A Numerically Stable Fast Newton-Type Adaptive Filter Based on Order Recursive Least Squares Algorithm

Youhua Wang, *Member, IEEE*, Kazushi Ikeda, *Member, IEEE*, and Kenji Nakayama, *Senior Member, IEEE*

***Abstract*—This paper presents a numerically stable fast Newton-type adaptive filter algorithm. Two problems are dealt with in the paper. First, we derive the proposed algorithm from an order-recursive least squares algorithm. The result of the proposed algorithm is equivalent to that of the fast Newton transversal filter (FNTF) algorithm. However, the derivation process is different. Instead of extending a covariance matrix of the input based on the min-max and the max-min criteria, the derivation shown in this paper is to solve an optimum extension problem of the gain vector based on the information of the Mth-order forward or backward predictor. The derivation provides an intuitive explanation of the FNTF algorithm, which may be easier to understand. Second, we present stability analysis of the proposed algorithm using a linear time-variant state-space method. We show that the proposed algorithm has a well-analyzable stability structure, which is indicated by a transition matrix. The eigenvalues of the ensemble average of the transition matrix are proved all to be asymptotically less than unity. This results in a much-improved numerical performance of the proposed algorithm compared with the combination of the stabilized fast recursive least squares (SFRLS) and the FNTF algorithms. Computer simulations implemented by using a finite-precision arithmetic have confirmed the validity of our analysis.**

***Index Terms*—Adaptive filter, FNTF algorithm, FRLS algorithm, LSL algorithm, numerical stability, RLS algorithm, stabilized FRLS algorithm.**

I. Introduction

THE recursive least squares (RLS) and the fast recursive least squares (FRLS) algorithms are two well-known approaches for solving the exact least squares solution in the transversal adaptive filters. Unfortunately, both algorithms suffer from the numerical instability problem under a finite-precision implementation [1]–[11]. In the RLS algorithm, a well-known example is the loss of symmetry and positive definiteness of the inverse correlation matrix [1]–[4]. This causes an explosive divergence of the RLS algorithm. On the other hand, the instability of the FRLS algorithm is mainly produced by a hyperbolic rotation (causing the eigenvalues to exit the unit circle) that has to be operated on the backward predictor in order to obtain the recursive equations for computing the gain vector [6], [7]. Several error feedback methods were proposed for overcoming the instability problem of the FRLS algorithm [10]–[12]. However, the stable performance obtained by the error feedback mechanism is maintained, provided that the forgetting factor λ is restricted to a range of $\lambda \in (1-1/3M, 1)$, where M denotes the order of the predictors [12]. This may degrade the tracking ability of the FRLS algorithm.

In the FRLS algorithm, however, if we assume that the recursions involve both order- and time-update, then the least squares solution can be obtained by using either the forward or backward predictor. Therefore, the stable structures of both the forward predictor and the backward predictor are retained. This leads to the algorithm we called the predictor-based least squares (PLS) algorithm, which includes the forward PLS (FPLS) and the backward PLS (BPLS) algorithms.

Although the PLS algorithm can be easily derived from the FRLS algorithm, very few investigations concerning its numerical properties are reported in the literature. In [13], we have introduced the PLS algorithm and investigated its numerical performance. It has been shown that three major instability sources reported in the literature, including the numerical instability of the conversion factor, the loss of symmetry, and the loss of positive definiteness of the inverse correlation matrix of the input, do not exist in the PLS algorithm. This results in a much improved numerical performance of the PLS algorithm compared with the RLS algorithm.

Unfortunately, the computational load of the PLS algorithm is $O(N^2)$. This makes it difficult to be implemented in real-time applications, even using today's DSP technology. In order to overcome this difficulty, a fast PLS algorithm is derived in this paper. The assumption for the fast PLS algorithm is the same as that of the fast Newton transversal filter (FNTF) algorithm [14], that is, if the input signal can be adequately modeled by an autoregressive of order M, where M can be selected much smaller than the order N of the adaptive filter, then the gain vector can be extended from M to N based on the predictor and the gain vector of order M without sacrificing performance. However, the derivation presented in this paper is different from that of [14]. Instead of extending a covariance matrix of the input based on the max-min and the min-max criteria that are somewhat difficult to understand, the derivation shown in this paper is to solve an optimum extension problem of the gain vector based on the

Manuscript received March 4, 2002; revised February 20, 2003. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sergios Theodoridis.

Y. Wang is with Panasonic Mobile Communications Kanazawa R&D Laboratory Co., Ltd., Kanazawa 920-0024, Japan (e-mail: wang.youhua@jp.panasonic.com).

K. Ikeda is with the Department of Systems Science, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (e-mail: kazushi@i.kyoto-u.ac.jp).

K. Nakayama is with the Department of Information and Systems Engineering, Kanazawa University, Kanazawa 920-8667, Japan (e-mail: nakayama@t.kanazawa-u.ac.jp).

Digital Object Identifier 10.1109/TSP.2003.815357

information of the Mth-order forward or backward predictor. Hence, the derivation gives an intuitive explanation of the FNTF algorithm.

The most important characteristics of the PLS algorithm and its fast version are their excellent numerical properties. In the paper, we adopt a linear time-variant state-space method for their stability analysis. The transition matrices of the PLS algorithm and its fast version are indicated, and their ensemble average is evaluated. The eigenvalues are shown all to be asymptotically less than unity. In [15] and [16], we gave the derivation of the fast BPLS algorithm and investigated its numerical and convergence properties. In this paper, we will present refined derivation and stability analysis for both the fast FPLS and the fast BPLS algorithms. We will show the equivalence between the proposed fast PLS algorithm and the FNTF algorithm and compare the numerical performance of the fast PLS algorithm with that of the combination of the stabilized FRLS and the FNTF algorithms [we will hereinafter call the stabilized FNTF (SFNTF) algorithm].

We notice that the least squares lattice (LSL) algorithm, which is also an order-recursive adaptive filter, was reported to have a well-behaved numerical performance [17]–[19]. The PLS algorithm is different from the LSL algorithm in that it provides an explicit relationship between the gain vector used in the transversal filter and the predictors used in the lattice filter. It is therefore possible to obtain a numerically well-behaved transversal adaptive filter and reduce the computational cost by exploiting the autoregressive property of the input signal that may be modeled by a lower order of predictors. Therefore, the PLS algorithm may be considered as the transversal counterpart of the adaptive lattice filter.

The organization of this paper is as follows: The PLS algorithm is introduced in the next section. The derivation of the fast PLS algorithm is presented in Section III. In Section IV, the numerical properties of the PLS and the fast PLS algorithms are analyzed. In Section V, some computer simulations, which were implemented by using a variety of wordlength arithmetic, were carried out to confirm the stability analysis and show the numerical performances among several typical least squares and fast Newton-type algorithms. The conclusion remarks are given in Section VI.

II. Predictor-Based Least Squares Algorithm

The PLS algorithm can be derived from any version of the FRLS algorithm. Since only one predictor, forward or backward, is needed, we can write two versions of the PLS algorithm by using the definitions and the symbols that are consistent with those used in [19] as follows.

For both versions, when time $n = 1, 2, 3, \ldots$, compute the order updates in the following sequence: $m = 1, 2, \ldots N$, where N is the final order of the predictor.

A. Forward Predictor-Based Least Squares (FPLS) Algorithm

$$\eta_m(n) = u(n) + \mathbf{a}_m^T(n-1)\mathbf{u}_m(n-1) \tag{1}$$

$$F_m(n) = \lambda F_m(n-1) + \gamma_m(n-1)\eta_m^2(n) \tag{2}$$

$$\gamma_{m+1}(n) = \frac{\lambda F_m(n-1)}{F_m(n)}\gamma_m(n-1) \tag{3}$$

$$\tilde{\mathbf{k}}_{m+1}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_m(n-1) \end{bmatrix} + \frac{\eta_m(n)}{\lambda F_m(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n-1) \end{bmatrix} \tag{4}$$

$$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) - \gamma_m(n-1)\eta_m(n)\tilde{\mathbf{k}}_m(n-1) \tag{5}$$

where $\eta_m(n)$ is the forward *a priori* prediction error, $F_m(n)$ is the minimum power of $\eta_m(n)$, $\gamma_m(n)$ is the conversion factor, $\tilde{\mathbf{k}}_m(n)$ is the normalized gain vector, $\mathbf{a}_m(n)$ is the tap-weight vector of the forward predictor, and $\mathbf{u}_m(n)$ is the tap-input vector.

B. Backward Predictor-Based Least Squares (BPLS) Algorithm

$$\psi_m(n) = \mathbf{c}_m^T(n-1)\mathbf{u}_m(n) + u(n-m) \tag{6}$$

$$B_m(n) = \lambda B_m(n-1) + \gamma_m(n)\psi_m^2(n) \tag{7}$$

$$\gamma_{m+1}(n) = \frac{\lambda B_m(n-1)}{B_m(n)}\gamma_m(n) \tag{8}$$

$$\tilde{\mathbf{k}}_{m+1}(n) = \begin{bmatrix} \tilde{\mathbf{k}}_m(n) \\ 0 \end{bmatrix} + \frac{\psi_m(n)}{\lambda B_m(n-1)} \begin{bmatrix} \mathbf{c}_m(n-1) \\ 1 \end{bmatrix} \tag{9}$$

$$\mathbf{c}_m(n) = \mathbf{c}_m(n-1) - \gamma_m(n)\psi_m(n)\tilde{\mathbf{k}}_m(n) \tag{10}$$

where $\psi_m(n)$ is the backward *a priori* prediction error, $B_m(n)$ is the minimum power of $\psi_m(n)$, and $\mathbf{c}_m(n)$ is the tap-weight vector of the backward predictor.

The filtering part is common for both the FPLS and BPLS algorithms.

$$e(n) = d(n) - \mathbf{w}_N^T(n-1)\mathbf{u}_N(n) \tag{11}$$

$$\mathbf{w}_N(n) = \mathbf{w}_N(n-1) + e(n)\gamma_N(n)\tilde{\mathbf{k}}_N(n) \tag{12}$$

where $e(n)$ is the *a priori* estimation error, $d(n)$ is the desired signal, and $\mathbf{w}_N(n)$ is the tap-weight vector of the adaptive filter.

To initialize the PLS algorithm at time $n = 0$, set

$$\mathbf{a}_m(0) = \mathbf{c}_m(0) = \mathbf{0}_m \tag{13}$$

$$F_m(0) = B_m(0) = \delta \tag{14}$$

$$\tilde{\mathbf{k}}_m(0) = \mathbf{0}_m \tag{15}$$

$$\gamma_m(0) = 1 \tag{16}$$

where $m = 1, 2, \ldots, N$, and δ is a small positive constant.

At each iteration $n \geq 1$, generate the first-order variables as follows:

$$\tilde{k}_1(n) = \frac{1}{\lambda}\Phi_1^{-1}(n-1)u(n) = \frac{u(n)}{\lambda\Phi_1(n-1)} \tag{17}$$

$$\gamma_1(n) = \frac{1}{1 + \tilde{k}_1(n)u(n)} = \frac{\lambda\Phi_1(n-1)}{\Phi_1(n)} \tag{18}$$

where $\Phi_1(n)$ is the first-order of the input sample correlation matrix that satisfies

$$\Phi_1(n) = \lambda\Phi_1(n-1) + u^2(n) \tag{19}$$

where $\Phi_1(0) = \delta$.

III. PLS-Based Fast Newton Adaptive Filter Algorithm

The PLS-based fast Newton adaptive filter algorithm, which we also call the fast PLS algorithm, can be derived based on either the FPLS or the BPLS algorithm.

Assume that the input signal can be represented by an autoregressive model of order M (AR(M)), implying that the use of the predictor of order M is adequate and that the tap weights of the forward predictor $\mathbf{a}_m(n)$ or the backward predictor $\mathbf{c}_m(n)$ that have order greater than M are zeros. The problem is how to extend the Mth-order normalized gain vector $\tilde{\mathbf{k}}_M(n)$ to the Nth-order normalized gain vector $\tilde{\mathbf{k}}_N(n)$ based on the knowledge of the Mth-order forward or backward predictor with optimum estimation and least increase of computation.

A. Fast FPLS Algorithm

For the fast FPLS algorithm, we want to show that when only the information of the Mth-order forward predictor is available, the optimum extension of the normalized gain vector $\tilde{\mathbf{k}}_m(n)$ for $m > M$ satisfies the following equation:

$$\tilde{\mathbf{k}}_m(n) = \begin{bmatrix} \mathbf{0}_{m-M} \\ \tilde{\mathbf{k}}_M(n-m+M) \end{bmatrix} + \sum_{i=0}^{m-M-1} \frac{\eta_M(n-i)}{\lambda F_M(n-i-1)} \times \begin{bmatrix} \mathbf{0}_i \\ 1 \\ \mathbf{a}_M(n-i-1) \\ \mathbf{0}_{m-M-i-1} \end{bmatrix}. \quad (20)$$

To prove (20), we first compute the FPLS algorithm up to the Mth order to obtain $\tilde{\mathbf{k}}_{M+1}(n)$ and the forward predictor $\mathbf{a}_M(n-1)$. Then, we write (4) for $m = M+1$ as

$$\tilde{\mathbf{k}}_{M+2}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{M+1}(n-1) \end{bmatrix} + \frac{\eta_{M+1}(n)}{\lambda F_{M+1}(n-1)} \times \begin{bmatrix} 1 \\ \mathbf{a}_{M+1}(n-1) \end{bmatrix}. \quad (21)$$

From the assumption, the last term of $\mathbf{a}_{M+1}(n-1)$ is zero, that is

$$\mathbf{a}_{M+1}(n-1) = \begin{bmatrix} \bar{\mathbf{a}}_M(n-1) \\ 0 \end{bmatrix} \quad (22)$$

where $\bar{\mathbf{a}}_M(n-1)$ denotes the elements of $\mathbf{a}_{M+1}(n-1)$ up to the Mth order.

We want to determine the tap-weight vector of the forward predictor $\mathbf{a}_{M+1}(n-1)$ or $\bar{\mathbf{a}}_M(n-1)$ so that the forward *a priori* prediction error $\eta_{M+1}(n)$ and its error power $F_{M+1}(n)$ can be minimized. Since

$$\begin{aligned}\eta_{M+1}(n) &= u(n) + \mathbf{a}_{M+1}^T(n-1)\mathbf{u}_{M+1}(n-1) \\ &= u(n) + \bar{\mathbf{a}}_M^T(n-1)\mathbf{u}_M(n-1)\end{aligned} \quad (23)$$

comparing (23) with (1), it is not difficult to see that when $\bar{\mathbf{a}}_M^T(n-1) = \mathbf{a}_M^T(n-1)$, the minimum forward *a priori* prediction error $\eta_M(n)$ (least squares solution) can be obtained. This means that

$$\eta_M(n) = \min\{\eta_{M+1}(n)\}|_{\bar{\mathbf{a}}_M^T(n-1)=\mathbf{a}_M^T(n-1)}. \quad (24)$$

On the other hand, from the definition of the forward *a posteriori* predictor error $f_{M+1}(n)$ [19]

$$\begin{aligned}f_{M+1}(n) &= \gamma_{M+1}(n-1)\eta_{M+1}(n) \\ &= u(n) + \mathbf{a}_{M+1}^T(n)\mathbf{u}_{M+1}(n-1) \\ &= u(n) + \bar{\mathbf{a}}_M^T(n)\mathbf{u}_M(n-1)\end{aligned} \quad (25)$$

we can readily deduce that

$$f_M(n) = \min\{f_{M+1}(n)\}|_{\bar{\mathbf{a}}_M^T(n)=\mathbf{a}_M^T(n)}. \quad (26)$$

Under the constraint of using $\eta_M(n)$ and $f_M(n)$, from (2), the minimum prediction error power we can get is

$$F_M(n) = \lambda F_M(n-1) + f_M(n)\eta_M(n) \quad (27)$$

which means

$$F_M(n) = \min\{F_{M+1}(n)\}. \quad (28)$$

Therefore, the optimum estimates of $\mathbf{a}_{M+1}(n-1), \eta_{M+1}(n)$, and $F_{M+1}(n-1)$ satisfy the following equation:

$$\frac{\eta_{M+1}(n)}{\lambda F_{M+1}(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_{M+1}(n-1) \end{bmatrix} = \frac{\eta_M(n)}{\lambda F_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_M(n-1) \\ 0 \end{bmatrix}. \quad (29)$$

Substituting (29) into (21), we get

$$\begin{aligned}\tilde{\mathbf{k}}_{M+2}(n) &= \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_{M+1}(n-1) \end{bmatrix} + \frac{\eta_M(n)}{\lambda F_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_M(n-1) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ \tilde{\mathbf{k}}_M(n-2) \end{bmatrix} + \frac{\eta_M(n-1)}{\lambda F_M(n-2)} \begin{bmatrix} 0 \\ 1 \\ \mathbf{a}_M(n-2) \end{bmatrix} \\ &\quad + \frac{\eta_M(n)}{\lambda F_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_M(n-1) \\ 0 \end{bmatrix}.\end{aligned} \quad (30)$$

Continuing in this procedure, we can prove (20). Therefore, the update equation for order N of the normalized gain vector $\tilde{\mathbf{k}}_N(n)$ can be written as

$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \mathbf{0}_{N-M} \\ \tilde{\mathbf{k}}_M(n-N+M) \end{bmatrix} + \sum_{i=0}^{N-M-1} \frac{\eta_M(n-i)}{\lambda F_M(n-i-1)} \begin{bmatrix} \mathbf{0}_i \\ 1 \\ \mathbf{a}_M(n-i-1) \\ \mathbf{0}_{N-M-i-1} \end{bmatrix}. \quad (31)$$

Notice from (31) that no additional computation is needed to obtain $\tilde{\mathbf{k}}_N(n)$, except for some delays when $m > M$. This is the key point that makes the computational reduction of the fast FPLS algorithm possible.

In summary, when only the information of the Mth-order forward predictor is available, the optimum extension of the predictor for $m > M$ satisfies the following relations:

$$\eta_m(n) = \eta_M(n) \tag{32}$$

$$F_m(n) = F_M(n) \tag{33}$$

$$\mathbf{a}_m(n) = \begin{bmatrix} \mathbf{a}_M(n) \\ \mathbf{0}_{m-M} \end{bmatrix}. \tag{34}$$

Based on the above relations, we can derive the extension of the conversion factor $\gamma_N(n)$ that is needed for obtaining the gain vector $\mathbf{k}_N(n) = \gamma_N(n)\tilde{\mathbf{k}}_N(n)$ as follows.

From (2) and (3), we rewrite the order-update equation of $\gamma_{m+1}(n)$ as

$$\frac{1}{\gamma_{m+1}(n)} = \frac{1}{\gamma_m(n-1)} + \frac{\eta_m^2(n)}{\lambda F_m(n-1)}. \tag{35}$$

For $m = M+1$, (35) is shown to be

$$\begin{aligned}\frac{1}{\gamma_{M+2}(n)} &= \frac{1}{\gamma_{M+1}(n-1)} + \frac{\eta_{M+1}^2(n)}{\lambda F_{M+1}(n-1)} \\ &= \frac{1}{\gamma_M(n-2)} + \frac{\eta_M^2(n-1)}{\lambda F_M(n-2)} + \frac{\eta_M^2(n)}{\lambda F_M(n-1)}.\end{aligned} \tag{36}$$

Following this way, $\gamma_N(n)$ can be obtained by

$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n-N+M)} + \sum_{i=0}^{N-M-1} \frac{\eta_M^2(n-i)}{\lambda F_M(n-i-1)}. \tag{37}$$

The summations on the right side of (31) and (37) can be further simplified. Let $\mathbf{g}_N(n)$ denote the summation part of (31); then, we can write

$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \mathbf{0}_{N-M} \\ \tilde{\mathbf{k}}_M(n-N+M) \end{bmatrix} + \mathbf{g}_N(n) \tag{38}$$

where the recursive equation for $\mathbf{g}_N(n)$ is given by

$$\begin{aligned}\begin{bmatrix} \mathbf{g}_N(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} 0 \\ \mathbf{g}_N(n-1) \end{bmatrix} + \frac{\eta_M(n)}{\lambda F_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_M(n-1) \\ \mathbf{0}_{N-M} \end{bmatrix} \\ &- \frac{\eta_M(n-N+M)}{\lambda F_M(n-N+M-1)} \begin{bmatrix} \mathbf{0}_{N-M} \\ 1 \\ \mathbf{a}_M(n-N+M-1) \end{bmatrix}.\end{aligned} \tag{39}$$

For the same reason, let $g(n)$ denote the summation part of (37); then $\gamma_N(n)$ becomes

$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n-N+M)} + g(n) \tag{40}$$

where $g(n)$ is given by

$$\begin{aligned}g(n) = g(n-1) &+ \frac{\eta_M^2(n)}{\lambda F_M(n-1)} \\ &- \frac{\eta_M^2(n-N+M)}{\lambda F_M(n-N+M-1)}.\end{aligned} \tag{41}$$

We note that the results of (38)–(41) are equivalent to the Version 2 of the FNTF algorithm.

B. Fast BPLS Algorithm

For the fast BPLS algorithm, the optimum extension of the normalized gain vector $\tilde{\mathbf{k}}_m(n)$ for $m > M$ is shown by

$$\tilde{\mathbf{k}}_m(n) = \begin{bmatrix} \tilde{\mathbf{k}}_M(n) \\ \mathbf{0}_{m-M} \end{bmatrix} + \sum_{i=0}^{m-M-1} \frac{\psi_M(n-i)}{\lambda B_M(n-i-1)} \times \begin{bmatrix} \mathbf{0}_i \\ \mathbf{c}_M(n-i-1) \\ 1 \\ \mathbf{0}_{m-M-i-1} \end{bmatrix}. \tag{42}$$

To prove (42), we first compute $\tilde{\mathbf{k}}_{M+1}(n)$ and $\mathbf{c}_M(n-1)$ in the BPLS algorithm. Then, we write (9) for $m = M+1$ as

$$\tilde{\mathbf{k}}_{M+2}(n) = \begin{bmatrix} \tilde{\mathbf{k}}_{M+1}(n) \\ 0 \end{bmatrix} + \frac{\psi_{M+1}(n)}{\lambda B_{M+1}(n-1)} \begin{bmatrix} \mathbf{c}_{M+1}(n-1) \\ 1 \end{bmatrix}. \tag{43}$$

From the assumption, the first term of $\mathbf{c}_{M+1}(n-1)$ is zero, that is

$$\mathbf{c}_{M+1}(n-1) = \begin{bmatrix} 0 \\ \bar{\mathbf{c}}_M(n-1) \end{bmatrix} \tag{44}$$

where $\bar{\mathbf{c}}_M(n-1)$ denotes the elements of $\mathbf{c}_{M+1}(n-1)$ up to the Mth order.

We want to determine the tap-weight vector of the backward predictor $\mathbf{c}_{M+1}(n-1)$ or $\bar{\mathbf{c}}_M(n-1)$ so that the backward *a priori* prediction error $\psi_{M+1}(n)$ and its error power $B_{M+1}(n)$ can be minimized. Since

$$\begin{aligned}\psi_{M+1}(n) &= \mathbf{c}_{M+1}^T(n-1)\mathbf{u}_{M+1}(n) + u(n-M-1) \\ &= \bar{\mathbf{c}}_M^T(n-1)\mathbf{u}_M(n-1) + u(n-M-1)\end{aligned} \tag{45}$$

from (6), we can see that the optimum predictor, which uses $u(n-1), \ldots, u(n-M)$ to predict $u(n-M-1)$, is $\mathbf{c}_M(n-2)$, which satisfies

$$\psi_M(n-1) = \mathbf{c}_M^T(n-2)\mathbf{u}_M(n-1) + u(n-M-1) \tag{46}$$

where $\psi_M(n-1)$ is the minimum prediction error (least squares solution).

Comparing (45) with (46), we can see that the minimum prediction error $\psi_M(n-1)$ can be obtained by simply choosing $\bar{\mathbf{c}}_M^T(n-1) = \mathbf{c}_M^T(n-2)$. That is

$$\psi_M(n-1) = \min\{\psi_{M+1}(n)\}|_{\bar{\mathbf{c}}_M^T(n-1)=\mathbf{c}_M^T(n-2)} \tag{47}$$

From the definition of the backward *a posteriori* predictor error $b_{M+1}(n)$ [19]

$$\begin{aligned}b_{M+1}(n) &= \gamma_{M+1}(n)\psi_{M+1}(n) \\ &= \mathbf{c}_{M+1}^T(n)\mathbf{u}_{M+1}(n) + u(n-M-1) \\ &= \bar{\mathbf{c}}_M^T(n)\mathbf{u}_M(n-1) + u(n-M-1)\end{aligned} \tag{48}$$

TABLE I
SUMMARY OF THE FAST PREDICTOR-BASED LEAST SQUARES (FAST PLS) ALGORITHM

Initialization:

For $n = 0$ and $m = 1, 2, \ldots, M$, set the parameters of the forward and backward predictors as follows:

$$\mathbf{a}_m(0) = \mathbf{c}_m(0) = \mathbf{0}_m, \quad F_m(0) = B_m(0) = \delta, \quad \tilde{\mathbf{k}}_m(0) = \mathbf{0}_m, \quad \gamma_m(0) = 1$$

Set the parameters of the extended gain vector and the adaptive filter as follows:

$$\mathbf{g}_N(0) = \mathbf{h}_N(0) = \mathbf{0}_N, \quad g(0) = h(0) = 0, \quad \mathbf{w}_N(0) = \mathbf{0}_N$$

At each iterations $n \geq 1$, compute the first-order variables as follows:

$$\Phi_1(n) = \lambda\Phi_1(n-1) + u^2(n) \qquad (\Phi_1(0) = \delta)$$

$$\tilde{k}_1(n) = u(n)/(\lambda\Phi_1(n-1))$$

$$\gamma_1(n) = \lambda\Phi_1(n-1)/\Phi_1(n)$$

1. The Fast Forward Predictor-Based Least Squares (Fast FPLS) Algorithm

Prediction:

For $n = 1, 2, 3, \ldots$, compute the following order updates in the sequence $m = 1, 2, \ldots, M$:

$$\eta_m(n) = u(n) + \mathbf{a}_m^T(n-1)\mathbf{u}_m(n-1)$$

$$F_m(n) = \lambda F_m(n-1) + \gamma_m(n-1)\eta_m^2(n)$$

$$\gamma_{m+1}(n) = \frac{\lambda F_m(n-1)}{F_m(n)}\gamma_m(n-1)$$

$$\tilde{\mathbf{k}}_{m+1}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_m(n-1) \end{bmatrix} + \frac{\eta_m(n)}{\lambda F_m(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n-1) \end{bmatrix}$$

$$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) - \gamma_m(n-1)\eta_m(n)\tilde{\mathbf{k}}_m(n-1)$$

Gain vector extension:

Based on $\eta_M(n)$, $F_M(n-1)$, $\mathbf{a}_M(n-1)$, $\eta_M(n-\tau)$, $F_M(n-\tau-1)$, $\mathbf{a}_M(n-\tau-1)$, $\tilde{\mathbf{k}}_M(n-\tau)$ and $\gamma_M(n-\tau)$, where $\tau = N - M$, compute $\tilde{\mathbf{k}}_N(n)$ and $\gamma_N(n)$ as follows:

$$\begin{bmatrix} \mathbf{g}_N(n) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{g}_N(n-1) \end{bmatrix} + \frac{\eta_M(n)}{\lambda F_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_M(n-1) \\ \mathbf{0}_{N-M} \end{bmatrix} - \frac{\eta_M(n-\tau)}{\lambda F_M(n-\tau-1)} \begin{bmatrix} \mathbf{0}_{N-M} \\ 1 \\ \mathbf{a}_M(n-\tau-1) \end{bmatrix}$$

$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \mathbf{0}_{N-M} \\ \tilde{\mathbf{k}}_M(n-\tau) \end{bmatrix} + \mathbf{g}_N(n)$$

$$g(n) = g(n-1) + \frac{\eta_M^2(n)}{\lambda F_M(n-1)} - \frac{\eta_M^2(n-\tau)}{\lambda F_M(n-\tau-1)}$$

$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n-\tau)} + g(n)$$

we can easily deduce that

$$b_M(n-1) = \min\{b_{M+1}(n)\}|_{\bar{\mathbf{c}}_M^T(n)=\mathbf{c}_M^T(n-1)}. \tag{49}$$

Under the constraint of using $\psi_M(n-1)$ and $b_M(n-1)$, from (7), the minimum prediction error power we can get is

$$B_M(n-1) = \lambda B_M(n-2) + b_M(n-1)\psi_M(n-1) \tag{50}$$

which means

$$B_M(n-1) = \min\{B_{M+1}(n)\}. \tag{51}$$

Therefore, the optimum estimates of $\mathbf{c}_{M+1}(n-1), \psi_{M+1}(n)$ and $B_{M+1}(n-1)$ satisfy the following equation:

$$\frac{\psi_{M+1}(n)}{\lambda B_{M+1}(n-1)} \begin{bmatrix} \mathbf{c}_{M+1}(n-1) \\ 1 \end{bmatrix} = \frac{\psi_M(n-1)}{\lambda B_M(n-2)} \begin{bmatrix} 0 \\ \mathbf{c}_M(n-2) \\ 1 \end{bmatrix}. \tag{52}$$

TABLE I *(Continued)*

2. The Fast Backward Predictor-Based Least Squares (Fast BPLS) Algorithm

Prediction:

For $n = 1, 2, 3, \ldots,$, compute the following order updates in the sequence $m = 1, 2, \ldots, M$:

$$\psi_m(n) = \mathbf{c}_m^T(n-1)\mathbf{u}_m(n) + u(n-m)$$
$$B_m(n) = \lambda B_m(n-1) + \gamma_m(n)\psi_m^2(n)$$
$$\gamma_{m+1}(n) = \frac{\lambda B_m(n-1)}{B_m(n)}\gamma_m(n)$$
$$\tilde{\mathbf{k}}_{m+1}(n) = \begin{bmatrix} \tilde{\mathbf{k}}_m(n) \\ 0 \end{bmatrix} + \frac{\psi_m(n)}{\lambda B_m(n-1)} \begin{bmatrix} \mathbf{c}_m(n-1) \\ 1 \end{bmatrix}$$
$$\mathbf{c}_m(n) = \mathbf{c}_m(n-1) - \gamma_m(n)\psi_m(n)\tilde{\mathbf{k}}_m(n)$$

Gain vector extension:

Based on $\psi_M(n)$, $B_M(n-1)$, $\mathbf{c}_M(n-1)$, $\psi_M(n-\tau)$, $B_M(n-\tau-1)$, $\mathbf{c}_M(n-\tau-1)$, $\tilde{\mathbf{k}}_M(n)$ and $\gamma_M(n)$, where $\tau = N - M$, compute $\tilde{\mathbf{k}}_N(n)$ and $\gamma_N(n)$ as follows:

$$\begin{bmatrix} \mathbf{h}_N(n) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{h}_N(n-1) \end{bmatrix} + \frac{\psi_M(n)}{\lambda B_M(n-1)} \begin{bmatrix} \mathbf{c}_M(n-1) \\ 1 \\ \mathbf{0}_{N-M} \end{bmatrix} - \frac{\psi_M(n-\tau)}{\lambda B_M(n-\tau-1)} \begin{bmatrix} \mathbf{0}_{N-M} \\ \mathbf{c}_M(n-\tau-1) \\ 1 \end{bmatrix}$$
$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \tilde{\mathbf{k}}_M(n) \\ \mathbf{0}_{N-M} \end{bmatrix} + \mathbf{h}_N(n)$$
$$h(n) = h(n-1) + \frac{\psi_M^2(n)}{\lambda B_M(n-1)} - \frac{\psi_M^2(n-\tau)}{\lambda B_M(n-\tau-1)}$$
$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n)} + h(n)$$

Filtering:

The filtering part is common for both the fast FPLS and the fast BPLS algorithms:

$$e(n) = d(n) - \mathbf{w}_N^T(n-1)\mathbf{u}_N(n)$$
$$\mathbf{w}_N(n) = \mathbf{w}_N(n-1) + e(n)\gamma_N(n)\tilde{\mathbf{k}}_N(n)$$

Substituting (52) into (43), we have

$$\begin{aligned} \tilde{\mathbf{k}}_{M+2}(n) &= \begin{bmatrix} \tilde{\mathbf{k}}_{M+1}(n) \\ 0 \end{bmatrix} + \frac{\psi_M(n-1)}{\lambda B_M(n-2)} \begin{bmatrix} 0 \\ \mathbf{c}_M(n-2) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{k}}_M(n) \\ 0 \\ 0 \end{bmatrix} + \frac{\psi_M(n)}{\lambda B_M(n-1)} \begin{bmatrix} \mathbf{c}_M(n-1) \\ 1 \\ 0 \end{bmatrix} \\ &\quad + \frac{\psi_M(n-1)}{\lambda B_M(n-2)} \begin{bmatrix} 0 \\ \mathbf{c}_M(n-2) \\ 1 \end{bmatrix}. \end{aligned} \tag{53}$$

Following the same procedure, (42) can be proved. Therefore, the update equation for $\tilde{\mathbf{k}}_N(n)$ can be written as

$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \tilde{\mathbf{k}}_M(n) \\ \mathbf{0}_{N-M} \end{bmatrix} + \sum_{i=0}^{N-M-1} \frac{\psi_M(n-i)}{\lambda B_M(n-i-1)} \times \begin{bmatrix} \mathbf{0}_i \\ \mathbf{c}_M(n-i-1) \\ 1 \\ \mathbf{0}_{N-M-i-1} \end{bmatrix}. \tag{54}$$

Notice also from (54) that no additional computation is needed for obtaining $\tilde{\mathbf{k}}_N(n)$, except for some delays when $m > M$.

In summary, when only the information of the Mth-order backward predictor is available, the optimum extension of the predictor for $m > M$ satisfies the following relations:

$$\psi_m(n) = \psi_M(n-m+M) \tag{55}$$

$$B_m(n) = B_M(n-m+M) \tag{56}$$

$$\mathbf{c}_m(n) = \begin{bmatrix} \mathbf{0}_{m-M} \\ \mathbf{c}_M(n-m+M) \end{bmatrix}. \tag{57}$$

The extension of the conversion factor $\gamma_N(n)$ can be calculated as follows. From (7) and (8), we rewrite the order update equation of $\gamma_{m+1}(n)$ as

$$\frac{1}{\gamma_{m+1}(n)} = \frac{1}{\gamma_m(n)} + \frac{\psi_m^2(n)}{\lambda B_m(n-1)}. \tag{58}$$

For $m = M+1$ and using the relations of (55) and (56), (58) can be written as

$$\begin{aligned} &\frac{1}{\gamma_{M+2}(n)} \\ &= \frac{1}{\gamma_{M+1}(n)} + \frac{\psi_{M+1}^2(n)}{\lambda B_{M+1}(n-1)} \\ &= \frac{1}{\gamma_M(n)} + \frac{\psi_M^2(n)}{\lambda B_M(n-1)} + \frac{\psi_M^2(n-1)}{\lambda B_M(n-2)}. \end{aligned} \tag{59}$$

Following this procedure, $\gamma_N(n)$ can be obtained by

$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n)} + \sum_{i=0}^{N-M-1} \frac{\psi_M^2(n-i)}{\lambda B_M(n-i-1)}. \tag{60}$$

The summations on the right side of (54) and (60) can be further simplified. Let $\mathbf{h}_N(n)$ denote the summation part of (54); then, we have

$$\tilde{\mathbf{k}}_N(n) = \begin{bmatrix} \tilde{\mathbf{k}}_M(n) \\ \mathbf{0}_{N-M} \end{bmatrix} + \mathbf{h}_N(n) \tag{61}$$

where the recursive equation for $\mathbf{h}_N(n)$ is given by

$$\begin{aligned} \begin{bmatrix} \mathbf{h}_N(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} 0 \\ \mathbf{h}_N(n-1) \end{bmatrix} + \frac{\psi_M(n)}{\lambda B_M(n-1)} \begin{bmatrix} \mathbf{c}_M(n-1) \\ 1 \\ \mathbf{0}_{N-M} \end{bmatrix} \\ &- \frac{\psi_M(n-N+M)}{\lambda B_M(n-N+M-1)} \begin{bmatrix} \mathbf{0}_{N-M} \\ \mathbf{c}_M(n-N+M-1) \\ 1 \end{bmatrix}. \end{aligned} \tag{62}$$

Let $h(n)$ denote the summation part of (60); then, $\gamma_N(n)$ can be written as

$$\frac{1}{\gamma_N(n)} = \frac{1}{\gamma_M(n)} + h(n) \tag{63}$$

where $h(n)$ is given by

$$\begin{aligned} h(n) = h(n-1) &+ \frac{\psi_M^2(n)}{\lambda B_M(n-1)} \\ &- \frac{\psi_M^2(n-N+M)}{\lambda B_M(n-N+M-1)}. \end{aligned} \tag{64}$$

We note that the results of (61) to (64) are equivalent to the Version 3 of the FNTF algorithm. Table I presents a summary of the fast FPLS algorithm and the fast BPLS algorithm.

TABLE II
COMPUTATIONAL COST OF SEVERAL LEAST SQUARES AND FAST NEWTON ALGORITHMS

Algorithm	Computational cost
RLS	$3N^2 + 11N + 8$
LSL	$22N + 4$
PLS	$1.5N^2 + 3.5N + 11$
SFNTF	$6M + 14 + 2N$
Fast PLS	$1.5M^2 + 1.5M + 14 + 2N$

C. Comparison of Computational Complexity

The computational cost of several typical least-squares algorithms, which are addressed in the introduction part, are listed in Table II (the cost includes only multiplications and divisions). From Table II, we can see that the PLS algorithm requires about 50% of the computational cost of the RLS algorithm. The computational cost of the LSL algorithm with error feedback is about $O(22N)$. However, if the tap-weight vector of the transversal filter is necessary, then the backward predictor $\mathbf{c}_m(n)$, which is computed by the least-squares version of the Levinson–Durbin recursions, is required to link the regression coefficients of the lattice filter and the tap weights of the transversal filter, leading to an $O(N^2)$ computational load [19]. The fast PLS algorithm requires $O(M^2) + O(N)$ computations, which is comparable to the SFNTF algorithm when M is small. This is usually satisfied in some applications like acoustic echo cancellation, in which a speech signal is used as the input [20]. Therefore, it may be desirable to use the fast PLS algorithm to achieve a numerically stable performance.

IV. Stability Analysis

The most important characteristics of the PLS algorithm and the fast PLS algorithm are their excellent numerical properties. In this section, we will prove these properties.

The prediction part of the PLS algorithm can be modeled by the following nonlinear state-space form [10]

$$\Theta(n) = f[\Theta(n-1), \mathbf{u}_m(n)] \tag{65}$$

where $\Theta(n)$ and $\mathbf{u}_m(n)$ denote the state-space variables and the tap-input vector, respectively. For a finite-precision implementation, roundoff errors are introduced so that $\hat{\Theta}(n) = \Theta(n) + \Delta\Theta(n)$. Assuming that the errors are small, (65) can be linearized in the presence of the roundoff errors, which leads to

$$\Delta\Theta(n) = \mathbf{A}(n)\Delta\Theta(n-1) + V(n) \tag{66}$$

where $V(n)$ represents the instantaneous contribution of the roundoff noise, and

$$\mathbf{A}(n) = \nabla_\Theta f[\Theta, \mathbf{u}_m(n)]|_{\Theta=\Theta(n-1)}. \tag{67}$$

This is a linear time-variant system with a signal-dependent $\mathbf{A}(n)$ matrix. Therefore, it is difficult to make an exact statement about the deterministic stability. Nevertheless, we can make certain statistical statements when the input signal $u(n)$ is stationary and ergodic. More precisely, it is shown in [10] that the state transition matrix

$$\mathbf{F}(n,0) = \mathbf{A}(n)\mathbf{A}(n-1)\ldots\mathbf{A}(1) \tag{68}$$

has an asymptotic constant eigendecomposition that can be used to decide the numerical stability of the original system. Using the averaging technique, it is shown in [10] that numerical stability of (65) is determined by the eigenvalues of $\lim_{n\to\infty} \mathrm{E}[\mathbf{A}(n)]$.

A. Numerical Properties of the FPLS Algorithm

For the FPLS algorithm, the state-space variables that involve the time-update recursions can be expressed by

$$\Theta_f(n) = \begin{pmatrix} \mathbf{a}_m(n) \\ F_m(n) \end{pmatrix}. \tag{69}$$

Substituting (1) into (5) and noticing that $\mathbf{k}_m(n-1) = \gamma_m(n-1)\tilde{\mathbf{k}}_m(n-1)$, we can write the first state-space variable as

$$\mathbf{a}_m(n) = \left(\mathbf{I}_m - \mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1)\right) \times \mathbf{a}_m(n-1) - u(n)\mathbf{k}_m(n-1) \tag{70}$$

where $\mathbf{I}_m$ is an $m \times m$ identity matrix.

For the second state-space variable, substituting (1) into (2), we get

$$\begin{aligned} F_m(n) &= \lambda F_m(n-1) + \gamma_m(n-1)\left(\mathbf{u}_m^T(n-1)\mathbf{a}_m(n-1)\right)^2 \\ &\quad + 2\gamma_m(n-1)u(n)\mathbf{u}_m^T(n-1)\mathbf{a}_m(n-1) \\ &\quad + \gamma_m(n-1)u^2(n). \end{aligned} \tag{71}$$

It is worth noting from (3) and (4) that the conversion factor $\gamma_m(n-1)$ and the gain vector $\mathbf{k}_m(n-1)$ are not state-space variables because both of them have the relation $\varphi_{m+1}(n) = \varphi_m(n-1)$, which means that the roundoff errors accumulated in a time sequence n will be bounded for a limited order sequence m.

From (67), the transition matrix of the forward predictor $\mathbf{A}_f(n)$ is obtained by differentiating the state-space model (69) with respect to its state-space variables, which results in

$$\mathbf{A}_f(n) = \begin{pmatrix} \frac{\partial(\mathbf{a}_m(n))}{\partial(\mathbf{a}_m(n-1))} & \frac{\partial(\mathbf{a}_m(n))}{\partial(F_m(n-1))} \\ \frac{\partial(F_m(n))}{\partial(\mathbf{a}_m(n-1))} & \frac{\partial(F_m(n))}{\partial(F_m(n-1))} \end{pmatrix} \tag{72}$$

where

$$\frac{\partial(\mathbf{a}_m(n))}{\partial(\mathbf{a}_m(n-1))} = \mathbf{I}_m - \mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1) \tag{73}$$

$$\frac{\partial(\mathbf{a}_m(n))}{\partial(F_m(n-1))} = \mathbf{0}_m \tag{74}$$

$$\begin{aligned} &\frac{\partial(F_m(n))}{\partial(\mathbf{a}_m(n-1))} \\ &= 2\gamma_m(n-1)\mathbf{u}_m^T(n-1)\mathbf{a}_m(n-1)\mathbf{u}_m^T(n-1) \\ &\quad + 2\gamma_m(n-1)u(n)\mathbf{u}_m^T(n-1) \\ &= 2\gamma_m(n-1)\left(\mathbf{u}_m^T(n-1)\mathbf{a}_m(n-1) \right. \\ &\quad \left. + u(n)\right)\mathbf{u}_m^T(n-1) \\ &= 2\gamma_m(n-1)\eta_m(n)\mathbf{u}_m^T(n-1) \end{aligned} \tag{75}$$

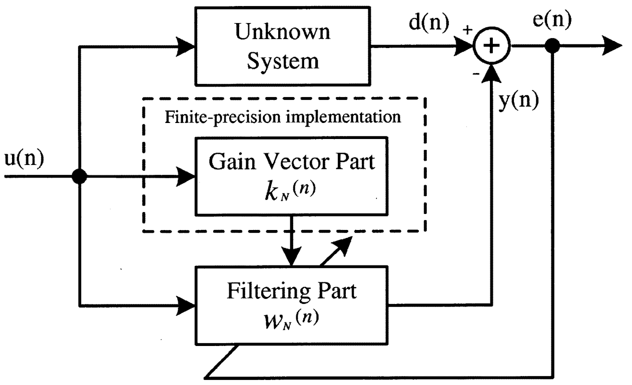


Fig. 1. Block diagram of the system identification used for simulation.

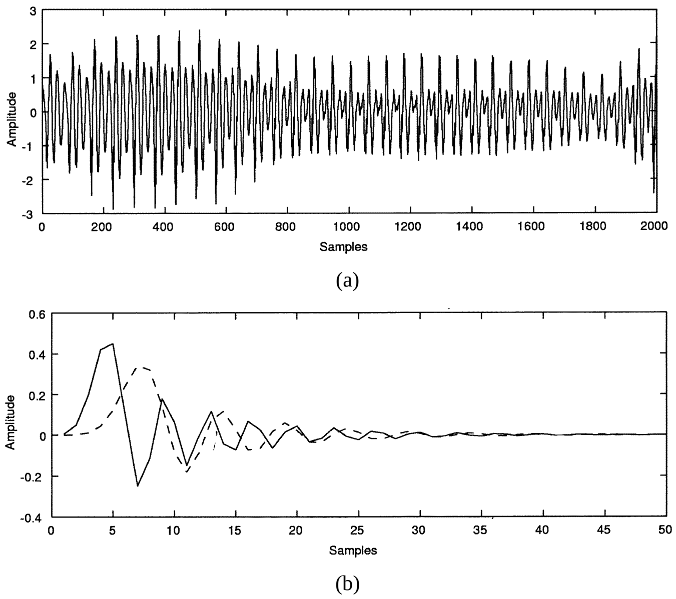


Fig. 2. (a) Input speech signal. (b) Impulse responses of the unknown system before change (solid line) and after change (dashed line).

$$\frac{\partial(F_m(n))}{\partial(F_m(n-1))} = \lambda. \tag{76}$$

Therefore, $\mathbf{A}_f(n)$ can be written as

$$\mathbf{A}_f(n) = \begin{pmatrix} \mathbf{I}_m - \mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1) & \mathbf{0}_m \\ 2\gamma_m(n-1)\eta_m(n)\mathbf{u}_m^T(n-1) & \lambda \end{pmatrix}. \tag{77}$$

Since $0 < \lambda < 1$ and $\mathbf{A}_f(n)$ is a block-lower-triangular, it remains to be shown that the eigenvalues of $\mathrm{E}[\mathbf{I}_m - \mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1)]$ are asymptotically all smaller than unity in magnitude.

For $1 - \lambda \ll 1$, from [19], we can derive

$$\lim_{n\to\infty} \Phi_m(n-1) \approx (1-\lambda)^{-1}\mathbf{R} \tag{78}$$

where $\Phi_m(n-1) = \sum_{i=1}^{n-1} \lambda^{n-i-1}\mathbf{u}_m(i)\mathbf{u}_m^T(i)$, and $\mathbf{R} = \mathrm{E}[\mathbf{u}_m(n-1)\mathbf{u}_m^T(n-1)]$.

Since $\mathbf{k}_m(n-1) = \Phi_m^{-1}(n-1)\mathbf{u}_m(n-1)$, it follows that

$$\lim_{n\to\infty} \mathrm{E}[\mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1)] \approx (1-\lambda)\mathbf{I}_m. \tag{79}$$

Consequently

$$\lim_{n\to\infty} \mathrm{E}[\mathbf{I}_m - \mathbf{k}_m(n-1)\mathbf{u}_m^T(n-1)] \approx \mathbf{I}_m - (1-\lambda)\mathbf{I}_m = \lambda\mathbf{I}_m \tag{80}$$

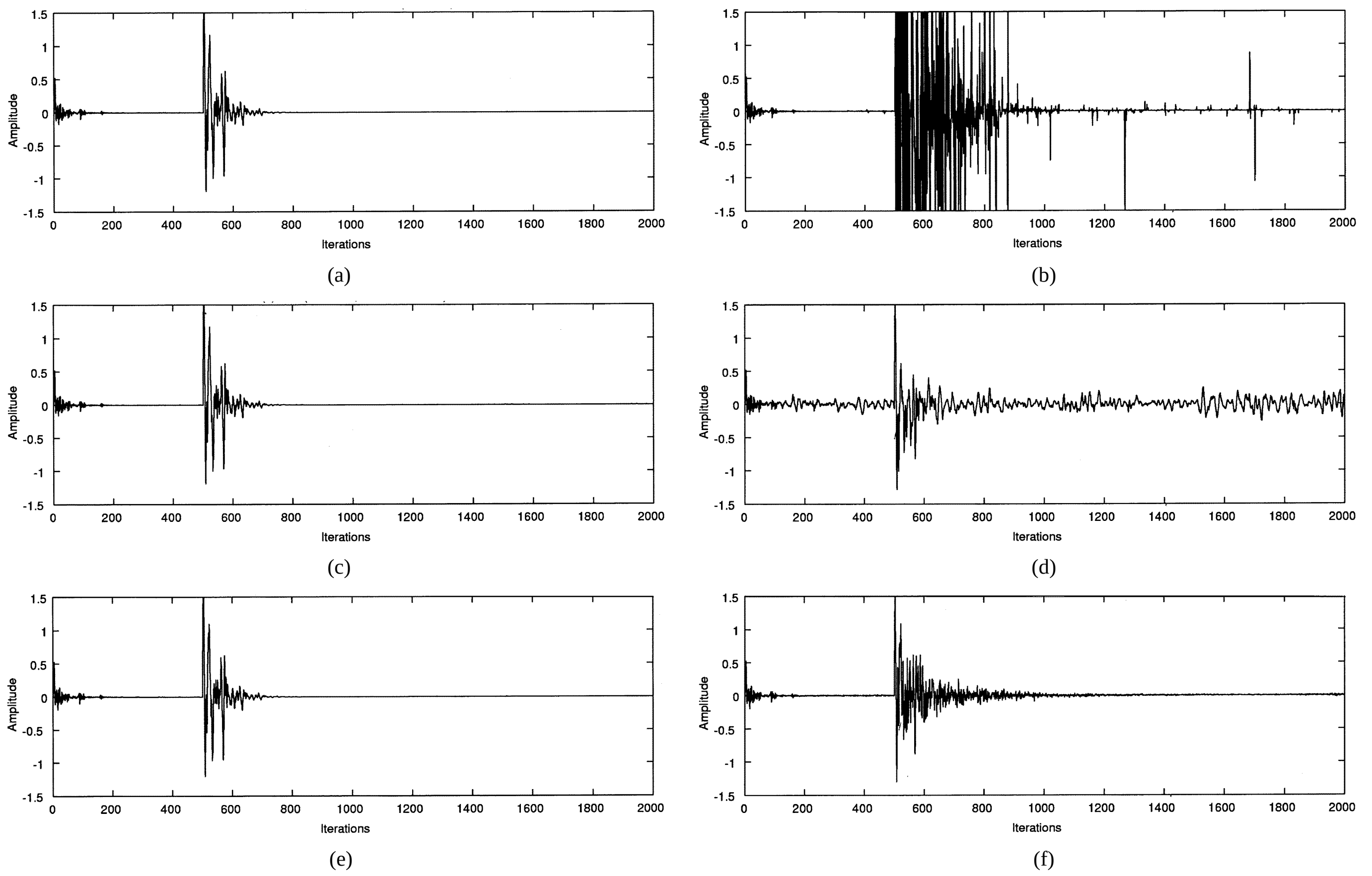


Fig. 3. Numerical performances of the RLS, the LSL, and the BPLS algorithms $(N = M = 50)$. The estimation error $e(n)$ of (a) RLS algorithm using double precision, (b) RLS algorithm using 20-bit mantissa, (c) LSL algorithm using 16-bit mantissa, (d) LSL algorithm using 8-bit mantissa, (e) BPLS algorithm using 16-bit mantissa, (f) the BPLS algorithm using 8-bit mantissa.

which confirms that all the eigenvalues of $\mathrm{E}[\mathbf{A}_f(n)]$ converge approximately to λ as $n \to \infty$.

B. Numerical Properties of the BPLS Algorithm

The state-space variables that involve the time-update recursions in the BPLS algorithm are given by

$$\Theta_b(n) = \begin{pmatrix} \mathbf{c}_m(n) \\ B_m(n) \end{pmatrix}. \tag{81}$$

Substituting (6) into (10) and (7), respectively, and noticing that $\mathbf{k}_m(n) = \gamma_m(n)\tilde{\mathbf{k}}_m(n)$, we have the two state-space variables

$$\begin{aligned} \mathbf{c}_m(n) = &\left(\mathbf{I}_m - \mathbf{k}_m(n)\mathbf{u}_m^T(n)\right)\mathbf{c}_m(n-1) \\ &- u(n-m)\mathbf{k}_m(n) \end{aligned} \tag{82}$$

$$\begin{aligned} B_m(n) = &\lambda B_m(n-1) + \gamma_m(n)\left(\mathbf{u}_m^T(n)\mathbf{c}_m(n-1)\right)^2 \\ &+ 2\gamma_m(n)u(n-m)\mathbf{u}_m^T(n)\mathbf{c}_m(n-1) \\ &+ \gamma_m(n)u^2(n-m). \end{aligned} \tag{83}$$

Notice also from (8) and (9) that the conversion factor $\gamma_m(n)$ and the gain vector $\mathbf{k}_m(n)$ are not state-space variables because they involve only the order-update recursions.

Following the same procedure as that of the FPLS algorithm, we can write the transition matrix of the backward predictor $\mathbf{A}_b(n)$ as

$$\mathbf{A}_b(n) = \begin{pmatrix} \mathbf{I}_m - \mathbf{k}_m(n)\mathbf{u}_m^T(n) & \mathbf{0}_m \\ 2\gamma_m(n)\psi_m(n)\mathbf{u}_m^T(n) & \lambda \end{pmatrix} \tag{84}$$

and prove that all the eigenvalues of $\mathrm{E}[\mathbf{A}_b(n)]$ also converge approximately to λ as $n \to \infty$.

C. Numerical Properties of the Fast PLS Algorithm

The numerical properties of the fast PLS algorithm are closely related to the PLS algorithm. From (39) and (62), we can see that the equations for extending the gain vector has the following common formula:

$$\mathbf{p}_N(n) = \mathbf{Q}\mathbf{p}_N(n-1) + \mathbf{q}_N(n) \tag{85}$$

where $\mathbf{p}_N(n)$ stands for $\mathbf{g}_N(n)$ or $\mathbf{h}_N(n)$. The matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{pmatrix} 0 & & \cdots & & 0 \\ 1 & \ddots & & & \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}. \tag{86}$$

Since all of the eigenvalues of $\mathbf{Q}$ are zeros and $\mathbf{q}_N(n)$ is computed by using either the FPLS or the BPLS algorithm, $\mathbf{p}_N(n)$ is stable. For the same reason, the conversion factor $\gamma_N(n)$ computed by (40) and (41) or (63) and (64) is also stable. Therefore, a numerically stable performance of the extended gain vector $\mathbf{k}_N(n)$ can be expected.

V. Simulation Result

Computer simulations were carried out to confirm the validity of our analysis and demonstrate the numerically stable performances of the PLS and the fast PLS algorithms. Since both the FPLS and the BPLS algorithms have very similar numerical properties, only the BPLS and the fast BPLS algorithms were investigated. An adaptive system identification problem, whose block diagram is shown in Fig. 1, was employed for the simulation. The adaptive filter is divided into two parts. Since we want to investigate the numerical properties of the gain vector, the gain vector part enclosed by the dashed line is computed by using a floating-point arithmetic that consists of an 8-bit exponent and a variable mantissa (including a sign bit). The filtering part is computed by using double-precision arithmetic. Fig. 2(a) shows a speech signal that was used as the tap-input signal. Fig. 2(b) shows two impulse responses of a 12th-order butterworth filter with different cutoff frequencies, which were used as the unknown system before and after a sudden change occurred at the number of 500 iterations. The number of tap weights of the adaptive filter was 50. The initial parameter $\delta = 1$, and the forgetting factor $\lambda = 0.97$ were used. For comparison purposes, the RLS algorithm (shown in [19, p. 569, Tab. 13.1]), the LSL algorithm with error feedback (shown in [19 , p. 687, Tab. 15.5]), and the combination of the stabilized FRLS (with a computational cost of $O(8N)$, as shown in [12 , p. 25, Tab. 7]), and Version 3 of the FNTF (SFNTF) algorithms were also implemented using the same simulation conditions. The optimally adjusted error feedback parameters $\sigma^1 = 1, \sigma^2 = 1.2$, and $\sigma^3 = 1$ were used in the SFRLS algorithm.

Fig. 3 shows the numerical performances of the RLS, the LSL, and the BPLS algorithms. As expected, the numerical performance of the BPLS algorithm is very robust to roundoff errors produced by finite-precision implementation. On the other hand, the numerical property of the RLS algorithm is seriously affected by finite-precision arithmetic. The LSL algorithm also has a well-behaved numerical performance. However, the accuracy of the LSL algorithm seems inferior to that of the BPLS algorithm when a lower wordlength was used.

Fig. 4 shows the numerical performances of the SFNTF and the fast BPLS algorithms. At first glance, no divergence of the SFNTF algorithm occurred under finite-precision implementation. However, if we investigate the numerical behavior of the conversion factor $\gamma_N(n)$, as shown in Fig. 5, we observe that the value of $\gamma_N(n)$ becomes zero after some iterations, which means that the adaptive filter stops adaptation. This fact can be seen clearly from Fig. 4(b). In this case, the SFNTF algorithm has no ability to track the change of the unknown system after 500 iterations. Although the numerical performance of the SFRLS algorithm is found to be improved by increasing the forgetting factor λ close to unity, this rescue method will degrade the tracking performance of the SFRLS algorithm. On the other hand, no such unstable behavior of $\gamma_N(n)$ was observed in the fast BPLS algorithm, which results in a much improved numerical and tracking performance of the fast BPLS algorithm compared with the SFNTF algorithm.

We notice from Figs. 3(f) and 4(d) that the accuracy of the BPLS and the fast BPLS algorithms is decreased when the number of mantissa bits used for implementation is reduced, leading to some deviations of the residual error from the ideal performance. Nevertheless, the numerically stable performances of both the BPLS algorithm and its fast version are unaffected.

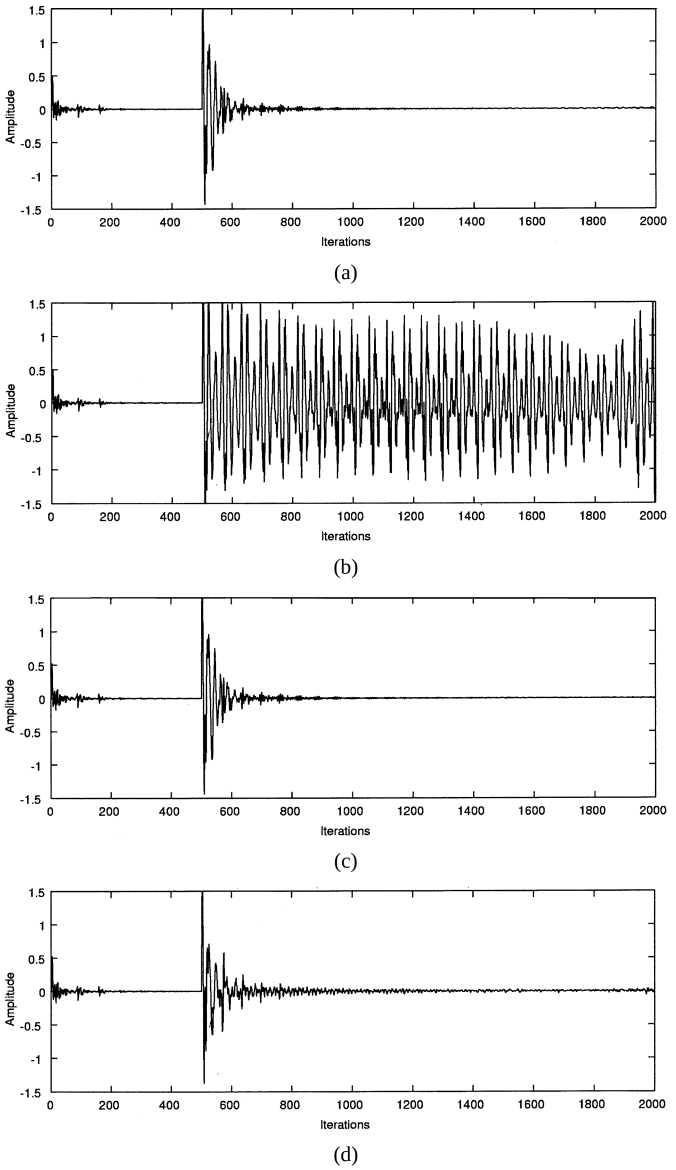


Fig. 4. Numerical performances of the SFNTF and the fast BPLS algorithms $(N = 50, M = 10)$. The estimation error $e(n)$ of (a) the SFNTF algorithm using 24-bit mantissa, (b) SFNTF algorithm using 12-bit mantissa, (c) fast BPLS algorithm using 12-bit mantissa, (d) fast BPLS algorithm using 8-bit mantissa.

VI. Conclusion

A numerically stable fast Newton-type adaptive filter—the fast PLS algorithm—has been proposed. The derivation is based on the order-recursive least-squares algorithm, and the result has shown to be equivalent to the FNTF algorithm. However, the derivation presented in this paper is direct and easier to understand. The numerical properties of the PLS and the fast PLS algorithms have been studied by using the linear time-variant state-space method. The transition matrices of the PLS algorithm and its fast version have been derived, and the eigenvalues of the ensemble average of these transition matrices have been

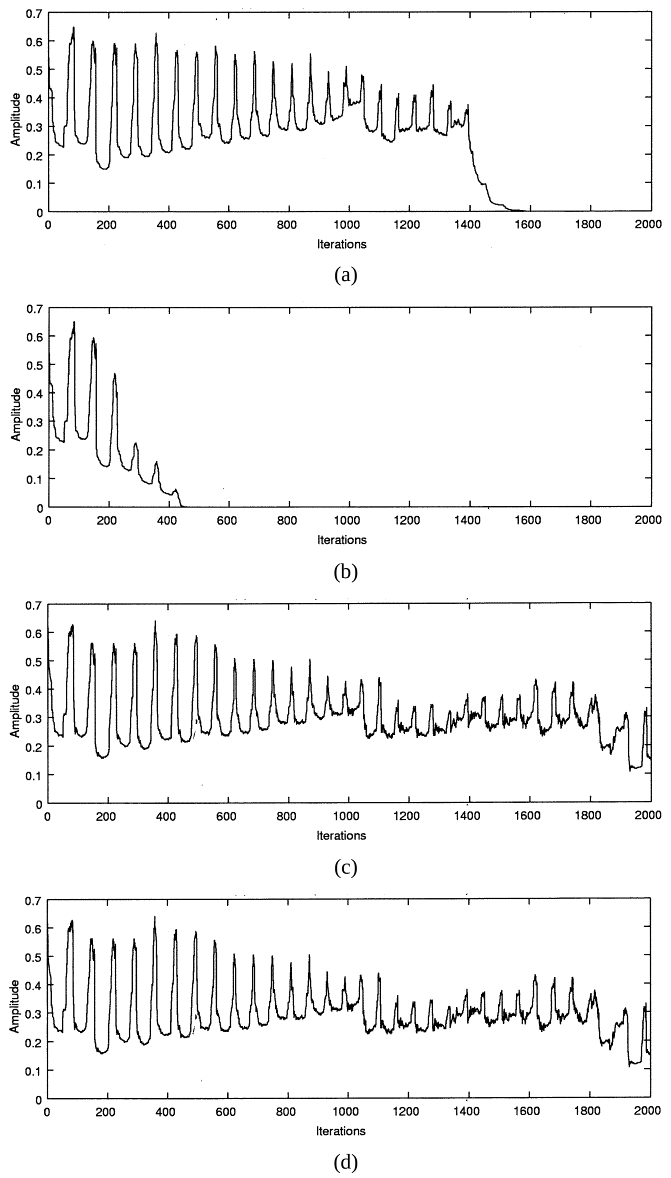


Fig. 5. Numerical performances of the conversion factor $(N = 50, M = 10)$. The conversion factor $\gamma_N(n)$ of (a) SFNTF algorithm using 24-bit mantissa, (b) SFNTF algorithm using 12-bit mantissa, (c) fast BPLS algorithm using 12-bit mantissa, (d) fast BPLS algorithm using 8-bit mantissa.

shown all to be asymptotically equal to λ. As a result, the PLS algorithm and its fast version can provide much-improved numerical performances compared with the RLS algorithm and the SFNTF algorithm. Therefore, the PLS algorithm may be used for replacing the RLS algorithm, and the fast PLS algorithm be applied to various fields, such as within an acoustic echo canceller, to provide a fast convergence rate and a numerically stable performance with less computation.

Acknowledgment

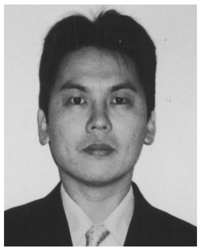
The authors are grateful to Prof. H. Sakai of Kyoto University for precious advice and comments on stability analysis of the PLS algorithm.

References

[1] S. Ljung and L. Ljung, “Error propagation properties of recursive least-squares adaptation algorithms,” *Automatica*, vol. 21, pp. 157–167, 1985.
[2] M. H. Verhaegen and P. V. Dooren, “Numerical aspects of different Kalman filter implementations,” *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 907–917, 1986.
[3] M. H. Verhaegen, “Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes,” *Automatica*, vol. 25, no. 3, pp. 437–444, 1989.
[4] G. E. Bottomley and S. T. Alexander, “A theoretical basis for the divergence of conventional recursive least squares filters,” in *Proc. ICASSP*, 1989, pp. 908–911.
[5] H. Sakai, “Survey on recent adaptive algorithms of RLS type,” *J. Acoust. Soc. Japan*, vol. 48, no. 7, pp. 493–500, 1992.
[6] 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, Feb. 1984.
[7] J. M. Cioffi, “Limited-precision effects in adaptive filtering,” *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 821–833, 1987.
[8] H. Schutze, “Numerical characteristics of fast recursive least squares transversal adaptation algorithms—A comparative study,” *Signal Process.*, vol. 27, pp. 317–331, 1992.
[9] D. W. Lin, “On digital implementation of the fast Kalman algorithm,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 998–1005, May 1984.
[10] D. T. M. Slock and T. Kailath, “Numerically stable fast transversal filters for recursive least squares adaptive filtering,” *IEEE Trans. Signal Processing*, vol. 39, pp. 92–114, Jan. 1991.
[11] J. L. Botto and G. V. Moustakides, “Stabilizing the fast kalman algorithms,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1342–1348, Sept. 1989.
[12] G. Glentis, K. Berberidis, and S. Theodoridis, “Efficient least squares adaptive algorithms for FIR transversal filtering,” *IEEE Signal Processing Mag.*, pp. 13–41, July 1999.
[13] Y. Wang and K. Nakayama, “Numerical performances of recursive least squares and predictor based least squares: A comparative study,” *IEICE Trans. Fundamentals*, vol. E80-A, no. 4, pp. 745–752, 1997.
[14] G. V. Moustakides and S. Therodoridis, “Fast Newton transversal filters—A new class of adaptive estimation algorithms,” *IEEE Trans. Signal Processing*, vol. 39, pp. 2184–2193, Oct. 1991.
[15] Y. Wang, K. Ikeda, and K. Nakayama, “A numerically stable fast Newton type adaptive filter based on order update fast least squares algorithm,” in *Proc. ICASSP*, 1998, pp. 1713–1716.
[16] K. Ikeda, S. Tanaka, and Y. Wang, “Convergence rate analysis of fast predictor-based least squares algorithm,” *IEEE Trans. Circuits Syst.—II*, vol. 49, pp. 11–15, Jan. 2002.
[17] X. Yu and Z. He, “A class of mixed transversal and ladder adaptive filters with pure order updates,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1464–1468, Sept. 1989.
[18] P. Strobach, “Recursive triangular array ladder algorithms,” *IEEE Trans. Signal Processing*, vol. 39, pp. 122–136, Jan. 1991.
[19] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[20] T. Petillion, A. Gillore, and S. Theodoridis, “The fast Newton transversal filter: An efficient scheme for acoustic echo cancellation in mobile radio,” *IEEE Trans. Signal Processing*, vol. 42, pp. 509–518, Mar. 1994.

Youhua Wang (M’96) received the M.S. degree from Shanghai University of Technology, Shanghai, China, in 1988 and the Ph.D. degree from Kanazawa University, Kanazawa, Japan, in 1995.

From 1988 to 1990, he held a teaching and research position at the Department of Electronic Engineering, Shenzhen University, Shenzhen, China. From 1995 to 1996, he worked on speech coding and DSP development technology with the Matsushita Communication Kanazawa R&D Laboratory. From 1996 to 1998, he was with the Department of Electrical and Computer Engineering, Kanazawa University. Since 1998, he has been with Panasonic Mobile Communications Kanazawa R&D Laboratory. His research interests lie in the areas of speech signal processing, adaptive filtering, and their applications to communications.



Kazushi Ikeda (M'96) received the B.E., M.E., and Ph.D. degrees in mathematical engineering and information physics from the University of Tokyo, Tokyo, Japan, in 1989, 1991 and 1994, respectively.

From 1994 to 1998, he was with the Department of Electrical and Computer Engineering, Kanazawa University, Kanazawa, Japan. Since 1998, he has been with the Department of Systems Science, Kyoto University, Kyoto, Japan. His research interests are focused on the fields of adaptive systems, including adaptive filters and neural networks.

Kenji Nakayama (SM'84) received the B.E. and Ph.D. 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 at TIT. He was with NEC Corporation, Kawasaki, Japan, from 1972 to 1988, where his research subjects were filter design methodology and signal processing algorithms. He joined the Department of Electrical and Computer Engineering, Kanazawa University, Kanazawa, Japan, in August 1988, where he is currently a Professor. His current research interests include neural networks and adaptive filters.