An automatic switching method used for a combined adaptive filter algorithm

An Automatic Switching Method Used for A Com[illegible]ed Adaptive Filter Algorithm

Youhua WANG Kenji NAKAYAMA

Graduate School of Natural Sci. & Tech., Kanazawa Univ.

2-40-20 Kodatsuno, Kanazawa 920, Japan

E-mail: nakayama@haspnn1.ec.t.kanazawa-u.ac.jp

Abstract

An automatic switching method, used for the combined fast adaptive filter algorithm consisting of the FTF and the normalized LMS algorithms, is proposed in this paper. Switching between the FTF and the NLMS algorithms is controlled by the difference of the MSE sequence, which corresponds to the slope of the MSE or the speed of variation of the unknown system. If the difference exceeds a threshold, then the FTF algorithm is selected, and vice versa. Simulation results, in a stationary as well as a nonstationary environment, show that the combined algorithm with the proposed switching method can provide less computations compared with the RLS algorithm, while maintaining the same performance as that of the RLS algorithm. Furthermore, compared with the FTF algorithm, it can achieve numerically stable operation.

1 Introduction

The FTF algorithm, which has been introduced to reduce the computational load of the RLS algorithm, presents many desirable features such as fast convergence rate and fast tracking. These features, however, can be preserved only when the implementation of the FTF algorithm is numerically stable. Unfortunately, the stable implementation of the FTF can not last for a long time without rescue, especially in a nonstationary environment where the forgetting factor should be less than one in order to obtain tracking capability [1]. On the other hand, the widely used NLMS algorithm, though has a slow convergence rate, is simple and numerically stable. Apparently, the FTF and the NLMS algorithm have some complementary characteristics. So if these two algorithms are combined, several desirable features of both algorithms can be preserved, while drawbacks can be overcome.

We have proposed a combined fast adaptive filter algorithm [2]. The main idea behind it is that whenever the tap weights have a large deviation from their optimum values, a large error results, the FTF algorithm is used to quickly turn the tap weights back to close proximity of their optimum values. Since the FTF can provide the least square solution like the RLS algorithm, usually only 2M - 3M iterations can make the tap weights close enough to the optimum values . After that, the NLMS algorithm is used. Thus, fast convergence rate can be obtained while numerical instability is avoided. When the unknown system is slowly varying with time, we usually need not use the FTF algorithm, since the NLMS algorithm is capable of tracking the slow time varying system well [2]-[4]. When the unknown system is varying with time fast, the fast tracking can be obtained by periodically implementing the FTF algorithm. The improved performance obtained by using the proposed algorithm was shown in [2].

The remaining problem is how to automatically switch between both the NLMS and the FTF algorithms. An automatic switching method is proposed in this paper. We first briefly describe the proposed combined adaptive algorithm. Then, we introduce the automatic switching method. Finally, the proposed method is verified by computer simulations. Various situations in a stationary and a nonstationary environments are taken into account in order to demonstrate the efficiency of the proposed method.

2 Combined Adaptive Filter Algorithm

Figure 1 shows the timing of operations by using the proposed algorithm. We may consider an adaptive process as being a dynamic electric system, which consists of a transient period and a steady-state period. 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 controlled by using the

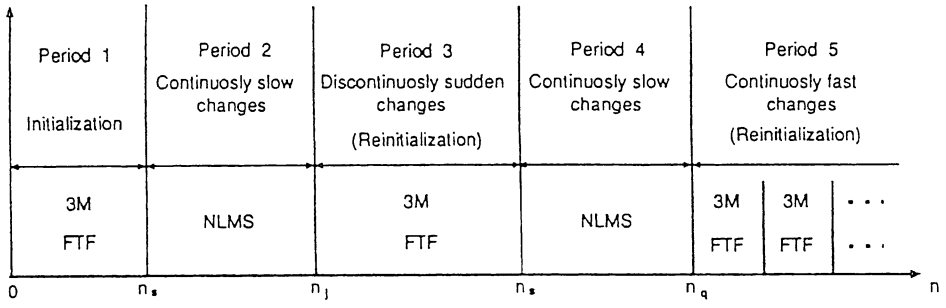

Figure 1: Timing of operations by using proposed method

difference of the MSE sequence. If the difference exceeds a threshold, then the FTF algorithm is used, otherwise, the NLMS algorithm is used. The switching method will be discussed in the next section.

In order to obtain fast tracking when the unknown system varies fast with time and avoid the possible instability produced by the FTF algorithm, the reinitialization of the FTF algorithm is periodically implemented in period 5. We may think that the discontinuities or bias caused by the reinitialization will produce unbearable large MSE. This can be true if we choose the reinitialization parameters δ and λ improperly. There are two reasons that can cause discontinuities. Firstly, the transient produced by removing the augmentation of series zero tap inputs before reinitialization. This make 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. Secondly, the near singular of the input correlation matrix, which can happen 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 large δ can be suppressed by reducing λ. Generally speaking, by suitably choosing δ and λ, we can obtain a tracking performance like that of the RLS algorithm without introducing noticeable discontinuities.

Figure 2 shows the flow chart of the combination of the two algorithms. We can see that the difference between the FTF and the NLMS algorithm 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 relationships 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.

There are several important features of the proposed

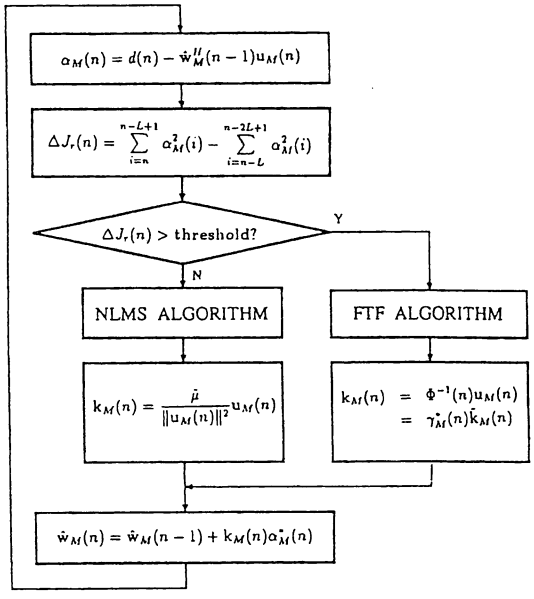

Figure 2: Flow chart of proposed method

algorithm. 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, several 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 transient, and a small step size used in the NLMS to provide small misadjustment in steady state operation.

3 Automatic Switching Method

In the combined adaptive algorithm, switching between the two algorithms is controlled by the difference of the MSE sequence, which is used for detecting the slope of the MSE or the speed of variation of the unknown system. In order to calculate the difference of the MSE, we first write the error produced between the desired and the adaptive filter output [5]

$$\begin{aligned}\alpha(n) &= d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \\ &= e_o(n) + \mathbf{w}_o^H\mathbf{u}(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \\ &= e_o(n) + \alpha_{ex}(n) \end{aligned} \quad (1)$$

where $\mathbf{w}_o$ is the tap-weight vector of the unknown system, $e_o(n)$ is the error produced by the measurement noise, and $\alpha_{ex}(n)$ is the excess error produced by both weight noise and weight lag.

Suppose all the data are real, the squared error of $\alpha(n)$ is

$$\begin{aligned}\alpha^2(n) &= (e_o(n) + \alpha_{ex}(n))^2 \\ &= e_o^2(n) + 2\alpha_{ex}(n)e_o(n) + \alpha_{ex}^2(n) \end{aligned} \quad (2)$$

The MSE of $\alpha_{ex}(n)$ can be obtained by taking the average of both side of Eq.(2)

$$\begin{aligned}\overline{\alpha^2(n)} &= \overline{e_o^2(n)} + 2\overline{\alpha_{ex}(n)e_o(n)} + \overline{\alpha_{ex}^2(n)} \\ &\approx \sigma^2 + \overline{\alpha_{ex}^2(n)} \end{aligned} \quad (3)$$

where we suppose that the mean value of $e_o(n)$ is zero, and also it is independent of $\overline{\alpha_{ex}(n)}$ so that $2\overline{\alpha_{ex}(n)e_o(n)} \approx 0$. σ^2 is the variance of the measurement noise.

The difference of the MSE can be written as

$$\begin{aligned}&\overline{\Delta\alpha^2(n)} = \overline{\alpha^2(n+\tau)} - \overline{\alpha^2(n)} \\ &= \left(\overline{e_o^2(n+\tau)} - \overline{e_o^2(n)}\right) + \left(\overline{\alpha_{ex}^2(n+\tau)} - \overline{\alpha_{ex}^2(n)}\right) \\ &\approx \left(\overline{\alpha_{ex}^2(n+\tau)} - \overline{\alpha_{ex}^2(n)}\right) = \overline{\Delta\alpha_{ex}^2(n)} \end{aligned} \quad (4)$$

where τ is a time delay constant. The following relative difference of the MSE is used to detect the timing of switching the algorithms.

$$\overline{\Delta\alpha_r^2(n)} = \frac{\overline{\Delta\alpha^2(n)}}{\sigma^2} \quad (5)$$

Experiments show that using Eq.(5) instead of Eq.(4) is more effective. This is because the MSE caused by both weight noise and weight lag can be considered as a misadjustment from σ^2. For example, suppose $\overline{\Delta\alpha^2(n)} = 0.05$. If $\sigma^2 = 0.001$, the misadjustment is 500%. However, if $\sigma^2 = 0.01$, the misadjustment is only 50%. In the previous case, the FTF algorithm should be used, in order to obtain fast tracking. In the latter case, however, the NLMS algorithm should be used, in order to obtain a small final misadjustment. On the other hand, we know that the performance of the FTF algorithm is closely related to σ^2. If $\sigma^2 = 0$, the FTF achieves its best performance. When σ^2 increases, however, the superior convergence rate and tracking speed of the FTF algorithm over the NLMS algorithm becomes lost [2][5]. So when the combined algorithm is implemented in a nonstationary environment, the FTF algorithm should be used less frequently for a large σ^2. Apparently, Eq.(5) satisfies these requirements.

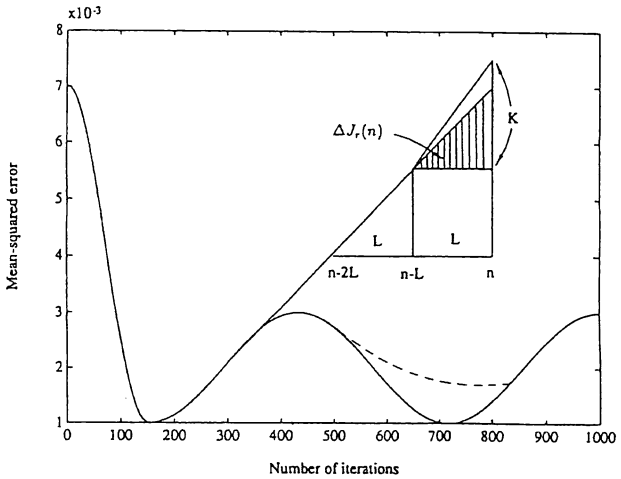

Figure 3: Calculating difference of MSE

In summary, as shown in Figure 3, we can calculate the difference of the MSE as

$$\Delta J_r(n) = \frac{|\Delta J(n)|}{\hat{J}_{min}} = \frac{1}{2}\left(\frac{|J_1(n) - J_2(n)|}{\hat{J}_{min}}\right) \quad (6)$$

$$J_1(n) = \sum_{i=n-L+1}^{n} \alpha^2(i) \quad (7)$$

$$J_2(n) = \sum_{i=n-2L+1}^{n-L} \alpha^2(i) \quad (8)$$

where $\hat{J}_{min}$ is the estimate of the minimum MSE, and L is the time delay constant as well as the number of samples used for averaging the MSE. We note that $\Delta J_r(n)$ obtained from Eq.(6) is always positive. This is because we want to detect the slope of the MSE, whenever the MSE increases or decreases. Otherwise, the tracking performance may be degraded as shown by dashed line in Figure 3, since $\Delta J_r(n)$ becomes minus when the MSE decreases, it will never exceed a threshold so that the NLMS is used.

The threshold Θ defined in Figure 3 is

$$\Theta = \frac{1}{2}KL \quad (9)$$

where K is a positive constant, which represents the slope of the MSE or the speed of variation of the unknown system.

In practical situation, Eqs.(6) and (9) can be rewritten in another equivalent simple forms

$$\Delta J_r(n) = |J_1(n) - J_2(n)| \quad (10)$$

$$\Theta = K\hat{J}_{min}L = K'L \quad (11)$$

We note from Eq.(11) that if $\hat{J}_{min}$ does not change with time, the threshold of Eq.(11) becomes a constant. The value of the threshold can be chosen experimentally.

In some applications, $\hat{J}_{min}$ may vary with time, and the threshold of Eq.(11) also varies with time. In this case, $\hat{J}_{min}$ can be replaced by $J(n) = min\{J_1(n), J_2(n)\}$, so that Eq.(11) becomes

$$\Theta = KJ(n)L \tag{12}$$

4 Simulation and Discussions

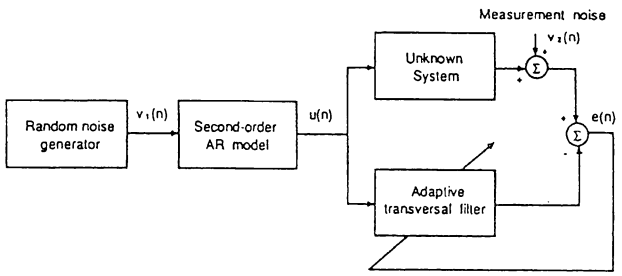

Figure 4: Block diagram of system identification

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

The simulations are implemented in a stationary and a nonstationary environment [6]. In the stationary environment, we suppose the unknown system is fixed, with some jumping parameters. In the nonstationary environment, we study the unknown system that has the time varying parameters.

4.1 Description of Simulation

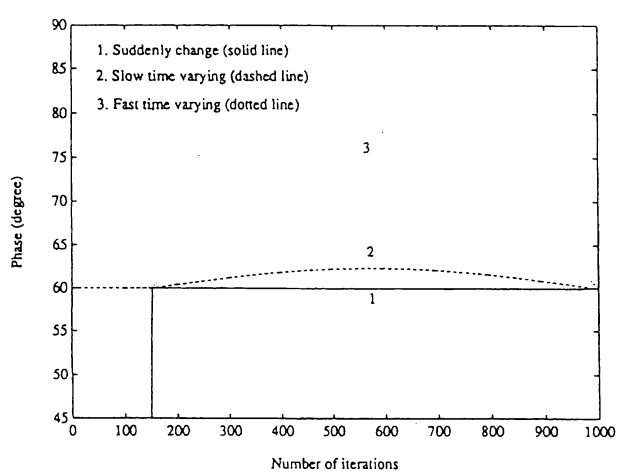

Figure 5: Change of phase of pole in unknown system

The block diagram of system identification is shown in Figure 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

$$H(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{13}$$

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 variable.

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 is adjusted to about 200 in the simulations. $\{v_2(n)\}$ represents the measurement noise with zero-mean and variance $\sigma_{v_2}^2 = 0.001$. Each experiment is repeated 100 times, each time using an independent realization of the process $\{v_1(n)\}$ and $\{v_2(n)\}$. The number of tap weights is set to 50 in all simulations. The computation precision is 32-bit floating-point arithmetic.

In the combined algorithm, switching from the NLMS to the FTF is determined by a threshold. Choosing the value of the threshold mainly depends on practical applications. In every simulation that follows, we choose $L = 10$ and $K' = 0.02$ in Eq.(11) so that the threshold $\Theta = 0.2$. The implementation of the FTF algorithm is fixed to 3M iterations. The step size used in the NLMS algorithm is $\tilde{\mu}/\|\mathbf{u}(n)\|^2$.

4.2 Simulation Results

Simulation 1: Fixed Unknown System with Jumping Parameters

In this simulation, we study the convergence performance of the combined 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 Figure 7. In the first period($n \leq n_j$), $\theta = \frac{\pi}{4}$ is used. θ is changed to $\frac{\pi}{3}$ in the second period($n > n_j$).

The learning curves obtained by three algorithms are shown in Figure 8(a). In Figure 8(b), we also show the probability. This means how many times FTF is used among 100 independent implementation. If FTF is used in every implementation, its probability will be 1. So Figure 8(b) clearly shows the interval of implementation of two algorithms. By using the proposed switching 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 NLMS to FTF for another 3M implementation.

From Figure 8, we can see that the convergence performance of NLMS is unsatisfactory, especially when the eigenvalues are widely disparate. By combining NLMS and FTF, performance is greatly improved, resulting in the convergence rate the same as RLS in initialization period. In reinitialization period, however, the convergence of the combined algorithm is faster than RLS, because fewer previous tap inputs are used. On the other hand, by using a small step size of NLMS, the combined algorithm can obtain smaller misadjustment compared with RLS.

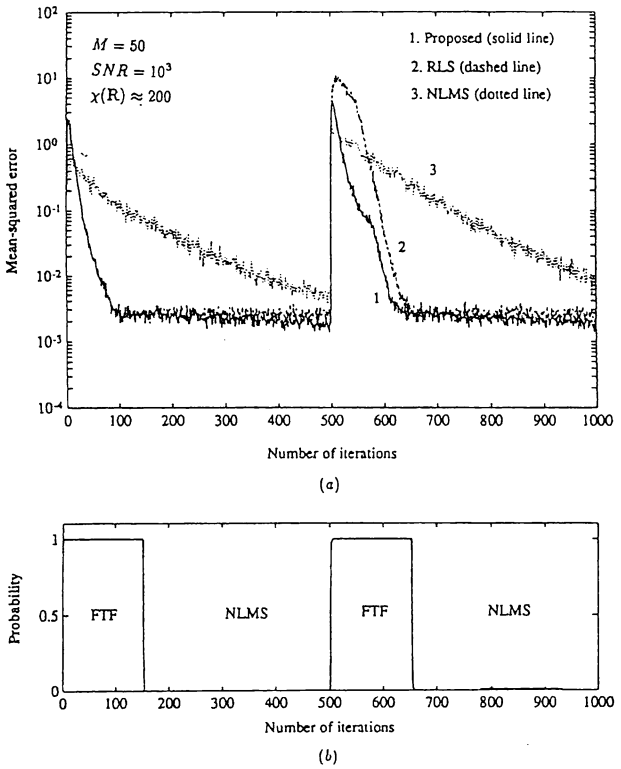

Figure 6: Convergence performance for unknown system with jumping parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.95$)+NLMS($\tilde{\mu} = 0.2$); RLS: $\delta = 5$, $\lambda = 0.95$; NLMS: $\tilde{\mu} = 1$; (a) Learning curves, (b) Probability of implementing FTF

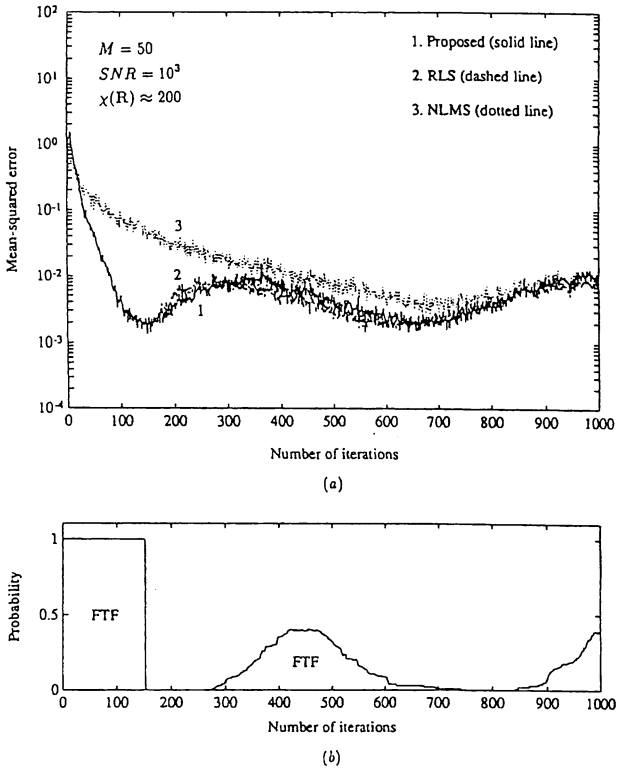

Figure 7: Tracking performance for unknown system with slow time variant parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.968$)+ NLMS($\tilde{\mu} = 0.2$); RLS: $\delta = 5$, $\lambda = \lambda_{opt} = 0.968$; NLMS: $\tilde{\mu} = 1$; (a) Learning curves, (b) Probability of implementing FTF

Simulation 2: Slow Time Varying Unknown System

The purpose of this simulation is to compare the tracking ability of the combined algorithm with that of the RLS algorithm.

The change of the phase of pole in the unknown system is (see also Figure 7)

$$\theta(n) = \begin{cases} \frac{\pi}{3} & (n \le n_s) \\ \frac{\pi}{3} + \theta_s in\left(\frac{(n-n_s)\pi}{N-n_s}\right) & (n > n_s) \end{cases} \qquad (14)$$

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

λ_{opt} used in RLS can be obtained experimentally by adjusting λ and by making the sum of the total extra MSE $\sum_{n=n_s}^{N} J_{tot}(n)$ to its minimum [2], where n_s is supposed 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 are shown in Figure 9. The proposed method implements FTF in the first 3M iterations. In the remaining part, NLMS is dominantly used. From the result, we can see that when the unknown system is varying with time slowly, the NLMS can provide the tracking performance similar to that of the RLS algorithm, even under the condition of wide spread of eigenvalues.

Simulation 3: Fast Time Varying Unknown System

This simulation is used to illustrate the superiority of the tracking ability by using the combined algorithm over NLMS, when the unknown system varies fast with time.

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

The simulation results are shown in Figure 10. When the speed of variation of the unknown system is increased, the proposed method automatically switches from NLMS to FTF more frequently, which results in the tracking performance almost the same as that of the RLS, but considerably better than that of NLMS. The discontinuities caused by periodically implementing FTF is unnoticeable.

5 Conclusion

We summarize the performance of the proposed combined algorithm with automatic switching method compared with other adaptive filter algorithms in Table 4. From the table, we can conclude that by combining the FTF and the NLMS algorithms, a performance similar to

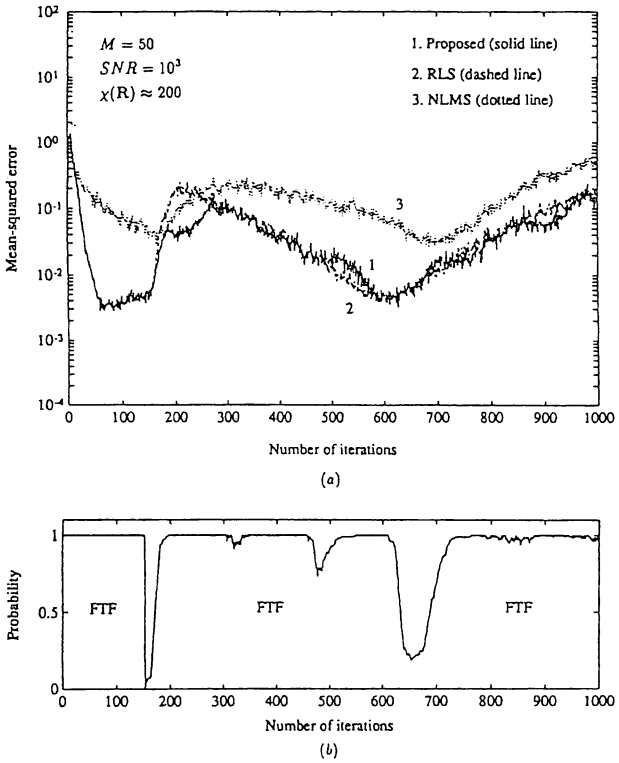

Figure 8: Tracking performance for unknown system with fast time variant parameters. Proposed: 7M FTF($\delta = 5$, $\lambda = 0.9$)+ NLMS($\tilde{\mu} = 0.2$); RLS: $\delta = 5$, $\lambda = \lambda_{opt} = 0.9$; NLMS: $\tilde{\mu} = 1$; (a) Learning curves, (b) Probability of implementing FTF

that of the RLS algorithm and a computational cost comparable to that of the LMS algorithm are obtained. Thus, the proposed method demonstrates several desirable features including fast convergence, fast tracking, small misadjustment, computational simplicity and numerical stability. Although the experiments shown in this paper are only involved in the field of system identification, the proposed algorithm is 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 would like to thank Mrs. Ma, Mr. Katayama and students in our laboratory for their stimulating discussions and kind help.

References

[1] J.M.Cioffi and T.Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-32, pp.304-337, Apr. 1984.

[2] Y.Wang and K.Nakayama,"A fast adaptive filter algorithm combining fast RLS and normalized LMS," Technical report of IEICE, DSP 92-94, pp.71-78, (1993-01).

[3] E.Eleftherion and D.D.Falconer,"Tracking properties and steady state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-34, pp.1097-1110, Oct. 1986.

[4] N.Bershad and O.Macchi,"Comparison of RLS and LMS algorithm for tracking a chirped signal," *Proc. ICASSP*, Glasgow, Scotland, pp.896-899, 1989.

[5] S.Haykin, *Adaptive Filter Theory*, Second edition, Printice-Hall, 1990.

[6] I.Kubo and K.Nakayama. "Comparisons among fast adaptive filter algorithms based on tracking property for time-varying systems(in Japanese)," *Proc. 6th Digital Signal Processing Simposium*, vol.A6-2, pp.162-167, Nov. 1991.