PAPER Special Section on Digital Signal Processing

Fast Implementation Technique for Improving Throughput of RLS Adaptive Filters

Kiyoshi NISHIKAWA† *and* **Hitoshi KIYA**†, *Regular Members*

SUMMARY This paper proposes a fast implementation technique for RLS adaptive filters. The technique has an adjustable parameter to trade the throughput and the rate of convergence of the filter according to the applications. The conventional methods for improving the throughput do not have this kind of adjustability so that the proposed technique will expand the area of applications for the RLS algorithm. We show that the improvement of the throughput can be easily achieved by rearranging the formula of the RLS algorithm and that there are no need for faster PEs for the improvement.

key words: *adaptive system, pipeline processing, digital signal processor, RLS adaptive filters*

1. Introduction

In this paper we propose a fast implementation technique of the recursive least squares (RLS) adaptive digital filters (ADFs) [1], [2]. This technique has an adjustable parameter so that we can trade the throughput of the filter (the total number of its output samples over a unit time) and the rate of convergence.

As a method to improve the throughput of an ADF, the pipeline technique is well known and widely used. Many architectures for the pipelined implementation of the least mean square (LMS) ADFs have been proposed [3]–[10] and using these architectures, LMS ADFs can be operated at desired throughput levels. The rates of convergence of them however are smaller than those of RLS ADFs, whose throughput rate is lower because of their computational requirements. Faster rate of convergence of RLS ADFs makes them attractive for some applications so that the improvement of throughput of them is desired.

For pipeline implementation of the RLS ADFs the systolic-array (SA) structures has been considered as a promised solution [1], [11], [12]. The SA structure is suitable for implement in VLSI because of its regularity [1]: An SA is made up of cells with the same structure and signals are passed through from one cell to the next. The SA structure improves the throughput of RLS ADFs, but it lacks flexibility and requires us to prepare all the processing elements (PEs) (e.g., multipliers). Once it is implemented in VLSI, parameters such as filter length cannot be adjusted. If we want to improve the throughput further, we must use faster PEs which will cost more.

As an alternative method, the pipelined Kalman (PIPKAL) algorithms has been proposed [13]. This algorithm does not require specially designed cells and can be implemented by adding delay elements to the standard RLS ADF. It is more flexible than the SA structure and allows us to reuse PEs, which reduces their numbers. This feature is suitable to implement it using programmable devices such as FPGA or digital signal processors (DSPs). Even if we use PIPKAL, however, we must use faster PEs to obtain further improvements in throughput. Moreover, there is computational redundancy in the PIPKAL formula [14].

In this paper, we propose a novel technique for implementing RLS ADFs. Using the technique we can trade the throughput and the rate of convergence of the filter according to the number of available PEs and/or the desired throughput rate. The proposed technique rearranges the RLS algorithm formula to produce several output samples during one update of the filter coefficients. Throughput can thus be improved without the need for faster PEs. Proposed technique is more effective when we are using several PEs simultaneously and their number is limited or when the ADFs are implemented on DSPs.

This paper is organized as follows. In Sect. 2, we summarize the notation and adaptive model used in this paper, and a review of the RLS algorithm is given. The proposed technique is described in Sect. 3. Consideration of the proposed technique through computer simulation is given in Sect. 4.

2. Standard RLS Algorithm

First, let us review the standard RLS algorithm [15]. We also summarize the notations and give the adaptive model used in this paper.

2.1 Notations and Adaptive Model

In the following, $\boldsymbol{w}(n)$ and $\boldsymbol{u}(n)$ denote the coefficient vector of the ADF and its tap-input vector respectively, and they are defined as

$$\boldsymbol{w}(n) = [w_0(n) \;\; w_1(n) \;\; \ldots \;\; w_{N-1}(n)]^T \quad (1)$$

$$\boldsymbol{u}(n) = [u(n) \;\; u(n-1) \;\; \ldots u(n-N+1)]^T \quad (2)$$

Manuscript received December 1, 1999.
Manuscript revised February 24, 2000.

†The authors are with the Department of Electrical Engineering, Tokyo Metropolitan University, Hachioji-shi, 192–0397 Japan.

where N indicates the length of the ADF and ^{T} shows the transpose operation. The desired signal is denoted by $d(n)$.

We assume that there is a true weight $\boldsymbol{w}_{opt}$ which satisfies the equation

$$d(n) = \boldsymbol{u}^T(n)\boldsymbol{w}_{opt} + \eta(n) \tag{3}$$

where $\eta(n)$ shows the output noise [16]. In the following consideration, we assume $\boldsymbol{w}_{opt}$ is a time-invariant system. We also assume all the signals are real for simplifying the notations.

2.2 Formula

The formula of the RLS algorithm is given as

$$\boldsymbol{k}(n) = \frac{\lambda^{-1}P(n-1)\boldsymbol{u}(n)}{1+\lambda^{-1}\boldsymbol{u}^T(n)P(n-1)\boldsymbol{u}(n)} \tag{4}$$

$$y(n) = \boldsymbol{w}^T(n-1)\boldsymbol{u}(n) \tag{5}$$

$$\xi(n) = d(n) - y(n) \tag{6}$$

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n) \tag{7}$$

$$P(n) = \lambda^{-1}\{P(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^T(n)P(n-1)\} \tag{8}$$

where the following variables are used:

λ	$\cdots$	Forgetting factor ($0 < \lambda \leq 1$)
$\boldsymbol{k}(n)$	$\cdots$	Kalman gain vector
$P(n)$	$\cdots$	Inverse of autocorrelation matrix of $\boldsymbol{u}(n)$

and the superscript $^{-1}$ shows the inverse of a matrix or a vector. A configuration of the RLS algorithm is shown in Fig. 1.

It is known that the standard RLS algorithm requires $2N^2 + 6N$ multiplications [2], [17] and an ADF must perform these calculations at each time n to produce an output sample $y(n)$. This means that if we want double the clock rate of $u(n)$ then each calculations should be performed at twice as faster speed. Note that the symmetric property of $P(n)$ is counted to evaluate the required calculations.

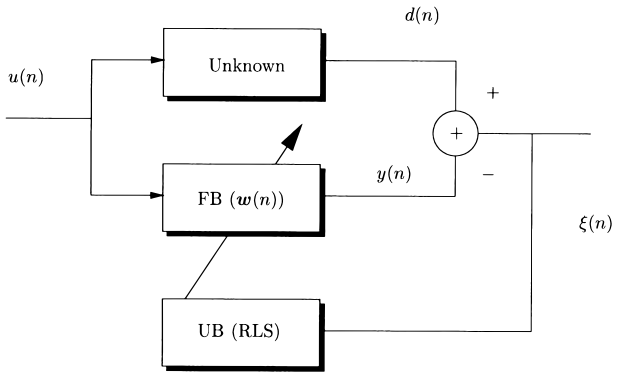


Fig. 1 Configuration of ADF based on RLS algorithm. It can be divided into two blocks: filtering block (FB) for calculating $y(n)$ and update block (UB) for updating $\boldsymbol{w}(n)$ according to $u(n)$ and $\xi(n)$.

3. Proposed Technique

Here, we describe the proposed implementation technique. We first show the basic idea of the proposed technique and then we consider an implementation method for the idea.

3.1 Basic Idea

First, we describe the basic idea of the proposed technique. From Fig. 1, we notice that the configuration of the RLS algorithm can be divided into two blocks: Namely (i) the filtering block (FB) and (ii) filter update block (UB). We can see from the figure that these two blocks can perform independently.

It is obvious from this figure that only the FB is required to produce the output signal $y(n)$ of the filter and the throughput of the filter is mainly determined by this block regardless of the accuracy of the adaptation. On the other hand, the UB contributes to the rate of convergence.

In the conventional configurations of RLS filters, these two blocks perform their works dependently. In other words, FB produces the output sample $y(n)$ only after UB updates filter coefficients $\boldsymbol{w}(n)$. Therefore, the throughput of the filter depends on the number of calculations in both FB and UB.

This fact shows that if FB produces several output samples $y(n)$ during one update interval we can improve the throughput of the filter without the need for faster PEs. Note that this idea may slightly degrade the rate of convergence.

We conclude that we can improve the throughput of the RLS ADFs by independently drive the two blocks if we accept a slight degrade of the rate of convergence.

3.2 Implementation Procedure

Based on the above idea, we propose the following procedure to implement the RLS ADFs:

1. Separate the FB and UB calculations of the RLS ADFs:
   - Calculation of FB

$$y(n) = \boldsymbol{u}^T(n)\boldsymbol{w}(n-1) \tag{9}$$

   - Calculation of UB

$$\boldsymbol{\Phi}(n) = P(n-1)\boldsymbol{u}(n) \tag{10}$$

$$\boldsymbol{k}(n) = \frac{1}{\lambda + \boldsymbol{u}^T(n)\boldsymbol{\Phi}(n)}\boldsymbol{\Phi}(n) \tag{11}$$

$$\xi(n) = d(n) - y(n) \tag{12}$$

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n) \tag{13}$$

$$P(n) = \lambda^{-1}\{P(n-1) - \boldsymbol{k}(n)\boldsymbol{\Phi}(n)\} \tag{14}$$

Note that the standard RLS ADFs calculate all of these equations at each time n.

2. Determine the ratio L of improvement in the throughput. Note that L may be determined according to the acceptable amount of calculations per each time n and/or the desired throughput rate.
3. Insert the calculations of FB into those of UB according to L. For example, if L is 3 the calculation at each time will be as the following:

- At time $n = i$:

$$y(n) = \boldsymbol{u}^T(n)\boldsymbol{w}(i-1) \quad (15)$$
$$\boldsymbol{\Phi}(i) = P(i-1)\boldsymbol{u}(i) \quad (16)$$
$$\xi(i) = d(i) - y(i) \quad (17)$$

- At time $n = i+1$:

$$y(n) = \boldsymbol{u}^T(n)\boldsymbol{w}(i-1) \quad (18)$$
$$\boldsymbol{k}(i) = \frac{1}{\lambda + \boldsymbol{u}^T(i)\boldsymbol{\Phi}(i)}\boldsymbol{\Phi}(i) \quad (19)$$

- At time $n = i+2$:

$$y(n) = \boldsymbol{u}^T(n)\boldsymbol{w}(i-1) \quad (20)$$
$$\boldsymbol{w}(n) = \boldsymbol{w}(i-1) + \boldsymbol{k}(i)\xi(i) \quad (21)$$
$$P(n) = \lambda^{-1}\{P(i-1) - \boldsymbol{k}(i)\boldsymbol{\Phi}(i)\} \quad (22)$$

Note that at each time n the output $y(n)$ will be calculated using the same $\boldsymbol{w}(i-1)$ during L (= 3 in this example) times. We, therefore, can say that the proposed technique uses decimated tap-input vectors by factor L for updating $\boldsymbol{w}(n)$.

Although the proposed technique improves the throughput of RLS ADFs the rate of convergence of the filter is degraded at the same time. However, the rate of degradation is proportional to L provided the unknown system is time-invariant or slowly varying. Therefore, L can be regarded as a parameter to trade the throughput and the rate of convergence of an ADF. Note that this improvement does not require to increase the speed of each calculation so that the same PEs can be used to implement an ADF with improved throughput performance.

We provide a brief consideration on the convergence of the proposed technique in Appendix.

4. Consideration and Simulation Results

In this section, we consider the effectiveness of the proposed technique. First, improvement of the throughput is confirmed. Then, we provide some results of computer simulation to show the effect on the rate of convergence.

4.1 Improvement of Throughput

Let us consider the improvement of the throughput using the proposed technique. Note that here we estimate the throughput of the filter in terms of the number of multiplications.

By carefully inserting the calculations of FB as described in Sect. 3.2, the required amount of calculations for producing an output sample can be decreased. For this decrement extra $(L-1)N$ multiplications and additions are required. Note that $L-1$ instead of L because the original RLS algorithm includes the calculation for producing one output sample (see Sect. 2.2).

Hence, the total number of multiplications required to produce L samples of the output $y(n)$ becomes

$$T_{\text{total}} = 2N^2 + 6N + (L-1)N. \quad (23)$$

In this equation, we counted the symmetric property of $P(n)$ [2]. The average number of multiplications per one output sample becomes

$$T_{\text{ave}}(L) = \frac{T_{\text{total}}}{L} = \frac{2}{L}N^2 + \left(1 + \frac{5}{L}\right)N. \quad (24)$$

We can calculate the total number of addition in the same way. Using this equation, we can estimate the throughput of the filter as the following.

In Fig. 2, we show the relation between the value of L and the average number of multiplications T_{ave} per one output sample $y(n)$ with varying the length of filter. From the figure, we see that $T_{\text{ave}}(L)$ decreases for the range $L = 2 \sim 10$.

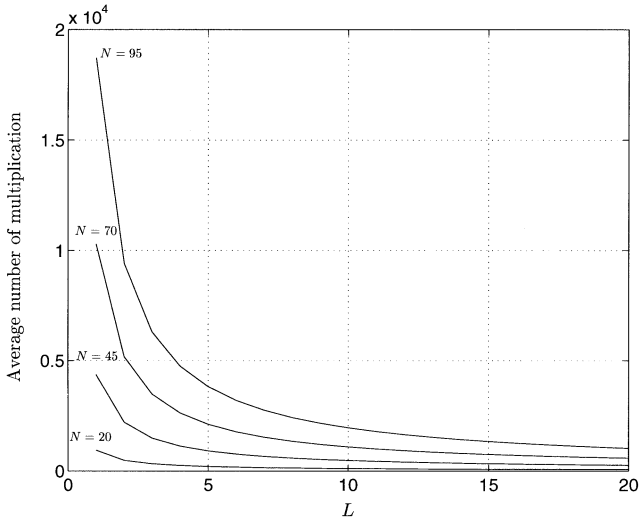


Fig. 2 Relation between values of L and the average number of multiplications per one output sample $y(n)$. Four different filter length N was used: $N = 20, 45, 70, 95$.

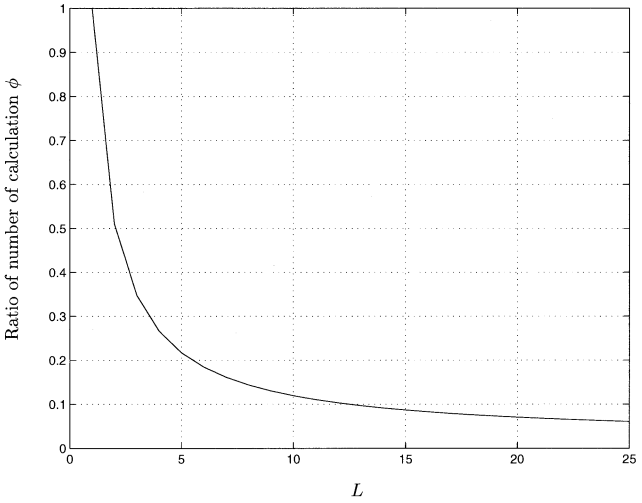


Fig. 3 Relation between values of ϕ and L.

The throughput of a filter can be considered to be in proportion to the inverse of the number of calculations for producing an output sample. Therefore, for the index to show the improvement of the throughput, we define the ratio of the required number of multiplications as

$$\begin{aligned}\phi &\stackrel{\text{def}}{=} \frac{T_{\text{ave}}(L)}{T_{\text{ave}}(1)} \\ &= \frac{2N^2/L + (1+5/L)N}{2N^2+6N} \\ &= \frac{2+L/N+5/N}{L(2+6/N)} \end{aligned} \tag{25}$$

where $T_{\text{ave}}(1)$ shows that of the standard RLS algorithm. When the condition $N \gg L$ is satisfied (25) becomes

$$\phi \sim \frac{1}{L}. \tag{26}$$

This equation shows that the required number of multiplications for producing one output sample is reduced by L. Therefore we can roughly estimate the ratio of improvement of the throughput as the inverse of ϕ, namely, it is in proportion to L.

Let us confirm this result through the calculation of (25). In Fig. 3, we show the value of ϕ. Note that the values of ϕ for different N are very close so that Fig. 3 contains only those of the case of $N = 20$. Conversely, the effect of the proposed technique does not depend on the length of the filter.

From the figure, we can say that the improvement of the throughput can be roughly estimated as

$$\phi^{-1} \sim L \tag{27}$$

for small values of L. This implies that when $L = 2$ the throughput of the filter is doubled, and when $L = 4$ four times. Note that this improvement can be done without the need for faster PEs.

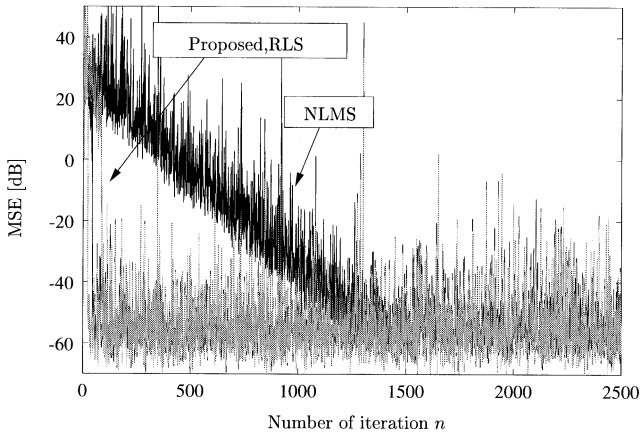


Fig. 4 Results of simulation under stable environment. The rate of convergence become smaller as L increases. The throughput of the filter, however, is increased proportionally to L (see Sect. 4.1).

4.2 Rate of Convergence

Next, we consider the effect of the proposed technique on the rate of convergence of the filter through the computer simulations.

We programmed systems identification problem with the unknown system of FIR filter whose length was 21. The ADF was also an FIR filter with the length N was 21. We compared the three algorithms, the proposed, the RLS, and the normalized LMS (NLMS) algorithms. We prepared two situations: (i) the unknown system was a time-invariant one, and (ii) the unknown system was a time-varying one by changing it at certain time to show the behavior of the proposed method in unstable environments.

We used the following values for parameters: the forgetting factor λ was 0.95 for the proposed and the RLS; α for the NLMS was 1; and L was 2, and 4 for the proposed. The input signal was AR(1) process with the AR coefficient a_0 was 0.95; and the white noise was added to $d(n)$ and the signal to noise ratio was 60 dB.

The results are shown in Fig. 4 and Fig. 6 and they are ensemble averages of 10000 independent processes for stable environment, and 40000 for unstable one. Figures 4 and 5 show the results in the time-invariant environment, and Fig. 6 those in the time-varying one.

4.2.1 Stable Environments

Figure 4 shows the results under the stable environment, and in Fig. 5, a detail of Fig. 4 is shown. From these figures, we can see the performance of an ADF using the proposed technique under the stable environments. We can confirm that the rate of convergence of the proposed method is slowed proportional to L times compared with that of the RLS: namely, the rate is doubled for the case of $L = 2$ and four times for $L = 4$. We must note that even the rate of convergence

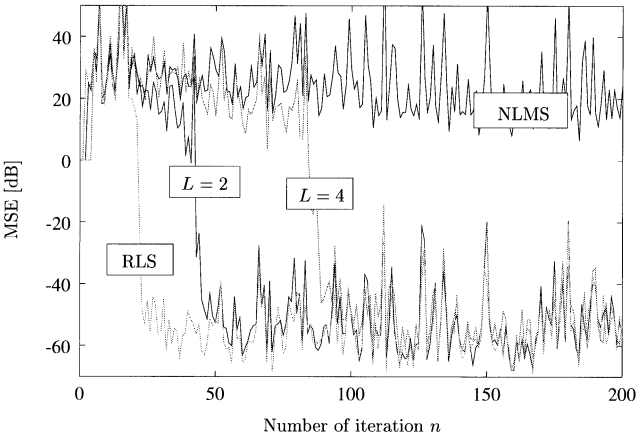


Fig. 5 Detail of simulation results under stable environment shown in Fig. 4.

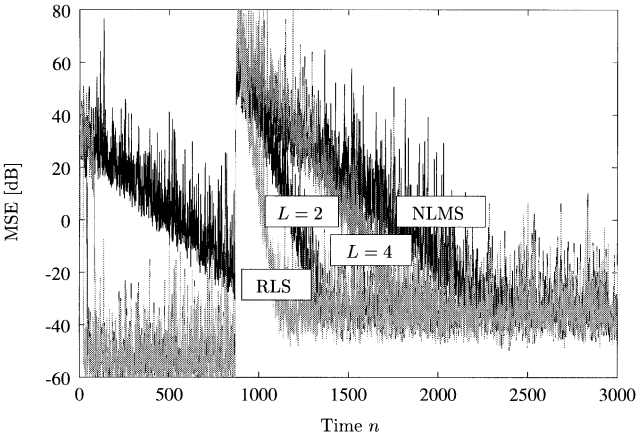


Fig. 6 Results of simulation under time-varying environment. The characteristics of the proposed method are same as those under the stable environment.

is degraded, the proposed technique can be performed at two or four times higher clock rate compared with the standard RLS as is described in the previous subsection. Moreover, we also notice that those degraded rate of convergences are still faster than that of the NLMS algorithm.

4.2.2 Time-Varying Environments

Next, from Fig. 6, the effect of the proposed technique under the time-varying environment can be seen. In this situation, the impulse response of the unknown system was changed at time $n = 700$ while its length unchanged.

It is shown that the proposed technique preserves the tracking property of the RLS algorithm although rate of convergence is lowered proportional to L as in the stationary case.

Note that the rates of convergence of the RLS and the proposed one become slower after the change of the unknown system. This phenomena is well known and is reported in references [18], [19].

5. Conclusion

In this paper, we proposed a fast technique for implementing the RLS ADFs which enables us to trade the throughput and the rate of convergence. We can adjust them according to the available amount of PEs and/or the desire throughput rate by varying the parameter L. Moreover, we showed that we can improve the throughput with no need for faster PEs. We can say that the proposed technique is suited for implementation of the RLS ADFs on FPGA or DSP chips.

Acknowledgement

This work was supported in part by the Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research 11650389.

References

[1] S. Haykin, Adaptive Filter Theory, 2nd ed., Prentice Hall, Englewood Cliffs, NJ, 1991.
[2] B. Farhang-Boroujeny, Adaptive Filters, John Wiley & Sons, Chichester, 1998.
[3] J. Thomas, "Pipelined systolic architectures for DLMS adaptvie filtering," Journal of VLSI Signal Processing, vol.12, pp.223–246, 1996.
[4] Q. Zhu, S.C. Douglas, and K.F. Smith, "A pipelined architecture for LMS adaptive FIR filter without adaptation delay," Proc. IEEE ICASSP'97, pp.1933–1936, Munich, 1997.
[5] K. Matsubara, K. Nishikawa, and H. Kiya, "A new pipelined architecture of the LMS algorithm without degradation of convergence characteristics," Proc. IEEE ICASSP'97, vol.V, pp.4125–4128, April 1997.
[6] S.C. Douglas, Q. Zhu, and K.F. Smith, "A pipelined architecture for LMS adaptive FIR filter architecture without adaptation delay," IEEE Trans. Signal Processing, vol.46, pp.775–779, March 1998.
[7] K. Matsubara, K. Nishikawa, and H. Kiya, "Pipelined LMS adaptive filter using a new look-ahead transformation," IEEE Trans. Circuits Syst. II, vol.46, no.1, pp.51–55, 1999.
[8] T. Kimijima, K. Nishikawa, and H. Kiya, "Pipelining of 2-dimensional adaptive filters based on the LDLMS algorithm," Proc. IEEE ISCAS'98, pp.V190–V193, Monterey, May 1998.
[9] A. Harada, K. Nishikawa, and H. Kiya, "A pipelined architecture for normalized LMS adaptive digital filters," IEICE Trans. Fundamentals, vol.E82-A, no.2, pp.223–229, Feb. 1999.
[10] T. Kimijima, K. Nishikawa, and H. Kiya, "An effective architecture of the pipelined LMS adaptive filters," IEICE Trans. Fundamentals, vol.E82-A, no.8, pp.1428–1434, Aug. 1999.
[11] K.J. Raghumath and K.K. Parhi, "Pipelined RLS adaptive filtering using scaled tangent rotations (STAR)," IEEE Trans. Signal Processing, vol.44, pp.2591–2604, Oct. 1996.
[12] K.J. Raghumath and K.K. Parhi, "Finite-precision error analysis of QRD-RLS and STAR-RLS adaptive filters," IEEE Trans. Signal Processing, vol.45, pp.1193–1209, May 1997.
[13] N.R. Shanbhag and K.K. Parhi, Pipelined Adaptive Digital Filters, Kluwer Academic Publishers, Massachusetts, 1994.

[14] K. Nishikawa and H. Kiya, "Low computational complexity implementation of pipelined RLS adaptive filters," Proc. ECCTD99, Stresa, Italy, Sept. 1999.
[15] B. Widrow and S.D. Stearns, Adaptive Signal Processing, Prentice Hall, Englewood Cliffs, NJ, 1985.
[16] V. Solo and X. Kong, Adaptive Signal Processing Algorithms, Prentice Hall, Englewood Cliffs, NJ, 1994.
[17] N. Kalouptsidis and S. Theodoridis, eds., Adaptive System Identification and Signal Processing Algorithms, Prentice Hall, Englewood Cliffs, NJ, 1993.
[18] E. Eleftheriou and D.D. Falconer, "Tracking properties and steady-state performance of RLS adaptive algorithms," IEEE Trans. on Acoust., Speech & Signal Processing, vol.ASSP-34, pp.1097–11098, Oct. 1986.
[19] J.M. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," IEEE Trans. on Acoust. Speech & Signal Processing, vol.ASSP-32, pp.304–337, April 1984.

Appendix: Consideration on Convergence of Proposed Technique

Here, we briefly consider the convergence of the proposed technique. Let us consider the cases of stationary environments. We assume that there is a true weight $\boldsymbol{w}_{\mathrm{o}}$ such that the output $d(n)$ is given as

$$d(n) = \boldsymbol{u}^T(n)\boldsymbol{w}_{\mathrm{o}} + \eta(n) \tag{A·1}$$

where $\eta(n)$ is the component of $d(n)$ that is uncorrelated with $\boldsymbol{u}(n)$ [16]. In the following, however, we neglect $\eta(n)$ in the sake of simplicity of description.

When the RLS algorithm is used, the optimum filter is given as

$$h_{opt} = \boldsymbol{R}^{-1}(n)\boldsymbol{p}(n) \tag{A·2}$$

where $\boldsymbol{R}(n)$ and $\boldsymbol{p}(n)$ are defined as

$$\boldsymbol{R}(n) = \sum_{i=0}^{n} \boldsymbol{u}(i)\boldsymbol{u}^T(i) \tag{A·3}$$

$$\boldsymbol{p}(n) = \sum_{i=0}^{n} \boldsymbol{u}(i)d(i). \tag{A·4}$$

By combining (A·4) with (A·1), $\boldsymbol{p}(n)$ can be rewritten as

$$\begin{aligned}\boldsymbol{p}(n) &= \sum_{i=0}^{n} \boldsymbol{u}(i)d(i) \\ &= \sum_{i=0}^{n} \boldsymbol{u}(i)\boldsymbol{u}^T(i)\boldsymbol{w}_{\mathrm{o}} \\ &= \boldsymbol{R}(n)\boldsymbol{w}_{\mathrm{o}}\end{aligned} \tag{A·5}$$

and then, the optimum filter is simply given as

$$\begin{aligned}\boldsymbol{h}_{opt} &= \boldsymbol{R}^{-1}(n)\boldsymbol{R}(n)\boldsymbol{w}_{\mathrm{o}} \\ &= \boldsymbol{w}_{\mathrm{o}}\end{aligned} \tag{A·6}$$

when the RLS algorithm is used.

In the proposed technique, the optimum filter is given as

$$\tilde{\boldsymbol{h}}_{opt} = \tilde{\boldsymbol{R}}^{-1}(n)\tilde{\boldsymbol{p}}(n) \tag{A·7}$$

where $\tilde{\boldsymbol{R}}(n)$ and $\tilde{\boldsymbol{p}}(n)$ are defined as

$$\tilde{\boldsymbol{R}}(n) = \sum_{i=0}^{m} \boldsymbol{u}(iL)\boldsymbol{u}^T(iL) \tag{A·8}$$

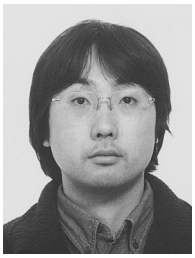
$$\tilde{\boldsymbol{p}}(n) = \sum_{i=0}^{m} \boldsymbol{u}(iL)d(iL). \tag{A·9}$$

Under the assumption of (A·1), we can apply the same manipulations as (A·5) and so that, the optimum filter is given as

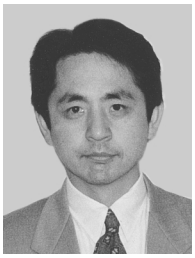
$$\tilde{\boldsymbol{h}}_{opt} = \boldsymbol{w}_{\mathrm{o}} = \boldsymbol{h}_{opt}. \tag{A·10}$$

Thus we can conclude that under the stationary environments, the proposed technique will converge to the identical optimum filter as the RLS algorithm.

Next, we consider the cases of non-stationary environments. In this case, (A·1) does not hold and the tracking property of the proposed technique varies depending on the value of L as demonstrated in Sect. 4.2.2. We can say that when the environment is slowly varying then the proposed technique can track it. On the other hand, if the environment varies quickly then the proposed technique probably fails to track it and loses its effectiveness.



Kiyoshi Nishikawa received the B.E., the M.E., and the D.E. degrees in electrical engineering from Tokyo Metropolitan University in 1990, 1992 and 1996 respectively. From 1992 to 1993, he was at the Computer Systems Laboratory, Nippon Steel Corp. as a researcher. Since 1993, he has been with Tokyo Metropolitan University as an Assistant Professor. His research interest includes the adaptive digital processing and hardware architecture of digital signal processing. He is a member of IEEE SP, CAS, Communications, and Computer Societies.



Hitoshi Kiya was born in Yamagata, Japan, on November 16, 1957. He received the B.E. and M.E. degrees in electrical engineering from Nagaoka University of Technology, Niigata, Japan, and the D.E. degree in electrical engineering from Tokyo Metropolitan University, Tokyo, Japan, in 1980, 1982, and 1987, respectively. In 1982, he joined Tokyo Metropolitan University, where he is currently a Professor of Electrical Engineering. His research interests are in digital signal processing, multirate systems, adaptive filtering, image processing, and efficient algorithms for VLSI implementation. Dr. Kiya is a Member of the Institute of Electrical and Electronics Engineers, Inc. (IEEE) of USA, the Image Electronics Engineers of Japan and the Institute of Television Engineers of Japan.