An Order-Recursive Lattice Algorithm for $H^\infty$ Adaptive Filtering

Hidenori Matsuzaki

Hikawa Kohbou, 2-253-3 102 Horinouchi-chyo, Ohmiya-ku, Saitama 330-0804, Japan
E-mail: QYQ03604@nifty.ne.jp

Abstract We present an order-recursive lattice algorithm for $H^\infty$ adaptive filtering. The standard $H^\infty$ filter algorithm is expressed in an alternative formula that is closed in a relaxed sense. On the basis of this formula, the algorithm is reformulated as an indefinite least-squares problem that can be solved by a two-dimensional block recursive least-squares (RLS) algorithm. Then an $H^\infty$ lattice filter is derived by transforming the block RLS algorithm into its multivariable least-squares lattice form. The resulting computational complexity is still proportional to the filter order, although $2 \times 2$ matrix computations are involved. The equivalence between the standard $H^\infty$ filter and its lattice version is verified by numerical experiments.

Keywords: adaptive filtering, $H^\infty$ filter, lattice filter

1. Introduction

The objective of $H^\infty$ filtering is to design filters that are robust in the sense that they limit the effect of modeling uncertainties on their performance and stability. In adaptive signal processing, $H^\infty$ filtering theory plays an important role by proving that the celebrated least mean square (LMS) filter is an optimal solution to a well-defined robustness criterion [1], although LMS had been regarded as a rough approximation to a steepest descent method. At the same time, it was shown that the recursive least-squares (RLS) filter can be derived by relaxing the same criterion. Moreover, the $H^\infty$ approach led to a so-called energy-conservation law, which has been extensively used to analyze the robustness of various kinds of adaptive filters (e.g., [2–4]). Thus, it established a unified framework for understanding the fundamental properties of adaptive filters [5, 6]. On the other hand, aside from the theoretical contributions mentioned above, the application of the $H^\infty$ adaptive filter still seems to be restricted compared with that of LMS or RLS, partly owing to its computational complexity. In such a situation, a series of works [7–9] are of practical value because they provided fast algorithms for a type of $H^\infty$ optimization problem.

When implemented in its standard form, an adaptive $H^\infty$ filter requires $O(M^2)$ computations per iteration, where $M$ is the order of the filter. This computational complexity is essentially the same as that of an RLS filter and would be prohibitive in real-time applications. However, for RLS filters, there are a number of fast forms [5] that realize the complexity of $O(M)$ computations per iteration by exploiting the time-shift property of input data. Among these fast forms, a class of order-recursive algorithms known as least-squares lattice (LSL) filters ( [5, 6]) are especially interesting because of their attractive features including a modular structure and numerical stability. Although the LSL filter is a mature topic in signal processing, to the author’s best knowledge, there seems to be no extension of it toward $H^\infty$ filtering. Hence, in this paper, we attempt to derive an order-recursive lattice algorithm for $H^\infty$ filters.

In the method of least squares, its solution is first explicitly represented as a closed-form function of past inputs and then the RLS algorithm is derived by updating the closed-form solution recursively. This closed-form solution plays the central role in deriving LSL filters; it enables one to examine how the solution depends on the filter order and, thereby, to devise an algorithm that is not only time-recursive but order-recursive. In contrast, the $H^\infty$ filter is formulated as an inherently time-recursive algorithm without introducing any closed-form solution (e.g., [10–13]). Hence, to derive an order-recursive algorithm for $H^\infty$ filtering, we must start by finding an alternative expression that substitutes for the role of a closed-form solution. Indeed, once such a “pseudo-closed-form” solution is given, it should be straightforward to obtain an $H^\infty$ lattice algorithm in view of the well-established derivation of LSL filters.

This paper is organized as follows. Section 2 introduces a typical $H^\infty$ adaptive filtering problem and its recursive solution. Section 3 derives a pseudo-closed-form expression for the standard solution, which is used to reformulate $H^\infty$ adaptive filtering as an indefinite least-squares problem. In Sec. 4, the indefinite least-squares problem is solved by employing a two-dimensional block RLS algo-
where \( d \in u_i \) contains measurement noise and modeling errors. The input data vector, \( w_0 \), is estimated, and \((1)\) subscripts denote that of vectors and matrices; \((ii)\) parentheses denote the time dependence of a scalar quantity, whereas subscripts denote that of vectors and matrices; \((iii)\) subscripts are also used for specifying the order of a quantity. \((iv)\) \( ^* \) represents complex conjugation for scalars and the Hermitian transpose for both matrices and vectors.

\[ \text{Algorithm 2} \]

3. Derivation of a Pseudo-Closed-Form Solution

In this section, we derive an alternative expression for the solution of Algorithm 2 that is not recursive but closed in a relaxed sense. For this purpose, we first apply the matrix inversion lemma to Eq. (4) and obtain

\[ P_i^{-1} = \lambda P_{i-1}^{-1} + (1 - \gamma_i^2)w_i^* u_i \]

(5)

Noting that \( P_{i-1} = \Pi > 0 \), we can express \( P_i^{-1} \) in the following closed form:

\[ P_i^{-1} = \lambda^{i+1}\Pi + (1 - \gamma_i^2) \sum_{k=0}^{i} \lambda^{i-k} u_k^* u_k \]

(6)

Let us now assume \( \gamma_i > 1 \). Then, by Eq. (6), \( P_i \) is guaranteed to be positive definite and thus Problem 1 can be solved by Algorithm 2. The assumption \( \gamma_i \in (1, \infty) \) is reasonable because in this range of \( \gamma_i \), an \( H^\infty \) adaptive filter formally includes both LMS and RLS filters as two special cases: indeed, it tends to the LMS as \( \gamma_i \to 1 \), and to the RLS as \( \gamma_i \to \infty \).

We next define another matrix \( \hat{P}_i \in \mathbb{C}^{M \times M} \) by

\[ \hat{P}_i^{-1} = \lambda P_{i-1}^{-1} + u_i^* u_i \]

From Eqs. (5) and (7), we can also write

\[ \hat{P}_i^{-1} = P_i^{-1} + \gamma_i^2 u_i^* u_i \]

(8)

Applying the matrix inversion lemma to Eq. (7), we have

\[ \hat{P}_i u_i = \left( \lambda P_{i-1}^{-1} + u_i^* u_i \right)^{-1} u_i \]

\[ = \left( \frac{\lambda^{-1} P_{i-1}^{-1} u_i^* P_{i-1}}{1 + \lambda^{-1} u_i^* P_{i-1} u_i} \right) u_i \]

\[ = \frac{\lambda^{-1} P_{i-1}^{-1} u_i}{1 + \lambda^{-1} u_i^* P_{i-1} u_i} \]  

(9)

Using Eq. (9), we can rewrite Eq. (3) as

\[ w_i = w_{i-1} + \hat{P}_i u_i \left( d(i) - u_i w_{i-1} \right) \]

(10)

Thus we have obtained another recursion for \( w_i \), involving \( \hat{P}_i \) instead of \( P_i \). Although \( \hat{P}_i \) is only slightly different from \( P_i \), it should be emphasized that the use of \( \hat{P}_i \) is crucial for our purpose, as we shall see in the following derivation.

To proceed, we introduce the following matrices:

\[ U_i = \begin{bmatrix} u_{i-1}^* & u_i \end{bmatrix} \in \mathbb{C}^{2 \times M} \]

(11)

\[ H_i = \begin{bmatrix} U_0^* & U_1^* & \cdots & U_i^* \end{bmatrix} \in \mathbb{C}^{2(i+1) \times M} \]

(12)

\[ R^{-1} = \begin{bmatrix} -\gamma_i^{-2} & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2 \times 2} \]

(13)

\[ W_i = \text{diag} \left( \lambda R^{-1}, \cdots, \lambda R^{-1}, R^{-1} \right) \in \mathbb{R}^{2(i+1) \times 2(i+1)} \]

(14)

\[ z_i = \begin{bmatrix} u_{i-1} w_{i-1} \\ d(i) \end{bmatrix} \in \mathbb{C}^{2 \times 1} \]

(15)

\[ y_i = \begin{bmatrix} z_0^* & z_1^* & \cdots & z_i^* \end{bmatrix} \in \mathbb{C}^{2(i+1) \times 1} \]

(16)
Note that, for \( i \geq 1 \), \( H_i, W_i, \) and \( y_i \) are recursively written as
\[
H_i = \begin{bmatrix} H_{i-1} & U_i \\ U_i^t & 0_{1 \times M} \end{bmatrix}, \quad H_0 = U_0 = \begin{bmatrix} 0_{1 \times M} \\ u_0 \end{bmatrix}
\]
(17)
\[
W_i = \begin{bmatrix} iW_{i-1} & 0 \\ 0 & R^{-1} \end{bmatrix}, \quad W_0 = R^{-1}
\]
(18)
\[
y_i = \begin{bmatrix} y_{i-1} \\ \gamma_i \end{bmatrix}, \quad y_0 = z_0 = \begin{bmatrix} 0 \\ d(0) \end{bmatrix}
\]
(19)

Also note that \( U_i \) has the time-shift property
\[
\begin{bmatrix} U_i \\ u(i - M - 1) \\ u(i - M) \end{bmatrix} = \begin{bmatrix} u(i - 1) \\ u(i) \end{bmatrix} U_{i+1}
\]
(20)

Then we can derive several properties of \( \hat{P}_i \).

**Lemma 3** Assume \( \gamma_f > 1 \), then for all \( i \geq 0 \),
\[
H_i^t W_i H_i = (1 - \gamma_f^{-2}) \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k + \gamma_f^{-2} u_i^t u_i \geq 0
\]
(21)

In particular, for \( \Pi > 0 \),
\[
\lambda^{i+1} \Pi + H_i^t W_i H_i > 0
\]
(22)

**Proof.** From the definitions in Eqs. (11)-(14), we have
\[
H_i^t W_i H_i = \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k - \gamma_f^{-2} \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k
gamma^{-2} \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k
gamma^{-2} \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k
gamma^{-2} \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k
gamma^{-2} \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k
\]

Then, Eq. (22) holds by \( \Pi > 0 \) and \( \lambda > 0 \).

**Lemma 4** Assuming \( \gamma_f > 1 \),
\[
\hat{P}_i = (\lambda^{i+1} \Pi + H_i^t W_i H_i)^{-1} > 0
\]
(23)

Also, \( \hat{P}_i \) can be recursively determined by
\[
\hat{P}_i = \lambda^{i+1} \hat{P}_{i-1} - \lambda^{-2} \hat{P}_{i-1} U_i \Gamma_i U_i^t \hat{P}_{i-1}, \quad \hat{P}_{-1} = \Pi^{-1}
\]
(24)

where \( \Gamma_i \in \mathbb{C}^{2 \times 2} \) is defined by
\[
\Gamma_i := (R + \lambda^{-1} U_i \hat{P}_{i-1} U_i^t)^{-1}
\]
(25)

**Proof.** From Eqs. (6), (8), and (21),
\[
\hat{P}_{i-1} = \lambda^{i+1} \Pi + (1 - \gamma_f^{-2}) \sum_{k=0}^{i} \lambda^{i-k} u_k^t u_k + \gamma_f^{-2} u_i^t u_i
\]
\[
= \lambda^{i+1} \Pi + H_i^t W_i H_i
\]

Thus, we have Eq. (23) and the positive-definiteness follows from Eq. (22). Noting Eqs. (17) and (18), we can rewrite Eq. (23) as
\[
\hat{P}_i = \left[ \lambda \left( \lambda^{i+1} \Pi + H_i^t W_i H_i \right) + U_i^t R^{-1} U_i \right]^{-1}
\]
\[
\left[ \lambda \left( \lambda^{i+1} \Pi + H_i^t W_i H_i \right) + U_i^t R^{-1} U_i \right]^{-1}
\]
\[

Then, applying the matrix inversion lemma to Eq. (26) and using the definition in Eq. (25), we obtain the recursion in Eq. (24).

**Proposition 5** Assuming \( \gamma_f > 1 \), the estimate \( w_i \) computed by the \( H^\infty \) adaptive filter Algorithm 2 can be written as
\[
w_i = \hat{P}_i H_i^t W_i y_i
\]
(27)

**Proof.** Recall \( w_{-1} = 0_{M \times 1} \). Then using Eq. (10), we have
\[
w_0 = 0_{M \times 1} + \hat{P}_0 u_0^t [d(0) - u_0 w_{-1}] = \hat{P}_0 u_0^t d(0)
\]

On the other hand, by Eqs. (17), (18), and \( u_{-1} = 0_{1 \times M} \),
\[
\hat{P}_0 H_0^t W_0 y_0 = \hat{P}_0 u_0^t R^{-1} y_0
\]
\[
\hat{P}_0 H_0^t W_0 y_0 = \hat{P}_0 u_0^t R^{-1} y_0
\]
\[
\hat{P}_0 H_0^t W_0 y_0 = \hat{P}_0 u_0^t R^{-1} y_0
\]
\[
\hat{P}_0 H_0^t W_0 y_0 = \hat{P}_0 u_0^t R^{-1} y_0
\]
\[
\hat{P}_0 H_0^t W_0 y_0 = \hat{P}_0 u_0^t R^{-1} y_0
\]

Thus, Eq. (27) holds for \( i = 0 \). We next assume
\[
w_{i-1} = \hat{P}_{i-1} H_{i-1}^t W_{i-1} y_{i-1}
\]
(28)

for some \( i \geq 1 \). Note that we can write
\[
u_i^t [d(i) - u_i w_{i-1}] = U_i^t R^{-1} \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]
(29)

Substituting Eqs. (24) and (29) into Eq. (10) yields
\[
w_i = w_{i-1} + \lambda^{-1} \hat{P}_{i-1} U_i^t \Gamma_i \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]
(30)

Moreover, by Eqs. (17) - (19),
\[
H_i^t W_i y_i = \lambda H_{i-1}^t W_{i-1} y_{i-1} + U_i^t R^{-1} z_i
\]
(31)

Using Eqs. (24), (25), (28), (30), and (31),
\[
\hat{P}_i H_i^t W_i y_i = \lambda H_{i-1}^t W_{i-1} y_{i-1} + U_i^t R^{-1} z_i
\]
\[
= \lambda^{-1} \hat{P}_{i-1} U_i^t \Gamma_i \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]
\[
= \lambda^{-1} \hat{P}_{i-1} U_i^t \Gamma_i \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]
\[
= \lambda^{-1} \hat{P}_{i-1} U_i^t \Gamma_i \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]
\[
= \lambda^{-1} \hat{P}_{i-1} U_i^t \Gamma_i \begin{bmatrix} d(i) - u_i w_{i-1} \end{bmatrix}
\]

Hence, by induction, Eq. (27) holds for all \( i \geq 0 \).
Although Eq. (27) appears to be a closed-form expression for \( w_i \), it is actually not; the past estimates \( \{w_i\}_{i=1}^{i-1} \) are incorporated in \( y_i \). For this reason, we refer to Eq. (27) as a "pseudo-closed-form" solution. Nonetheless, it enables us to reformulate \( H^\infty \) filtering as an indefinite least-squares problem, as stated in the following theorem.

**Theorem 6** Consider the indefinite least-squares problem

\[
\min_{w \in \mathbb{C}^M} \left\{ x^* w \Pi w + (y_i - H_i w_i)^* W_i (y_i - H_i w_i) \right\}
\]

(32)

where \( H_i, W_i, \) and \( y_i \) are defined by Eqs. (12), (14), and (16), respectively. Also, assume \( \gamma_i > 1 \). Then the minimizing solution of Eq. (32) exists and coincides with the estimate \( w_i \) that results from the \( H^\infty \) adaptive filter of Algorithm 2 at time instant \( i \geq 0 \).

**Proof.** Since \( w_i \) is indefinite, Eq. (32) is not solvable in general. However, in our case, \( x^* \Pi w + (y_i - H_i w_i)^* W_i (y_i - H_i w_i) > 0 \) holds by Lemma 3. Consequently, the minimizing solution exists and can be written as Eq. (27) by setting \( \tilde{P}_i = x^* \Pi + H_i^* W_i H_i \) (cf., [5, p.709, thm. 44.1]). Hence, the theorem holds by Proposition 5.

\[ \square \]

4. **Order-Recursive \( H^\infty \) Adaptive Filter**

Although Eq. (32) is an indefinite least-squares problem, we can treat it as if it were a standard least-squares problem because \( x^* \Pi w + (y_i - H_i w_i)^* W_i (y_i - H_i w_i) > 0 \) is guaranteed by Lemma 3. However, we should note that the quantities incor-porated into \( H_i \) and \( y_i \) are two-dimensional vector \( z_i \) and a \( 2 \times M \) matrix \( U_i \). This type of multivariable problem can be solved recursively by employing the block RLS algorithm (cf., [5, p.552]) as summarized below.

**Algorithm 7** (\( H^\infty \) block adaptive filter)

Let \( U_i, R, \) and \( z_i \) be the quantities defined by Eqs. (11), (13), and (15), respectively. Then, given \( \gamma_i > 1, \lambda \in [0, 1], \) and \( \Pi > 0 \), the estimate \( w_i \) of the \( H^\infty \) adaptive filter can be equivalently computed as follows. Start with \( w_{-1} = 0_{M \times 1}, P_{-1} = \Pi^{-1}, \) and \( \xi(-1) = 0 \). Then iterate for \( i \geq 0 \)

\[
\begin{align*}
\Gamma_i &= \left( R + \lambda^{-1} U_i P_{i-1} U_i^* \right)^{-1} \\
e_i &= z_i - U_i w_{i-1} \\
w_i &= w_{i-1} + \lambda^{-1} P_{i-1} U_i^* \Gamma_i e_i \\
P_i &= \lambda^{-1} P_{i-1} - \lambda^{-2} P_{i-1} U_i^* \Gamma_i U_i P_{i-1} \\
r_i &= z_i - U_i w_i \\
\xi(i) &= \lambda \xi(i-1) + e_i^* \Gamma_i e_i
\end{align*}
\]

where \( e_i \in \mathbb{C}^{2 \times 1} \) represents the a priori estimation error vector, \( r_i \in \mathbb{C}^{2 \times 1} \) the a posteriori estimation error vector, and \( \xi(i) > 0 \) the resulting minimum cost of Eq. (32).

In Algorithm 7, the a priori estimation error vector represents

\[
\begin{bmatrix}
e_i & 0 & U_i w_{i-1}
\end{bmatrix} = \begin{bmatrix} 0 & d(i) - u_i \end{bmatrix} = \begin{bmatrix} 0 & e(i) \end{bmatrix}
\]

(33)

Hence, the actual a priori estimation error \( e(i) \) corresponds to the second component of \( e_i \). Similarly, the actual a posteriori estimation error \( r(i) \) resides in the second component of \( r_i \). To summarize,

\[
\begin{bmatrix}
e(i) & r(i)
\end{bmatrix} = \begin{bmatrix} 0 & 1 & e_i \end{bmatrix}, \quad r(i) = \begin{bmatrix} 0 & 1 & r_i \end{bmatrix}
\]

(34)

Also, we can immediately show

\[ r_i = R \Gamma_i e_i \]

(35)

Thus \( R \Gamma_i \) can be interpreted as the conversion factor.

In the case of the scalar-valued RLS algorithm, its order-recursive form is the well-known LSL filter. Therefore, in view of the derivation of the LSL filter (cf., [5,14]), we can transform Algorithm 7 into its multivariable lattice form almost by inspection. Although the lattice algorithm can be implemented in various forms, as a practical example, we present the a priori error-feedback form of Algorithm 7, which tends to exhibit good performance under a finite-precision condition.

**Algorithm 8** (A priori error-feedback \( H^\infty \) lattice filter)

Suppose \( \gamma_i > 1, \lambda \in [0, 1], \eta > 0 \) and define

\[ \Pi = \eta^{-1} \text{diag}(\lambda^{-2}, \lambda^{-3}, \cdots, \lambda^{-M+2}) > 0 \]

Then, the a priori estimation error \( e(i) = d(i) - u_i w_{i-1} \) that results from the \( H^\infty \) adaptive filter Algorithm 2 can be equivalently computed as follows.

1. Initialization. From \( m = 0 \) to \( m = M - 1 \), set

\[
\kappa_m = R^{-1}, \quad \beta_m = 0_{2 \times 1}, \quad \zeta_m = \eta^{-1} \lambda^{-m-2}
\]

2. For \( i \geq 0 \), repeat

(a) Set

\[
\begin{bmatrix}
\alpha_{i-1} & \beta_{i-1} & u_i \\
\end{bmatrix} = \begin{bmatrix} u(i-1) \\
\end{bmatrix}, \quad \begin{bmatrix}
\alpha_i & \beta_i & d(i) \\
\end{bmatrix} = \begin{bmatrix} d(i-1) - r(i-1) \\
\end{bmatrix}
\]

(b) From \( m = 0 \) to \( m = M - 1 \), repeat

\[
\begin{align*}
\zeta_m(i) &= \lambda \zeta_m(i-1) + \alpha_m^* \Gamma_m r_{m-1} \\
\rho_m^2(i) &= \lambda \rho_m^2(i-1) + \beta_m^2 r_{m-1} \\
\beta_{m+1}^2(i) &= \beta_{m-1}^2 + \rho_m^2(i-1) \alpha_m \alpha_m \\
\alpha_{m+1} \beta_m^2 &= \alpha_m \beta_m = \kappa_m(i-1) \beta_{m-1} \beta_{m-1} \\
e_{m+1} &= \kappa_m(i-1) \beta_{m-1} \\
k_m &= \kappa_m(i-1) + \beta_m^2 r_{m-1} + \rho_m^2(i) \\
\rho_m^2 &= \rho_m^2(i) + \beta_m^2 r_{m-1} + \rho_m^2(i) \\
\Gamma_{m+1} &= \Gamma_m - \kappa_m(i-1) \beta_{m-1} \rho_m^2(i) / \rho_m^2(i)
\end{align*}
\]
Table 1 Variables used in Algorithm 8

| Variable | Size | Description |
|----------|------|-------------|
| $c_{M}^{(i)}$ | $\mathbb{R} > 0$ | sum of forward prediction error squares |
| $s_{M}^{(i)}$ | $\mathbb{R} > 0$ | sum of backward prediction error squares |
| $\alpha_{M}$ | $C^{2 \times 1}$ | forward prediction a priori error vector |
| $\beta_{M}$ | $C^{2 \times 1}$ | backward prediction a priori error vector |
| $e_{M}$ | $C$ | joint process estimation a priori error vector |
| $s_{M}$ | $C$ | joint process estimation reflection coefficient |
| $R_{M}$ | $C^{2 \times 2}$ | conversion factor matrix |
| $e(i)$ | $C$ | a posteriori estimation error |
| $r(i)$ | $C$ | a posteriori estimation error |

(2-c) Then the outputs are obtained by

$$ e(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} e_{M,1}, \quad r(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} R_{M,1} e_{M,1} $$

Proof. An outline is given in Appendix A.

Table 2 Estimated computational complexity

| Algorithm | $x$ | $+$ | $\times$ |
|-----------|-----|-----|--------|
| the a priori error-feedback LSL | 12M | 0 M | 4M |
| Algorithm 8 | 42M+2 | 31M+2 | 7M |

Fig. 1 Signal flow graph of Algorithm 8

8 is still proportional to the filter order $M$, whereas it is nearly four times as much as that of the LSL version.

5. Numerical Examples

To verify the equivalence between the standard $H^\infty$ filter Algorithm 2 and its lattice form Algorithm 8, we perform a simulation where the two algorithms attempt to estimate $w^0$ in the signal model of Eq. (1) with parameters $M = 10$, $\lambda = 0.996$, $\eta = 10^2$, and variance of measurement noise $\sigma_w^2 = 10^{-8}$. The resulting ensemble-average (50 runs) mean-square errors are plotted in Fig. 2 for $\gamma_f = 1.001$, 1.01, and 1.2. The curves generated by Algorithm 8 are indistinguishable from those of Algorithm 2 for each value of $\gamma_f$, and thus we can see only three curves in Fig. 2 as expected.

We next compare the performance of Algorithm 8 with those of RLS and NLMS in a nonstationary environment, which obeys the first-order random-walk model

$$ w_i^0 = w_{i-1}^0 + q_i (i \geq 0), \quad w_0^0 = 0_{M \times 1} $$

(36)

where $q_i \in \mathbb{C}^{M \times 1}$ denotes a zero-mean random vector whose variance is given by $\sigma_q^2 M$. Figure 3 plots the ensemble-average (100 runs) mean-square errors in the above environment with $M = 10$, $\gamma_f = 1.01$, $\lambda = 0.996$, $\eta = 10^2$, $\sigma_w^2 = 10^{-12}$, and $\sigma_q^2 = 10^{-3}$. The step size of NLMS is set at $\mu = 1.0$. In Fig. 3, we find that both RLS and Algorithm 8 converge about three times faster than NLMS. After about 40 iterations, RLS starts diverging, while NLMS and Algorithm 8 achieve constant mean-square errors. This result indicates, as a whole, the superiority of Algorithm 8 over RLS and NLMS in nonstationary environments.

Figure 4 plots ensemble-averaged (100 runs) mean-square errors of Algorithm 8 for $\gamma_f = 1.01$, 1.1, and 1.5 with the same settings as Fig. 3. In Fig. 4, we see that Algorithm 8 becomes less sensitive to the estimation error and thus tends to diverge as $\gamma_f$ increases. This result reflects the fact that $\gamma_f$ represents a trade-off between the
convergence rate and the sensitivity to the estimation error [15]. Hence, a small $\gamma_f$ should be chosen, particularly in a nonstationary environment.

Lastly, we compare the behavior of Algorithm 8 with those of traditional LSL filters; the a priori error feedback LSL and the generalized sliding window RLS lattice (GSWRLSL) [16]. In this experiment, the parameters are set to $M = 20$, $\lambda = 0.996$, and $\gamma_0^2 = 10^{-6}$. Also, an impulsive noise of unit amplitude is applied to the input sequence at time instant $i = 200$. In Fig. 5, until the impulsive noise occurs, the a priori error feedback LSL exhibits the fastest convergence rate. Algorithm 8 shows almost the same convergence rate as the LSL, although its steady-state error increases as $\gamma_f$ decreases. The GSWRLSL with window size $L = 25$ shows the slowest convergence rate and largest steady-state error in this region. After $i = 200$, the convergence rates of the LSL and GSWRLSL are markedly degraded, whereas the GSWRLSL recovers faster than the LSL. On the other hand, Algorithm 8 preserves a faster convergence rate than the other two algorithms.

**Discussion:** Since the performance of a lattice filter reduces to its fixed-order original form, let us examine the differences among the lattice filters seen in Fig. 5 in view of their fixed-order standard algorithms.

In $H^\infty$ filtering, matrix $P_i$ defined by Eq. (4) can be interpreted as the estimation error covariance matrix, and its inverse, $P_i^{-1}$, corresponds to the information matrix, which is expected to increase as estimation proceeds. From Eq. (5), we see that $P_i^{-1}$ is actually monotonically increasing because $1 - \gamma_f^2 > 0$ is assumed. For convenience, we rewrite Eq. (5) as

$$P_{Hinf,i}^{-1} = \lambda P_{Hinf,i-1}^{-1} + (1 - \gamma_f^2)u_i^*u_i$$  \hspace{1cm} (37)

where $P_i$ is replaced by $P_{Hinf,i}$ so that it can be explicitly distinguished from that of the RLS filter, which satisfies

$$P_{RLS,j}^{-1} = \lambda P_{RLS,j-1}^{-1} + u_i^*u_i$$  \hspace{1cm} (38)

Comparing Eqs. (37) and (38), we immediately have $P_{Hinf,i}^{-1} \leq P_{RLS,j}^{-1}$ or equivalently, $P_{Hinf,i} \geq P_{RLS,j}$, which holds at each time instant for the same input $\{u_i\}$ and initial value. Recalling that the estimation gain is approximately proportional to the estimation error covariance, the inequality indicates that the $H^\infty$ filter maintains a larger gain than the RLS filter and, hence, can respond to disturbances in an extended time span, especially when we choose $\gamma_f \approx 1$.

In contrast, the information matrix $P_{inf,i}^{-1}$ of the sliding window algorithm is not necessarily monotonically increasing. Indeed, letting $L$ denote the window size, $P_{SW,j}$ obeys the following two-staged update [16].

$$\left(P_{SW,j-1}^d\right)^{-1} = P_{SW,j-1}^{-1} - \eta_0 \lambda L^{-1} u_{j-L}^*u_{j-L}$$  \hspace{1cm} (39)

$$P_{SW,j}^{-1} = \lambda \left(P_{SW,j-1}^d\right)^{-1} + u_i^*u_i$$  \hspace{1cm} (40)

where $\eta_0 \in (0, 1]$ and $P_{SW,j}^d$ is an intermediate matrix quantity that represents the so-called down-dating operation. Owing to the subtraction on the right-hand side of

**Fig. 3 Comparison of RLS, NLMS and Algorithm 8 in a nonstationary environment**

**Fig. 4 Effect of $\gamma_f$ in a nonstationary environment**

Eq. (39), $P_{SW,j}^{-1}$ remains moderately small and thus results in an overall large gain. However, note that such a large gain is obtained by ignoring (or giving much less weight to) the inputs outside the window at the expense of increased steady-state mean-square errors, which could be even larger than in the case of $H^\infty$ filtering.

6. **Conclusion**

We derived a pseudo-closed-form expression for the standard $H^\infty$ adaptive filter, which allowed us to implement the filter in the form of a two-dimensional block RLS algorithm. By transforming this block RLS filter into its multivariable LSL form, an $H^\infty$ lattice algorithm was obtained. Several numerical experiments verified the equivalence between the standard $H^\infty$ filter and the derived lattice version, along with the advantage of $H^\infty$ filtering in nonstationary and time-varying environments. However, since the $H^\infty$ lattice algorithm involves $2 \times 2$ matrices, its computational cost turned out to be nearly four times larger than that of the LSL filter. Also, the numerical reliability of the $H^\infty$ lattice filter under finite-precision conditions
still must be evaluated, although floating point arithmetic units are usually available even in embedded systems today. In this context, as a future work, the $H^\infty$ lattice algorithm should be implemented in an array form based on QR decompositions or some other method.

Fig. 5 Lattice filters in a suddenly time-varying environment

References

[1] B. Hassibi, A. H. Sayed and T. Kailath: $H^\infty$ optimality of the LMS algorithm, IEEE Trans. Signal Process., Vol.44, No.2, pp.267-280, 1996.

[2] A. H. Sayed and M. Rupp: Robustness Issues in Adaptive Filtering, DSP Handbook, CRC Press, 1998.

[3] A. H. Sayed: Adaptive Filters, John Wiley Sons, 2008.

[4] S. Haykin: Adaptive Filter Theory, 5th edition, Pearson, 2013.

[5] K. Nishiyama: An $H_\infty$ optimization and its fast algorithm for time-variant system identification, IEEE Trans. Signal Process., Vol.52, No.5, pp.1335-1342, 2004.

[6] T. Katsumata, K. Nishiyama, and K. Sato: Error analysis and numerical stabilization of the fast $H_\infty$ filter, IEICE Trans. Fundam., Vol.E93-A, No.6, pp.1153-1162, 2010.

[7] K. Nishiyama: Algorithm understanding of the J-Fast $H_\infty$ filter based on linear prediction of input signal, IEICE Trans. Fundam., Vol.E95-A, No.7, pp.1175-1179, 2012.

[8] B. Hassibi, A. H. Sayed and T. Kailath: Linear estimation in Krein spaces - Part1: Theory & Part2: Applications, IEEE Trans. Autom. Control, Vol.41, No.1, pp.18-33 & 34-49, 1996.

[9] M. Fujita, A. Maruyama, T. Taniguchi, and K. Uchida: Finite horizon discrete-time $H_\infty$ filter with application to an active vision system, Proc. 32nd IEEE Conference on Decision and Control, pp.2194-2196, 1993.

[10] X. Shen and L. Deng: Game theory approach to discrete $H^\infty$ filter design, IEEE Trans. Signal Process., pp.1092-1095, 1997.

Appendix A Outline of Derivation of Algorithm 8

In view of the derivation of LSL filters [5, 14], we can transform the block RLS algorithm Algorithm 7 into its lattice form Algorithm 8 by inspection. This appendix provides an outline of this procedure. Since the lattice algorithm involves many variables, we focus on several relations among them. Given these examples, other relations should be readily verified in the same manner.

Let us begin by recalling the a priori error-feedback algorithm [5, p.680] that repeats the following computations from $m = 0$ to $m = M$ at each time instant $i \geq 0$:

$$
\hat{e}_m^f(i) = \lambda \hat{e}_m^f(i-1) + \alpha_m(i) \gamma_m(i-1) - \beta_m(i+1) \gamma_m(i)
$$

$$
\hat{e}_m^b(i) = \lambda \hat{e}_m^b(i-1) + \beta_m(i) \gamma_m(i)
$$

$$
\beta_{m+1} = \beta_m(i) - \hat{e}_m(i) \gamma_m(i)
$$

$$
\alpha_{m+1} = \alpha_m(i) - \kappa_m(i-1) \beta_m(i)
$$

$$
\kappa_m(i) = \kappa_m(i-1) + \beta_m(i+1) \gamma_m(i)\beta_{m+1}(i)
$$

$$
\kappa_m(i) = \kappa_m(i-1) + \alpha_m(i) \gamma_m(i) \beta_{m+1}(i)
$$

$$
\gamma_{m+1} = \gamma_m(i) - \gamma_m(i) \beta_m(i) \gamma_m(i) - \beta_m(i) \gamma_m(i)
$$

where all the variables are scalars. Our objective is to transform Algorithm 7 into its lattice form in view of the above relations.

Appendix A1 Order-update relations

In the block RLS algorithm Algorithm 7, data matrix $H_i$ and reference data vector $y_i$ are augmented in such a way that they contain two elements at each time instant, as defined in Eqs. (11), (12), (15), and (16). Except for this temporal augmentation in $H_i$ and $y_i$, Algorithm 7 preserves the fundamental structure of the RLS algorithm. Thus the order-update relations of the LSL algorithm still hold for its block RLS version, whereas some scalar quantities must be replaced with vectors or matrices. Hence, for example, we can immediately transform the LSL relation Eq. (42) into its block RLS counterpart

$$
\hat{e}_{m+1,i} = \hat{e}_{m,i} - \kappa_m(i-1) \beta_{m,i}
$$
where \( e_{m,i+1}, e_{m,i}, \) and \( \beta_{m,i} \) are now two-dimensional vectors. Note that the definition of \( k_m(i) \) remains unchanged because it does not depend on the temporal augmentation [5, ch. 32].

As another example, we examine the order-update relation for \( \Gamma_{m,i} \). In so doing, we first consider the order update of \( U_i \in \mathbb{C}^{2 \times M} \) defined by Eq. (11). Namely, for \( m = 0, 1, \ldots, M - 1 \), we construct \( U_{m,i} \in \mathbb{C}^{2 \times m} \) by extracting the \( m \) leftmost columns of \( U_i \) so that \( U_{m+1,i} \in \mathbb{C}^{2 \times (m+1)} \) can be recursively given by

\[
U_{m+1,i} = \begin{bmatrix} U_{m,i} & \bar{z}_{m,i} \end{bmatrix}
\]

where

\[
\bar{z}_{m,i} = \begin{bmatrix} u(i-m) \\ u(i-m+1) \end{bmatrix}
\]

We next define \( H_{m,i} \in \mathbb{C}^{2(i+1) \times m} \) and \( z_{m,i} \in \mathbb{C}^{2(i+1) \times 1} \) by

\[
H_{m,i} = \begin{bmatrix} U_{m,0} \\ \vdots \\ U_{m,i} \end{bmatrix}, \quad z_{m,i} = \begin{bmatrix} \bar{z}_{m,0} \\ \vdots \\ \bar{z}_{m,i} \end{bmatrix}
\]

Hence, \( H_{m+1,i} \in \mathbb{C}^{2(i+1) \times (m+1)} \) can be order-updated as

\[
H_{m+1,i} = \begin{bmatrix} H_{m,i} & z_{m,i} \end{bmatrix}
\]

We also define

\[
P_{m,i} = (x^{i+1} \Pi_m + H_{m,i}^* W_i H_{m,i})^{-1}
\]

\[
\Pi_m = \eta^{-1} \text{diag}(\lambda^{-1}, \lambda^{-2}, \ldots, \lambda^{-(m+1)})
\]

Then, according to [5, ch. 32],

\[
P_{m+1,i} = \begin{bmatrix} P_{m,i} & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
+ \frac{1}{\eta^{-i} x^{i-m} + \epsilon_{m,i}^h(i)} \begin{bmatrix} -w_{m,i} \\ -w_{m,i} \end{bmatrix}^*
\]

where \( w_{m,i} \in \mathbb{C}^{m} \) is defined by

\[
w_{m,i} = \arg \min_{w \in \mathbb{C}^{m}} \left\{ x^{i+1} \|w\|_W^2 + \|z_{m,i} - H_{m,i} w\|_W^2 \right\}
\]

and \( \epsilon_{m,i}^h(i) \) is the corresponding minimum value of Eq. (52).

Note that the resulting a posteriori backward prediction error can be written as

\[
b_{m,i} = \text{diag} (0, \cdots, 0, 1, 1) \left( z_{m,i} - H_{m,i} w_{m,i} \right)
\]

\[
= \bar{z}_{m,i} - U_{m,i} w_{m,i}
\]

Now recalling the block RLS Algorithm 7 and using the matrix inversion lemma, we obtain

\[
\Gamma_{m+1,i} = \left( R + x^{i+1} U_{m,i} P_{m+1,i} U_{m+1,i}^* \right)^{-1}
\]

\[
= R^{-1} - R^{-1} U_{m+1,i} P_{m+1,i} U_{m+1,i}^* R^{-1}
\]

In view of Eqs. (51) and (54), we can rewrite Eq. (55) as

\[
\Gamma_{m+1} = \Gamma_{m} - R^{-1} b_{m,i} (R^{-1} b_{m,i})^*/\epsilon_{m,i}^h(i)
\]

where \( \epsilon_{m,i}^h(i) \) is defined by

\[
\epsilon_{m,i}^h(i) = \eta^{-1} x^{i-m} + \epsilon_{m,i}^h(i)
\]

Substituting \( \beta_{m,i} = (R \Gamma_{m,i})^{-1} b_{m,i} \) into Eq. (56), we finally have

\[
\Gamma_{m+1,i} = \Gamma_{m} - \Gamma_{m} (\Gamma_{m} \beta_{m,i})^*/\epsilon_{m,i}^h(i)
\]

**Appendix A2** Time-update relations

In addition to order-update operations, a lattice filter needs to time-update quantities defined by inner products. To derive a general formula for such a time-update operation, we first consider a generic matrix of the form

\[
\begin{bmatrix} x^* \bar{H} & z \end{bmatrix}
\]

where \( x, z \in \mathbb{C}^{k \times 1} \) and \( \bar{H} \in \mathbb{C}^{k \times q} \). Also, consider a vector \( w_z \in \mathbb{C}^{k \times 1} \) defined by

\[
w_z = \arg \min_{w \in \mathbb{C}^k} \left\{ x^* \|w\|_U^2 + \|z - \bar{H} w\|_W^2 \right\}
\]

where \( \Pi > 0 \) and \( W = \text{diag}(1, 1, \cdots, 1) \). Then define the weighted inner product

\[
\Delta = x^* W \bar{z} \quad \text{where} \quad \bar{z} = z - \bar{H} w_z
\]

Now suppose that a \( p \times (q + 2) \) matrix is appended to the bottom of Eq. (59) as shown below

\[
\begin{bmatrix} x^* \bar{H} & z \end{bmatrix} =: \begin{bmatrix} x_1^* \bar{H}_1 & z_1 \end{bmatrix}
\]

where \( a, b \in \mathbb{C}^{p \times 1} \) and \( \bar{U} \in \mathbb{C}^{p \times q} \). We also consider \( w_{z_1} \) and \( w_{x_1} \) defined by

\[
w_{z_1} = \arg \min_{w \in \mathbb{C}^p} \left\{ x_1^* \|w\|_{U_1}^2 + \|z_1 - \bar{H}_1 w\|_{W_1}^2 \right\}
\]

\[
w_{x_1} = \arg \min_{w \in \mathbb{C}^p} \left\{ x_1^* \|w\|_{U_1}^2 + \|x_1 - \bar{H}_1 w\|_{W_1}^2 \right\}
\]

where \( W_1 := \text{diag}(1, W, R^{-1}) \). Then, we can define another weighted inner product that corresponds to the matrix in Eq. (62):

\[
\Delta_1 = x_1^* W_1 \bar{z}_1, \quad \text{where} \quad \bar{z}_1 = z_1 - \bar{H}_1 w_{z_1}
\]

We next introduce a posteriori errors

\[
\hat{a} = a - \bar{U} w_{x_1}, \quad \hat{b} = b - \bar{U} w_{z_1}
\]

and a priori errors

\[
\hat{a}_0 = a - \bar{U} w_x, \quad \hat{b}_0 = b - \bar{U} w_z
\]

that satisfy

\[
\hat{a} = R \hat{a}_0, \quad \hat{b} = R \hat{b}_0
\]

respectively. Note that, by the block RLS algorithm,

\[
w_{z_1} = w_z + \bar{P}_1 \bar{U}^* \hat{b}_0
\]
where
\[ \hat{P}_1 = (\lambda^{k\pi} + \bar{H}_1 W_1 \bar{H}_1) \], \quad \bar{\Gamma} = (R + \lambda^{-1} \bar{U} \hat{P}_1 \bar{U}^*)^{-1} \]

Also, the solutions of Eqs. (63) and (64) can be written as
\[ w_{2i} = \hat{P}_1 \bar{H}_1 W_2, \quad w_{2i} = \hat{P}_1 \bar{H}_1 W x \] (70)

Then, using the above expressions, we obtain
\[ \Delta_t = \begin{bmatrix} x^* a^* \end{bmatrix} \begin{bmatrix} \lambda W & 0 \\ 0 & R^{-1} \end{bmatrix} \begin{bmatrix} z \\ b \end{bmatrix} \begin{bmatrix} \bar{H} \\ \bar{\Gamma} \end{bmatrix} w_{2i} \]
\[ = \lambda x^* W z + a^* R^{-1} b - (\lambda x^* \bar{W} + a^* R^{-1} \bar{U}) w_{2i} \]
\[ = \lambda x^* W z + \lambda \bar{W} (w_2 - w_{2i}) + a^* R^{-1} \bar{b} \]
\[ = \lambda \Delta + a^* R^{-1} \bar{\Gamma}^{-1} R^{-1} \bar{b} \]
\[ = \lambda \Delta + a^* R^{-1} \bar{\Gamma} \tilde{b}_a \] (71)

The time-update formula Eq. (71) has the same form as that of the LSL filter [5, p.533], except that estimation errors are now vectors and the conversion factor is a matrix. Because of this fact, we can formally transform time-update relations of the LSL filter into their block RLS versions by replacing some scalar variables with their corresponding vector or matrix variables. For instance, we can transform Eq. (41) into
\[ \zeta_m^t(i) = \lambda \zeta_m^t(i-1) + \alpha_m^t \Gamma_{m,j-1} a^m_n \] (72)
by replacing \( \alpha_m(i) \) and \( \gamma_m(i-1) \) with \( \alpha_m,j_i \) and \( \Gamma_{m,j-1} \), respectively.

**Appendix A3 Time-update relation for \( \kappa_m(i) \)**

As an example, let us examine the time-update relation of \( \kappa_m(i) \), which is defined by
\[ \kappa_m(i) = \rho_m(i)/\zeta_m^t(i) \] (73)

In view of their LSL time-update relations [5, ch. 40] and Eq. (71), \( \rho_m(i) \) and \( \zeta_m^t(i) \) satisfy the recursions
\[ \rho_m(i) = \lambda \rho_m(i-1) + e^m_n \Gamma_{m,j} \beta_m,i, j \] (74)
\[ \zeta_m(i) = \lambda \zeta_m(i-1) + b^m_n \Gamma_{m,j} \beta_m,i, j \] (75)

with \( \rho_m(-1) = 0 \) and \( \zeta_m^t(-1) = \eta^{-1} \lambda^{-m-2} \), respectively.

We next introduce the normalized variables
\[ b^m_{m,j} = \Gamma_m^{1/2} \beta_{m,i, j}, \quad r^m_{m,j} = \Gamma_m^{1/2} e_{m,j} \] (76)
where \( \Gamma_m^{1/2} \Gamma_m^{1/2} = \Gamma_{m,i,j} \). With these normalized variables, Eqs. (75) and (76) can be rewritten as
\[ \rho_m(i) = \lambda \rho_m(i-1) + r^m_{m,j} b^m_{m,j} \] (77)
\[ \zeta_m^t(i) = \lambda \zeta_m^t(i-1) + || b^m_{m,j} ||^2 \] (78)

From Eqs. (77) and (78), we obtain
\[ \zeta_m(i) = \lambda \zeta_m(i-1) + || b^m_{m,j} ||^2 \] (79)
Recall that \( \rho_m(-1) = 0 \) and \( \zeta_m^t(-1) = 0 \), then by Eqs. (77) and (79),
\[ \rho_m(i) = \sum_{k=0}^{i} \lambda^{i-k} r^m_{m,k} b^m_{m,k}, \quad \zeta_m^t(i) = \sum_{k=0}^{i} \lambda^{i-k} || b^m_{m,k} ||^2 \] (80)

We further define two augmented vectors
\[ \tilde{b}_{m,j} = \begin{bmatrix} b^m_{m,0} \\ \vdots \\ b^m_{m,j} \end{bmatrix}, \quad \tilde{r}_{m,j} = \begin{bmatrix} r^m_{m,0} \\ \vdots \\ r^m_{m,j} \end{bmatrix} \] (81)

Then Eq. (80) can be simplified to
\[ \rho_m(i) = \tilde{r}_{m,j}^t \Lambda_{m,j} \tilde{b}_{m,j}, \quad \zeta_m^t(i) = \tilde{b}_{m,j}^t \Lambda_{m,j} \tilde{b}_{m,j} \] (82)
where \( \Lambda_{m,j} = \text{diag}(\lambda, \cdots, \lambda, 1) \). From Eqs. (57), (73), and (82), we can also write
\[ \kappa_m(i) = \arg \min_{\kappa_m \in C} \left\{ \sum_{j=0}^{i} \lambda_j^{i-1} \lambda + \tilde{b}_{m,j}^t \Lambda_{j} \tilde{b}_{m,j} \right\} \] (83)

where \( \eta_m = \gamma_m^{i+2} \). Note that Eq. (83) characterizes \( \kappa_m(i) \) as the solution to a scalar-valued least-squares problem, namely,
\[ \kappa_m(i) = \arg \min_{\kappa_m \in C} \left\{ \eta_m^{i+1} \kappa_m^2 + \sum_{j=0}^{i} \left( \tilde{r}_{m,j}^t \Lambda_{j} \tilde{b}_{m,j} \right) \right\} \] (84)

Applying the RLS algorithm to Eq. (84) and using Eqs. (57) and (76), we have the time-update relation
\[ \kappa_m(i) = \kappa_m(i-1) + \frac{b^m_{m,j} \left( r^m_{m,j} - \tilde{b}_{m,j} \kappa_m(i-1) \right)}{\eta_m^{i+1} \lambda + \tilde{b}_{m,j}^t \Lambda_{j} \tilde{b}_{m,j} - \left( \tilde{r}_{m,j}^t \Lambda_{j} \tilde{b}_{m,j} \right)} \] (85)

The time-update relations for \( \kappa_m(i) \) and \( \zeta_m^t(i) \) can be derived similarly.

Hidenori Matsuoka received his B.S. degree in physics from Ibaraki University in 1986 and his Dr. Eng. degree in control theory from Shinshu University in 2009. Dr. Matsuoka is a practicing engineer engaged in embedded systems development. His research interest includes control theory and its application to signal processing.

(Received June 22, 2019; revised October 27, 2019)