential for overcoming the instability problem of the FTF algorithm.

Another serious problem encountered in the FTF is its tracking ability. The FTF algorithm is very sensitive to the variation of the forgetting factor λ. In a nonstationary environment, where λ should be less than unity in order to obtain the tracking ability, we find that the FTF algorithm has an increased instability. In order to overcome this problem, use of a gear-shifting method instead of changing the forgetting factor is proposed in [5]. This is to redefine time $n = 0$ as the time at which the gear-shifting is to occur and then to weight the present tap weights $\hat{\mathbf{w}}(n)$ through an appropriate choice of μ. Since $\mu = \lambda^{-M}\delta$, this method is in fact the same as initialization with a soft constraint condition. In other words, the tracking ability is obtained by discarding all previous tap inputs. However, in a nonstationary environment, where continuous change of the tracking is necessary, this method is inefficient. Many questions remain open regarding the improvement of the tracking property of the fast RLS algorithms.

2.2 Performance of the Normalized LMS

The NLMS algorithm can be written as [1]

$$\alpha(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \tag{1}$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu_{nlms}(n)\mathbf{u}(n)\alpha^*(n), \tag{2}$$

where $\mu_{nlms}(n) = \tilde{\mu}/\|\mathbf{u}(n)\|^2$. $\|\mathbf{u}(n)\|^2$ is the squared Euclidean norm of the tap-input vector $\mathbf{u}(n)$, and $\tilde{\mu}$ is a positive real scaling factor.

The main advantage of the NLMS over the LMS is twofold. Firstly, when $\tilde{\mu} = 1$ the NLMS algorithm can provide the fastest convergence rate among the scalar step sized LMS algorithms [1]. Secondly, $\mu_{nlms}(n)$ is a variable step size, which equals the reciprocal of the instantaneous estimate of the input power. So the NLMS algorithm automatically satisfies the stability condition and does not require the prior knowledge of the input power.

However, the problem caused by using the NLMS algorithm is its large misadjustment. If the NLMS algorithm is implemented in a stationary environment and $\tilde{\mu} = 1$, the misadjustment can reach 100%. This can be explained by using the equation developed in [1]. The misadjustment can be calculated as

$$\mathcal{M} \approx \frac{\mu \sum_{i=1}^{M} \lambda_i}{2 - \mu \sum_{i=1}^{M} \lambda_i} \tag{3}$$

where $\sum_{i=1}^{M} \lambda_i = \mathrm{tr}(\mathbf{R}) = \mathrm{tr}(E[\mathbf{u}(n)\mathbf{u}^H(n)]) = E[\mathrm{tr}(\mathbf{u}(n)\mathbf{u}^H(n))] = E[\|\mathbf{u}(n)\|^2]$. The statastical performance of $\mu_{nlms}(n)$ is

$$\mu = E[\mu_{nlms}(n)] = \frac{1}{E[\|\mathbf{u}(n)\|^2]} \tag{4}$$

So we get the misadjustment of the NLMS algorithm

$$\mathcal{M} \approx \frac{\mu \sum_{i=1}^{M} \lambda_i}{2 - \mu \sum_{i=1}^{M} \lambda_i} = \frac{1}{2-1} = 100\% \tag{5}$$

Choosing $\tilde{\mu} < 1$ can reduce the misadjustment, but at the same time it reduces the convergence rate.

In order to overcome this problem, we introduce a modified step size into the NLMS algorithm. Use of the modified step size can produce a fast convergence rate when the MSE is large, and obtain a small misadjustment when the MSE is small.

The modified step size can be written as follows

$$\mu(n) = \frac{1}{\|\mathbf{u}(n)\|^2(1 + a\hat{J}_{min}/\alpha^2(n))} \tag{6}$$

where $\alpha(n)$ is the error calculated in Eq.(1), $\hat{J}_{min}$ is the estimate of the minimum MSE, a is a positive constant. Appropriate choice of a can provide a good trade-off between fast tracking and small misadjustment.

The performance of the modified step size is follows. When $\alpha^2(n)$ is large, $a\hat{J}_{min}/\alpha^2(n) << 1$ and $\mu(n) \approx 1/\|\mathbf{u}(n)\|^2$. Thus a fast convergence rate is obtained. When $\alpha^2(n)$ approaches J_{min}, we have

$$\begin{aligned}\mu(n) &\approx \frac{1}{\|\mathbf{u}(n)\|^2(1 + a\hat{J}_{min}/J_{min})} \\ &\approx \frac{1}{\|\mathbf{u}(n)\|^2(1+a)}\end{aligned} \tag{7}$$

and the misadjustment is therefore reduced to about $\frac{100}{1+a}\%$.

3. Combination of the FTF and the NLMS

The main idea, on which the proposed algorithm is based, is that whenever the tap weights have a large deviation from their optimum values and a large error results, the FTF algorithm is used to turn the tap weights quickly back to the close proximity of their optimum values. Since the FTF can provide the least square solution like the RLS algorithm, usually only 2M - 3M iterations is needed to make the tap weights closely enough to the optimum values . After that, the NLMS algorithm is used. Thus, fast convergence rate can be obtained and numerical instability avoided. When the unknown system varies slowly with time, we usually need not use the FTF algorithm, since the NLMS algorithm is capable of tracking slow time varying system. When the unknown system varies fast with time, fast tracking can be obtained by periodically implementing the FTF algorithm.

3.1 Combination Method

Figure 1 shows the timing of operations in the proposed algorithm. We let the FTF algorithm always perform in the transient period (period 1, 3, 5), and the NLMS algorithm in the steady-state period (period 2, 4). The interval of implementing the FTF algorithm is fixed (for example 3M iterations) in order to guarantee the stability. Automatically switching from one algorithm to the other is determined by a threshold Θ. Θ is a prescribed value which can be defined as

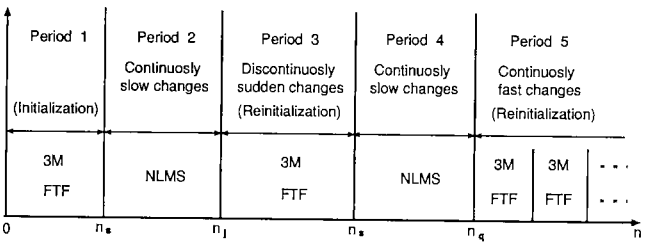


Fig. 1 Timing of operations in the proposed method.

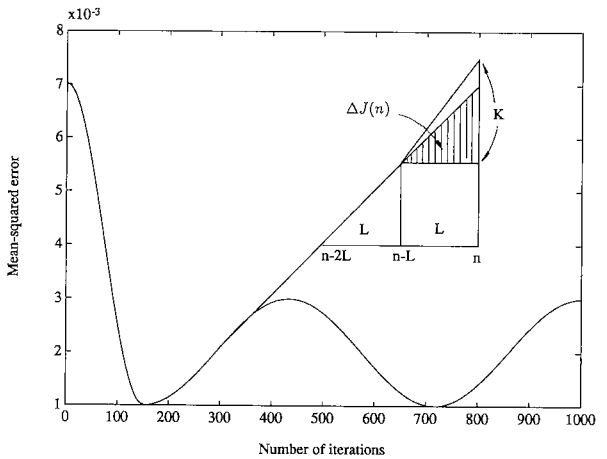


Fig. 2 Calculation of the slope of the MSE.

$$\Theta = \frac{1}{2}KL \tag{8}$$

where K is a positive constant which represents the slope or the speed of variation of the MSE as shown in Fig. 2. L is the number of samples used for averaging the MSE.

If the difference of the MSE $\Delta J_r(n)$ is greater than the threshold Θ, the FTF algorithm is used, otherwise, the NLMS algorithm is used. From Fig. 2, we can write $\Delta J_r(n)$ as

$$\Delta J_r(n) = \frac{|\Delta J(n)|}{\hat{J}_{min}} = \frac{1}{2}\left(\frac{|J_1(n) - J_2(n)|}{\hat{J}_{min}}\right) \tag{9}$$

$$J_1(n) = \sum_{i=n-L+1}^{n} \alpha^2(i) \tag{10}$$

$$J_2(n) = \sum_{i=n-2L+1}^{n-L} \alpha^2(i) \tag{11}$$

where $\hat{J}_{min}$ is the estimate of the minimum MSE.

The reason for adding $\hat{J}_{min}$ into the calculation of $\Delta J_r(n)$ is twofold. Firstly, $\Delta J(n)$ caused by both the weight noise and the weight lag can be considered as an increment of the misadjustment $\Delta\mathcal{M}$ from J_{min}. For example, suppose $\Delta J(n) = 0.005$. If $J_{min} = 0.001$, then $\Delta\mathcal{M} = 500\%$. So the FTF algorithm should be used, in order to obtain fast tracking. However, if $J_{min} = 0.01$, then $\Delta\mathcal{M} = 50\%$. In this case, the NLMS algorithm

should be used, in order to obtain a small final misadjustment. Secondly, we know that the performance of the FTF algorithm is closely related to J_{min}. If $J_{min} = 0$, the FTF achieves its best performance. When J_{min} increases, however, the superior convergence and tracking performance of the FTF over the NLMS becomes lost [1]. So when the combined algorithm is implemented in a nonstationary environment, the FTF algorithm should be used less frequently for a large J_{min}. Apparently, Eq.(9) satisfies these requirements [10].

In practical situations, Eqs.(8) and (9) can be written in another simple equivalent form

$$\Theta = K\hat{J}_{min}L \tag{12}$$

$$\Delta J_r(n) = |J_1(n) - J_2(n)| \tag{13}$$

In order to obtain fast tracking when the unknown system varies fast with time and to avoid the possible instability, the reinitialization of the FTF algorithm is periodically implemented in period 5. One may think that the discontinuities caused by the reinitialization will produce an unbearable large MSE. This is true if one chooses the reinitialization parameters δ and λ improperly. There are two factors that can cause discontinuities. One is the transient produced by removing the augmentation of a series of zero tap inputs before reinitialization. The transient makes the desired signal $d(n)$ undergo about M iterations of transient period, in which $d(n)$ do not contain the correct information of the unknown system. This problem can be solved by choosing $\lambda < 1$ to avoid accumulating incorrect data. The other factor is the input correlation matrix being nearly singular when the reinitialization parameter δ is too small. By increasing δ, the possible singularity of the correlation matrix can be avoided. The bias produced by a large δ can be suppressed by reducing λ. Generally speaking, by properly choosing δ and λ, we can obtain a tracking performance like that of the RLS algorithm without introducing noticeable discontinuities.

Figure 3 shows the flow chart of the combination of the two algorithms. We can see that the difference between the FTF and the NLMS algorithms is the calculation of the Kalman gain vector $\mathbf{k}_M(n)$. In the FTF algorithm, $\mathbf{k}_M(n)$ is calculated by using the relationship between forward and backward prediction instead of complex matrix manipulation. We note that all the information contained in the inverse input correlation matrix $\Phi^{-1}(n)$ is also contained in $\mathbf{k}_M(n)$. So $\mathbf{k}_M(n)$ is the true Kalman gain. In the NLMS algorithm, however, $\mathbf{k}_M(n)$ is replaced by a simple scalar step size multiplied by the tap-input vector. Apparently, $\mathbf{k}_M(n)$ is only an approximation of the true Kalman gain. Experiments show that the nearer the bottom of the error surface, the better the approximation.

The proposed algorithm has several important features. First, we note that the implementation of the FTF algorithm is within about 3M iterations. Keeping the FTF algorithm stable within this interval is relatively simple. Thus, some important advantages of the FTF over the NLMS, such as fast convergence rate and fast tracking, can be utilized. Second, the gear-shifting property can be realized in the proposed algorithm, as a small λ can be used in the FTF to provide fast tracking in a transient and a small step size can be used in the NLMS to provide a small misadjustment in a steady-state operation.

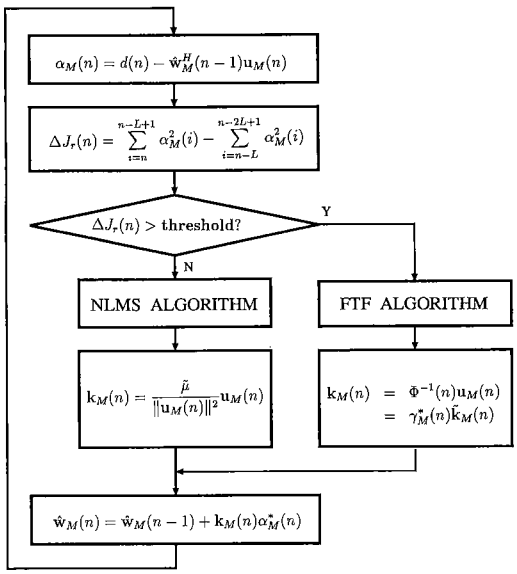


Fig. 3 Flow chart of the proposed combination method.

3.2 Comparison Between the Proposed Method and the Periodic Restarting Method

The periodic restarting method, which was introduced in [9] to overcome the instability problem of the FTF algorithm, also uses the combination of the FTF and the LMS algorithms. The basic idea, however, is entirely different from the proposed method.

Figure 4 shows the timing of operations in the periodic restarting method. Comparing with Fig. 1, we can see that the use of the two algorithms is inversed. The periodic restarting method takes the LMS as an auxiliary algorithm to provide an estimate of the desired response when the FTF algorithm is reinitialized. In contrast, the proposed method takes the FTF as an auxiliary algorithm to provide the least square solution, when the characteristics of the unknown system are suddenly changed or varying fast with time. The periodic restarting method may result in a significant reduction in tracking speed [1]. This is because the LMS algorithm used in the period restarting method can be implemented away from the bottom of the error surface if the jumping parameters happen to appear during reinitialization of the FTF algorithm. Apparently, the LMS algorithm cannot give a good perfor-

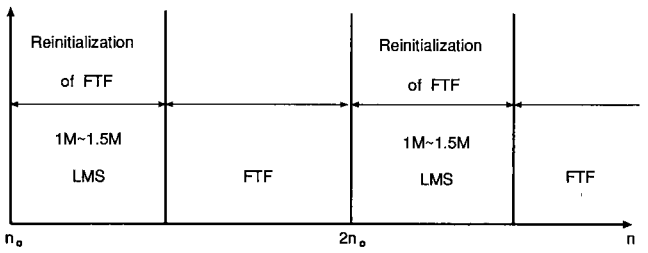


Fig. 4 Timing of operations in the periodic restarting method.

mance during this period. Since the variation of the unknown system is unpredictable and the reinitialization is periodical, the tracking speed can be affected to a large extent. However, this problem does not exist in the proposed method, since the NLMS algorithm is always implemented around the bottom of the error surface where it gives a good performance.

3.3 Boundary of Tracking Ability Between the NLMS and the RLS Algorithms

Experiments show that the NLMS algorithm can perform well when the parameters of the unknown system vary slowly with time. However, when the parameters of the unknown system vary fast and continuously, the limitation of the tracking ability of the NLMS algorithm becomes apparent. Eleftheriou et al. have given a boundary to evaluate the tracking ability between the RLS and the LMS [9]. When these two algorithms are implemented in a nonstationary environment, the total extra MSE introduced in the RLS algorithm can be calculated according to

$$\begin{aligned} J_{tot} &= J_{est} + J_{lag} \\ &= \frac{1-\lambda}{1+\lambda} M J_{min} + \frac{1}{2(1-\lambda)} tr(\mathbf{R})\sigma_w^2 \end{aligned} \tag{14}$$

where J_{est} is the extra MSE due to the weight noise and J_{lag} is the extra MSE due to the weight lag. σ_w^2 represents the variance of the nonstationarity.

The total extra MSE introduced in the LMS algorithm is[†]

$$J_{tot} = J_{est} + J_{lag} = \frac{\mu tr(\mathbf{R})}{2 - \mu tr(\mathbf{R})} J_{min} + \frac{M\sigma_w^2}{2\mu} \tag{15}$$

If the NLMS algorithm is used, Eq.(15) can be modified by replacing μ with $1/tr(\mathbf{R})$. The total extra MSE then becomes

$$J_{tot} = J_{min} + \frac{1}{2} M tr(\mathbf{R})\sigma_w^2 \tag{16}$$

The boundary condition defined in [9] can be written as

$$B_{lag} = \frac{J_{lag(LMS)}}{J_{lag(RLS)}} \tag{17}$$

Since the NLMS can provide the fastest convergence rate among the scalar step sized LMS, we have

$$B_{lag} = \frac{J_{lag(NLMS)}}{J_{lag(RLS)}} = (1-\lambda)M \tag{18}$$

Figure 5 (a) shows the boundary condition of Eq.(18). Apparently, when $B_{lag} \leq 1$, the NLMS algorithm gives no inferior tracking performance to that of the RLS algorithm.

However, in evaluating the tracking performance of an adaptive filter algorithm, one should consider not only the error caused by the weight lag, but also the error caused by the weight noise. If both of errors are considered, the boundary condition becomes

$$\begin{aligned} B_{tot} &= \frac{J_{tot(NLMS)}}{J_{tot(RLS)}} \\ &= \frac{J_{min} + \frac{1}{2} M tr(\mathbf{R})\sigma_w^2}{\frac{1-\lambda}{1+\lambda} M J_{min} + \frac{1}{2(1-\lambda)} tr(\mathbf{R})\sigma_w^2} \end{aligned} \tag{19}$$

Supposing $tr(\mathbf{R}) = M\sigma_u^2$, and deviding the denominator and the numerator in Eq.(19) by J_{min}, we have

$$B_{tot} = \frac{1 + \frac{1}{2} M^2 \rho \sigma_w^2}{\frac{1-\lambda}{1+\lambda} M + \frac{1}{2(1-\lambda)} M \rho \sigma_w^2} \tag{20}$$

where $\rho = SNR = \sigma_u^2 / J_{min}$ is the signal-to-noise ratio.

At first glance, the boundary of Eq.(20) is more stringent to the NLMS than that of Eq.(18), since if J_{lag} (NLMS) and J_{lag} (RLS) are equal, J_{est} of the NLMS is always greater than that of the RLS. However, it should be noted that J_{tot} of the RLS is λ dependent. As a result, we can show that under certain conditions, the tracking performance of the NLMS is better than that of the RLS. For example, if $\sigma_w^2 = 10^{-6}, M = 50$, and $\rho = 10^3$, the optimum forgetting factor calculated from Eq.(14) is $\lambda_{opt} = 0.968$. Substituting these values into Eq.(20), we can obtain the boundary of Eq.(20). The result is shown in Fig. 5 (b). From the figure, we see that the region of λ, in which the RLS has a smaller total extra MSE than that of the NLMS is quite limited. In practical situations, σ_w^2 and ρ are variable parameters. For example, if ρ is reduced to 10^2, the optimum forgetting factor λ_{opt} should be chosen to be 0.98 as shown in Fig. 5 (c). If λ keeps its previous optimum value of 0.968, the NLMS has almost the same total extra MSE as the RLS does. Furthermore, if the modified NLMS, which can further reduce the weight noise error, is used, the total extra MSE of the NLMS is always smaller than

[†]The original form used in [11] is

$$J_{tot} = \frac{\mu}{2} tr(\mathbf{R}) J_{min} + \frac{M\sigma_w^2}{2\mu}$$

This is to suppose $\mu << 1$ so that $2 - \mu tr(\mathbf{R}) \approx 2$. However, the more accurate form of weight noise error is defined by Eq.(3), which results in the present form.

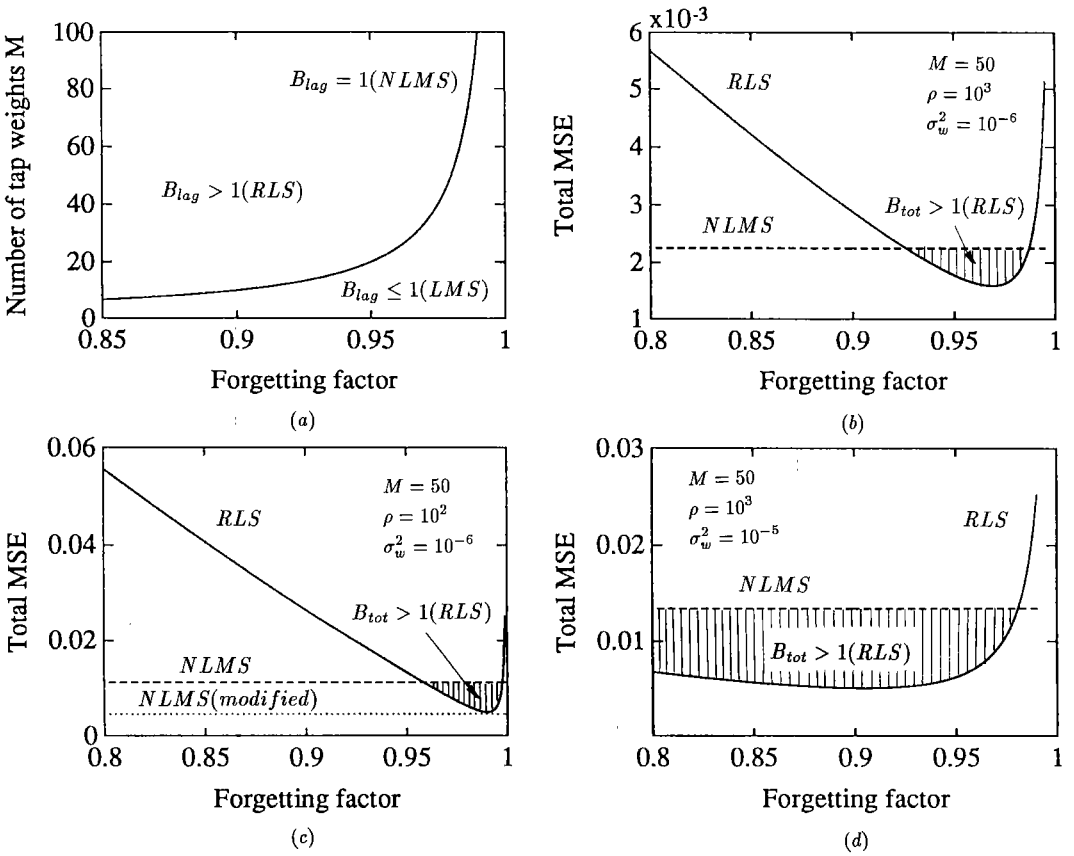


Fig. 5 Boundary conditions (a) based on the weight lag error, (b), (c) and (d) based on the total extra MSE.

that of the RLS, as shown by the dotted line in Fig. 5 (c). The above analysis has been verified by simulation and the simulation results will be presented in Sect. 4.

However, when the unknown system varies fast with time, the nonstationarity parameter σ_w^2 becomes large, which makes the lag error the dominant term in the total MSE and thus $B_{tot} \approx B_{lag}$. As can be seen in Fig. 4 (d), the tracking ability of the RLS is superior, especially when the number of tap weights increases and the eigenvalue of the input correlation matrix is widely spread. The proposed method periodically implements the FTF algorithm in such a case, which results in the tracking performance being almost the same as that of the RLS algorithm. We will demonstrate this by several simulations in Sect. 4.

4. Simulation

In this section, we will carry out some simulations on system identification to show the efficiency of the proposed method.

The simulations are implemented in a stationary and a nonstationary environment. In the stationary environment, we suppose that the unknown system is fixed, with some jumping parameters. In the nonstationary environment, we study the unknown system that has time varying parameters.

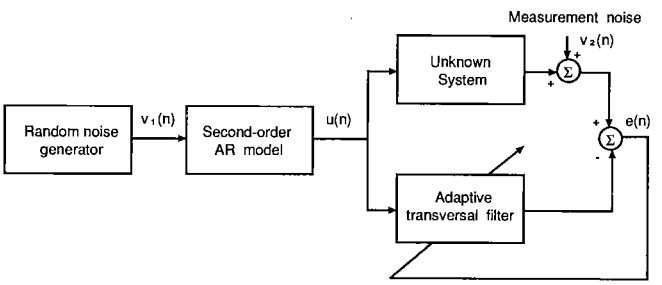


Fig. 6 Block diagram of system identification.

4.1 Description of the Simulation

The block diagram of system identification is shown in Fig. 6. The unknown system is supposed to be a second-order AR model with adjustable parameters. The transfer function of the unknown system can be written as

$$H(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{21}$$

where $b_0 = 1, a_1 = -2rcos(\theta), a_2 = r^2$. In the simulations, we suppose that $r = 0.85$ is fixed, and θ is

varying.

The zero-mean white noise $v_1(n)$ is put through another second-order AR model to produce colored tap input $u(n)$ with a variance of $\sigma_u^2 = 1$. The eigenvalue spread can be adjusted by changing the coefficients of the AR model. $v_2(n)$ represents the measurement noise with zero-mean and variance $\sigma_{v_2}^2 = 0.001$. Each experiment is repeated 100 times and each time an independent realization of the process $\{v_1(n)\}$ and $\{v_2(n)\}$ is used. The number of tap weights is set to 50 in all simulations. The computation precision is 32-bit floating-point arithmetic.

In the proposed method, switching from the NLMS to the FTF is determined by the threshold. Choice of the value of the threshold depends mainly on the application. In all the simulations that follow, we choose $L = 10$ and $K = 20$ in Eq.(12) and $J_{min} = \sigma_{v_2}^2 = 0.001$. So the threshold $\Theta = 0.2$. The implementation of the FTF algorithm is fixed to 3M iterations.

4.2 Simulation Results and Discussions

Simulation 1: Fixed Unknown System with Jumping Parameters

In this simulation, we study the convergence performance of the proposed algorithm and compare the result with those of the NLMS and the RLS algorithms.

The change of the phase of pole in the unknown system is shown in Fig. 7. In the first period($n \leqq n_j$), $\theta = \frac{\pi}{6}$ is used. θ is changed to $\frac{\pi}{4}$ in the second period($n > n_j$).

The learning curves obtained by three algorithms are shown in Fig. 8 (a). In Fig. 8 (b), we show the probability, which is the number of times the FTF is used in the 100 independent implementations. If the FTF is used in every implementation, the probability is 1. So Fig. 8 (b) clearly shows the interval of implementation of the two algorithms. In the proposed method, the FTF algorithm is implemented in the first 3M samples, and when the unknown system suddenly changes at a jumping point n_j, the algorithm is automatically switched from the NLMS to the FTF for another 3M implementations.

From Fig. 8, we can see that the performance of the NLMS is unsatisfactory, especially when the eigenvalues are widely disparate. By combining the NLMS and the FTF, the performance is greatly improved, which results in the convergence performance being the same as that of the RLS in the initialization period. In the reinitialization period, however, the convergence of the proposed method is faster than the RLS because fewer previous tap inputs are used. Furthermore, by using the modified step size of the NLMS, the proposed method can obtain a smaller misadjustment compared with the RLS.

Simulation 2: Slow Time-varying Unknown System

The purpose of this simulation is to compare the tracking ability of the proposed method with that of the RLS algorithm. The simulation is done under two situations. One corresponds to Fig. 5 (b) and (c). The effect of eigenvalue spread is considered in the other situation.

The change of the phase of pole in the unknown system is (see Fig. 7)

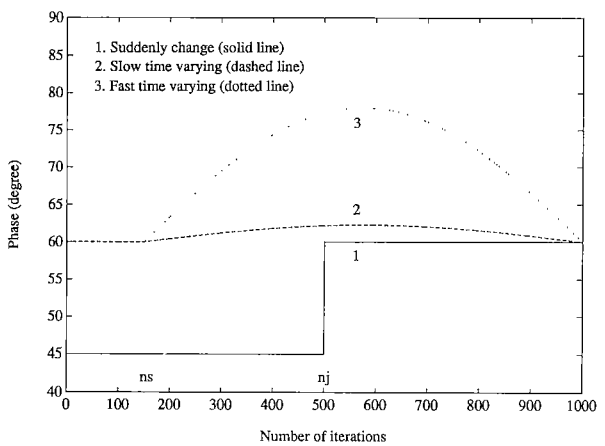


Fig. 7 Change of the phase of pole in the unknown system.

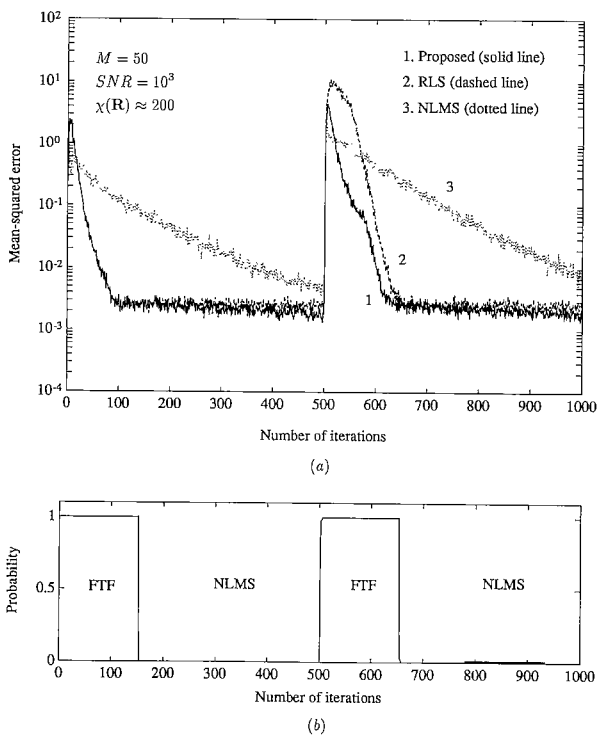


Fig. 8 Convergence performance for the unknown system with jumping parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.95$)+NLMS (modified step size with $a = 10$); RLS: $\delta = 5$, $\lambda = 0.95$; NLMS: $\tilde{\mu} = 1$; (a) Learning curves, (b) Probability of implementing the FTF.

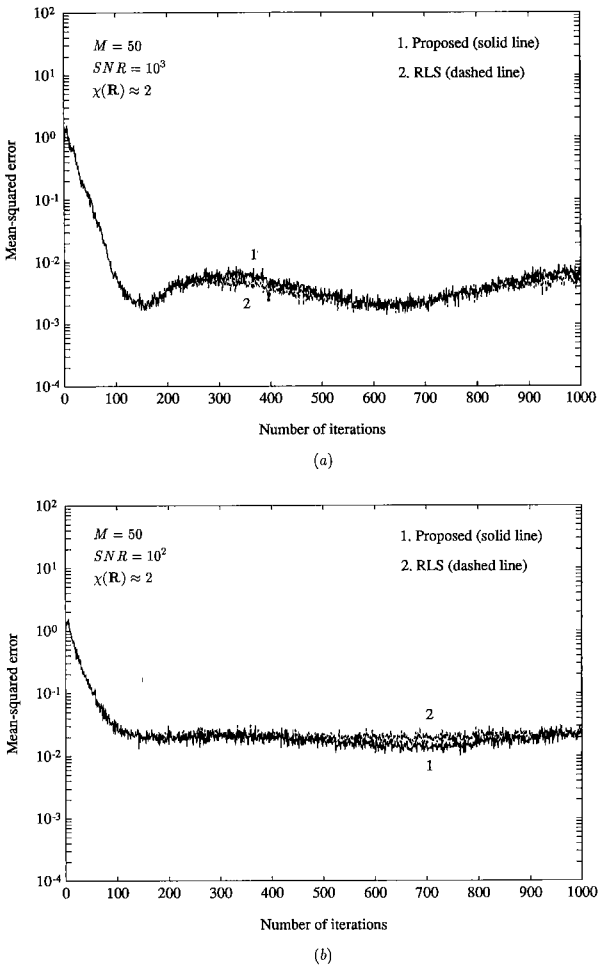


Fig. 9 Tracking performance for the unknown system with slow time variant parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.968$)+ NLMS (modified step size with variable a); RLS: $\delta = 5$, $\lambda = \lambda_{opt} = 0.968$. (a) The RLS is superior to the NLMS ($a = 1$), (b) The NLMS ($a = 2$) is superior to the RLS.

$$\theta(n) = \begin{cases} \frac{\pi}{4} & (n \leqq n_s) \\ \frac{\pi}{4} + \theta_s sin\left(\frac{(n-n_s)\pi}{N-n_s}\right) & (n > n_s) \end{cases} \tag{22}$$

where θ_s is a constant and its value represents the speed of variation of the unknown system. In this simulation, we choose $\theta_s = \pi/80$.

λ_{opt} used in the RLS can be obtained experimentally by adjusting λ and by making the sum of the total extra MSE, $\sum_{n=n_s}^{N} J_{tot}(n)$, a minimum, where n_s is assumed to be the measurement point at which the algorithm is converged and the unknown system begins to vary with time. We choose $n_s = 150$ and get $\lambda_{opt} = 0.968$ in this simulation.

The simulation results for the first situation are shown in Fig. 9. In order to confirm the boundary condition discussed above, we purposely let the FTF algorithm implements only in the first 3M iterations. The results are coincided with our analysis, and show that the NLMS algorithm has a superior tracking performance under certain conditions.

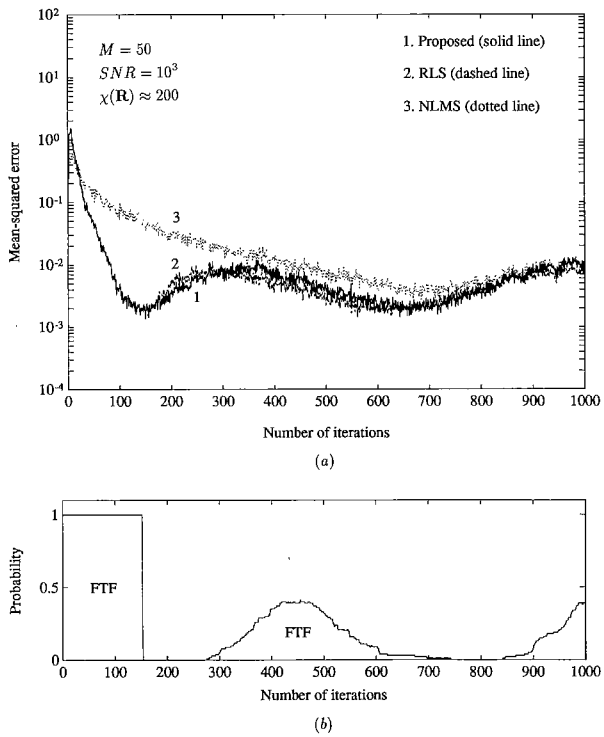


Fig. 10 Tracking performance for the unknown system with slow time variant parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.968$)+ NLMS (modified step size with $a = 2$); RLS: $\delta = 5$, $\lambda = \lambda_{opt} = 0.968$; NLMS: $\tilde{\mu} = 1$; (a) Learning curves, (b) Probability of implementing the FTF.

The simulation results for the second situation are shown in Fig. 10. The proposed method implements the FTF in the first 3M iterations. In the remaining part, the NLMS is dominantly used.

From the results, we can see that when the unknown system is varying with time slowly, the NLMS algorithm gives the tracking performance similar to that of the RLS algorithm.

Simulation 3: Fast Time-varying Unknown System

This simulation is used to demonstrate the superiority of the tracking ability of the proposed method over that of the NLMS, when the unknown system varies fast, especially under the condition of a wide eigenvalue spread.

In this simulation, we choose $\theta_s = \pi/10$ ($\lambda_{opt} = 0.9$). This further increases the variation of the unknown system (see Fig. 7).

The simulation results are shown in Fig. 11. When the speed of variation of the unknown system is increased, the proposed method automatically switches from the NLMS to the FTF more frequently, giving

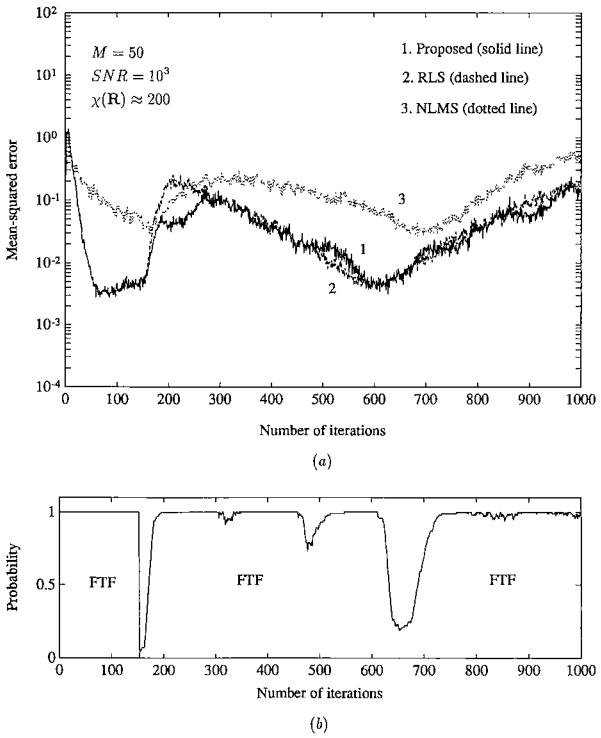


Fig. 11 Tracking performance for the unknown system with fast time variant parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.9$)+ NLMS (modified step size with $\alpha = 2$); RLS: $\delta = 5$, $\lambda = \lambda_{opt} = 0.9$; NLMS: $\tilde{\mu} = 1$. (a) Learning curves, (b) Probability of implementing the FTF.

rise to a tracking performance which is almost the same as that of the RLS, but is far better than that of the NLMS. The discontinuities caused by periodically implementing the FTF is unnoticeable.

4.3 Summary of Performance Comparisons

We summarize the performance of the proposed algorithm in comparison with other adaptive filter algorithms in Table 1. As shown in this table, the proposed method demonstrates many of the highly desirable features of an adaptive filter algorithm, including fast convergence, fast tracking, small misadjustment, computational simplicity and numerical stability.

5. Conclusion

The new combined fast adaptive filter algorithm and its automatic switching method have been proposed. From Table 1, we can conclude that a performance of the RLS algorithm and a computational cost of the LMS algorithm can be achieved. Although the experiments described in this paper belong to the field of system identification, the proposed algorithm is expected to be applicable to other fields as well.

Table 1 Summary.

Properties	NLMS	RLS		Fast RLS		Proposed
		λ=1	λ<1	λ=1	λ<1	
Convergence rate	×	○	○	○	○	○
Misadjustment	×	○	△	○	△	○
Tracking	□	×	○	×	○	○
Computational load	○	×	×	○	○	○
Numerical stability	○	○	○	×	×	○

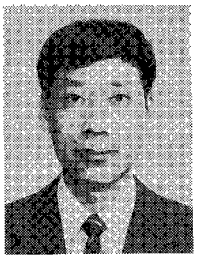
○ : Good × : Bad △ : Depend on λ □ : Dapend on application

Acknowledgements

The authors wish to thank Dr. Z. Ma, Mr. H. Katayama, Dr. J-H. Zhou and students in our laboratory for their stimulating discussions and kind help.

References

[1] Haykin,S., *Adaptive Filter Theory*, Second edition, Printice-Hall, 1990.
[2] Harris, R.W., et al., "A variable step (VS) adaptive filter algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-34, no.2, pp.309-316, Apr. 1986.
[3] Sugiyama, A., et al., "A fast covergence algorithm for adaptive FIR filters," *Proc. ICASSP*, Glasgow, Scotland, pp.892-895, 1989.
[4] Boray, G.K., and Srinath, M.D., "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuit and System*, vol.39, No.1, pp.1-10, Jan. 1992.
[5] Cioffi, J.M., and Kailath, T., "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-32, no.2, pp.304-337, Apr. 1984.
[6] Slock, D.T.M., and Kailath, T., "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol.39, pp.92-114, Jan. 1991.
[7] Kubo, I., and Nakayama, K., "Comparisons among fast adaptive filter algorithms based on tracking property for time-varying systems," *Proc. 6th Digital Signal Processing Symposium*, vol.A6-2, pp.162-167, Nov. 1991.
[8] Bershad, N., and Macchi, O., "Comparison of RLS and LMS algorithm for tracking a chirped signal," *Proc. ICASSP*, Glasgow, Scotland, pp.896-899, 1989.
[9] Eleftherion, E., and Falconer, D.D., "Tracking properties and steady state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-34, no.5, pp.1097-1110, Oct. 1986.
[10] Wang, Y., and Nakayama, K., "An automatic switching method used for a combined adaptive filter algorithm," *Proc. 1993 Joint Technical Conference on Circuits / Systems, Computers and Communications*, Nara, Japan, vol.1, pp.426-431, Jul. 1993.
[11] Widrow, B., et al., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. of the IEEE*, vol.64, no.8, pp.1151-1162, Aug. 1976.



Youhua Wang received the M.S. degree in 1988 from Shanghai University of Technology, Shanghai, China. From 1988 to 1990, he was a Research Assistant at the Advanced Technology Research Center, Shenzhen University, Shenzhen, China. He is presently a graduate student doing the doctorate degree at the Graduate School of Natural Sci. & Tech., Kanazawa University, Kanazawa, Japan. His current research interests include digital signal processing, adaptive filtering, and their application to communications.

Kenji Nakayama received the B.E. and Dr. degrees in electronics engineering from Tokyo Institute of Technology (TIT), Tokyo, Japan, in 1971 and 1983, respectively. From 1971 to 1972 he was engaged in the research on classical network theory in TIT. He was involved in NEC Corporation from 1972 to 1988. He joined the Department of Electrical and Computer Engineering at Kanazawa University, in Aug. 1988, where he is currently a Professor. His current research interests include adaptive signal processing and artificial neural networks.