RESEARCH NOTE

A QRD-Based Array Lattice Form 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 In this research note, we present an array lattice algorithm for $H^\infty$ adaptive filtering, in which pre-arrays are transformed into post-arrays by $J$-unitary transformations regarding an indefinite signature matrix $J$. The array form is derived by factorization with respect to an indefinite Hermitian matrix, which functionally corresponds to the so-called conversion factor in recursive least-squares algorithms. The numerical stability of the array lattice form in finite-precision arithmetic is verified by simulation in comparison with the standard $H^\infty$ filter implementation.

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

1. Introduction

In adaptive filtering, it has been recognized that there is a trade-off between robustness against disturbances and efficiency measured from the rate of convergence [1, 2]. Indeed, despite the fact that the least-mean-square (LMS) filter is robust particularly in nonstationary environments, it converges much slower than the recursive least-squares (RLS) filter, which is more sensitive to disturbances [3, 4]. Hence, an algorithm cannot be highly robust and efficient at the same time. According to $H^\infty$ theory, however, it is possible to design a filter that can achieve a balance between robustness and efficiency [5]. For such $H^\infty$ adaptive filters, other than the standard form, there exist computationally efficient implementations [6, 7]. Also, in this paper, we shall consider yet another implementation, which could be classified as an array lattice form.

For the RLS filter, there is an algorithmic variant known as the array form, which is described not as an explicit set of equations but as a sequence of operations on arrays (matrices) containing square-root factors of original quantities [1, 2]. Namely, a pre-array is usually triangularized to yield a post-array from which the quantities needed to construct the next pre-array are read off and, in this way, the procedure can be repeated. Since the square-root factors assume values within smaller dynamic ranges and the triangularization is performed by numerically well-behaved unitary transformations, the array forms are more reliable than the original algorithms in finite-precision arithmetic. Particularly in the case of least-squares lattice (LSL) filters, although several array forms have been proposed, a QRD-based algorithm that employs QR decomposition (QRD) for triangularization would be the most popular one. For this reason, in this paper, we also attempt to derive such a QRD-based array form for the $H^\infty$ lattice filter presented in [8].

In principle, an array form can be derived by factorizing the original algorithm in some way. As far as the least-squares problems are concerned, square-root factorization is naturally employed for this purpose, because the key quantities are guaranteed to be positive. On the other hand, in $H^\infty$ filtering, we must deal with indefinite quantities, and hence, an alternative factorization is required. In fact, as we shall see, such a factorization becomes possible by introducing a $J$-unitary transformation for some indefinite signature matrix $J$. Moreover, in this case, the QR decomposition is performed using a sequence of a Givens rotation and a hyperbolic Givens rotation.

This paper is organized as follows. In section 2, we formulate the $H^\infty$ adaptive filtering problem and review the lattice form solution derived in [8]. In Section 3, we introduce a decomposition for a matrix variable related to the conversion factor and thereby transform the lattice algorithm into an array form. Section 4 presents several numerical examples to demonstrate the numerical stability of the $H^\infty$ array lattice filter. This research note is an addendum to [8] and readers are invited to refer to it. In particular, an error in [8] is corrected in Appendix 1.

(Notation) We adopt the following notational conventions: (i) capital letters are used for matrices and small letters are used for both vectors and scalars; (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; (v)
\[ \lambda \in \text{where} \]

Consider the signal model

\[ d(i) = u_i w^0 + v(i) \quad i \in \{0, 1, \cdots \} \]  

(1)

where \( d(i) \in \mathbb{C} \) is a measured output, \( u_i \in \mathbb{C}^{1 \times M} \) a known input data vector, \( w^0 \in \mathbb{C}^{M \times 1} \) a weight vector to be estimated, and \( v(i) \) is a known disturbance that typically contains measurement noise and modeling errors. The input data vector \( u_i \) is assumed to have the shift structure \( u_i = [u(i) u(i-1) \cdots u(i-M+1)] \), where \( u(i) \) is a known scalar sequence. Also, the disturbance is assumed to have finite energy, namely, \( \sum |v(i)|^2 \) < \( \infty \) (\( \forall i \) > 0).

Now let \( \omega \) denote an estimate of \( w^0 \) using the observations of \( d(0), d(1), \cdots, d(i) \). Then, for some given scalars \( \lambda \in (0, 1], \gamma_f > 0, N \gg 0 \), and positive-definite matrix \( \Pi \in \mathbb{R}^{M \times M} \), determine estimates \( \{\omega_j, \omega_{j+1}, \cdots, \omega_N\} \) so that for all \( i \in \{0, 1, \cdots, N \} \)

\[
\frac{1}{N+1} \sum_{j=0}^{N} |u_j w_j - u_j \omega_j|^2 < \gamma_f^2
\]  

(2)

can be guaranteed regardless of \( w^0 \) and \( v(i) \) \( \forall i \) > 0.

Problem 1 contains a parameter \( \gamma_f \), which restricts the energy ratios from disturbances to estimation errors. Hence, an algorithm that computes the solution under the smallest possible value of \( \gamma_f \) is said to be \( H^\infty \) optimal. In this sense, the LSM filter is known to be \( H^\infty \) optimal and satisfies condition (2) with \( \gamma_f = 1 \) [3]. However, it should be noted that this optimality is achieved at the cost of reduced efficiency. Indeed, the RLS filter that exhibits a much higher rate of convergence can satisfy condition (2) with a relaxed value of \( \gamma_f = 2 \) [4]. This fact leads us to the idea of the suboptimal \( H^\infty \) filter, which realizes a balance between robustness and efficiency by choosing \( \gamma_f \in (1, 2) \) properly. For such filters, there exists a standard algorithm whose complexity is on the order of \( M^2 \) [5]. Also, an order-recursive lattice form is presented in [8].

Algorithm 2 (A priori error-feedback \( H^\infty \) lattice filter)
Suppose \( \gamma_f > 1, \lambda \in (0, 1], \) and \( \eta > 0 \) and define

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

(3)

\[
R^{-1} = \begin{bmatrix} -\lambda \gamma_f^2 & 0 \\ 0 & 1 \end{bmatrix}
\]  

(4)

Then, the a priori estimation error \( e(i) = d(i) - u_i w_{i-1} \) and the a posteriori estimation error \( r(i) = d(i) - u_i \omega_i \) that result from a solution of Problem 1 can be computed as follows.

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

\[
\Gamma_{m,-1} = R^{-1}, \quad \beta_{m,-1} = 0_{2 \times 1}
\]

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

(2-a) Set

\[
\alpha_{0,i} = \beta_{0,i} = \begin{bmatrix} u(i-1) \\ u(i) \end{bmatrix}, \quad \Gamma_{0,i} = R^{-1}
\]

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

\[
\zeta_{m,i}(i) = \lambda \zeta_{m,i}(i-1) + \alpha_{m,i} \beta_{m,j} + \alpha_{m,j} \beta_{m,i} (5)
\]

\[
\zeta_{m,i}(i) = \lambda \zeta_{m,i}(i-1) + \beta_{m,i} \beta_{m,i} (6)
\]

\[
\beta_{m+1,i} = \beta_{m,i,1} - \zeta_{m,i}(i-1) \alpha_{m,i} (7)
\]

\[
\alpha_{m+1,i} = \alpha_{m,i} - \kappa_{m,i}(i-1) \beta_{m,i,1} (8)
\]

\[
e_{m+1,i} = e_{m,i} + \kappa_{m,i}(i-1) \beta_{m,i,1} (9)
\]

\[
\kappa_{m,i}(i) = \kappa_{m,i}(i-1) + \beta_{m,i,1} \beta_{m,i,1} e_{m,i,1} / \zeta_{m,i} (10)
\]

\[
\zeta_{m,i}(i) = \kappa_{m,i}(i-1) + \beta_{m,i,1} \beta_{m,i,1} / \zeta_{m,i} (11)
\]

\[
\zeta_{m,i}(i) = \kappa_{m,i}(i-1) + \alpha_{m,i} \beta_{m,i,1} / \zeta_{m,i} (12)
\]

\[
\Gamma_{m+1,i} = \Gamma_{m,i} - \beta_{m,i,1} \beta_{m,i,1} / \zeta_{m,i} (13)
\]

(2-c) Then the outputs are obtained by

\[ e(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} e_{M,i}, \quad r(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} R \Gamma_{M,i} e_{M,i} (14) \]

Note 3 In this algorithm, the matrix variable \( \Gamma_{m,i} \) can be interpreted as the conversion factor that converts the a priori estimation error \( e_{m,i} \) into the a posteriori estimation error \( r_{m,i} \) [8]. Namely, we can write \( r_{m,i} = R \Gamma_{M,i} e_{M,i} \).

Compared with the standard \( H^\infty \) filter, Algorithm 2 has several advantages including a reduced computational cost that is on the order of \( M \) and a modular structure that enables us to increase the estimation order without recalculating all previous values. However, when finite-precision arithmetic is required, an array variant of Algorithm 2 would be preferable for reasons mentioned in Section 1. Also, the array structure itself seems to be of theoretical interest and thus we examine it in the next section.

3. QRD-Based Array Lattice Algorithm

In the case of the RLS filter, an array form is obtained by the square-root factorization of the original algorithm, the key variables of which are positive scalars or positive definite matrices. However, in our \( H^\infty \) filtering, some quantities are not guaranteed to be positive. Indeed, the matrix \( \Gamma_{m,i} \) that plays a pivotal role in Algorithm 2 is an indefinite matrix for which no square-root factor exists. Hence, to transform Algorithm 2 into an array form, we would be preferable for reasons mentioned in Section 1.
must find another type of factorization. In so doing, we note that, from Eq. (4), \( \Gamma_{0,i} \) can be factored as
\[
\Gamma_{0,i} = R^{-1} = LSL^* \tag{15}
\]
where we define
\[
L = \begin{bmatrix}
A^{1/2} / \gamma_f & 0 \\
0 & 1
\end{bmatrix}, 
S = \begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\tag{16}
\]
In view of this fact, we attempt to generalize the factorization (15) to all \( \Gamma_{m,i} \) \((m = 0, 1, \ldots, M) \) as follows.

**Proposition 4** For \( m = 0, 1, \ldots, M \) and \( i \geq 0 \), \( \Gamma_{m,i} \in \mathbb{C}^{2 \times 2} \) is invertible and Hermitian.

**Proof.** As shown in [8], \( \Gamma_{m,i} \) is originally defined as
\[
\Gamma_{m,i} = \left( R + X^* U_{m,i} \tilde{P}_{m,i} U_{m,i}^* \right)^{-1}
\tag{17}
\]
where \( U_{m,i} \) and \( \tilde{P}_{m,i} \) is a positive definite matrix that does not appear in Algorithm 2 explicitly. Hence, by definition, \( \Gamma_{m,i} \) is invertible and Hermitian. \( \square \)

**Lemma 5** For all \( m \in \{0, 1, \ldots, M\} \) and \( i \geq 0 \), \( \Gamma_{m,i} \in \mathbb{C}^{2 \times 2} \) has two nonzero real eigenvalues with opposite signs, namely, \( \lambda_i(\Gamma_{m,i}) = 1 \) and \( L_i(\Gamma_{m,i}) = 1 \) hold.

**Proof.** For \( m = 0 \), since \( \Gamma_{0,i} = R^{-1} = \text{diag}(\{\lambda \gamma_f^2, 1\}) \), we immediately have \( I_i(\Gamma_{0,i}) = 1 \) and \( L_i(\Gamma_{0,i}) = 1 \). We next define a 3 \( \times \) 3 invertible Hermitian matrix
\[
X = \begin{bmatrix}
\frac{\lambda_i^2}{\lambda_i} & \frac{\lambda_i}{\lambda_i} & \frac{\lambda_i}{\lambda_i} \\
\frac{\lambda_i}{\lambda_i} & \frac{\lambda_i}{\lambda_i} & \frac{\lambda_i}{\lambda_i} \\
\frac{\lambda_i}{\lambda_i} & \frac{\lambda_i}{\lambda_i} & \frac{\lambda_i}{\lambda_i}
\end{bmatrix}
\tag{18}
\]
From Eqs. (6) and (13), the Shur complements of \( X \) with respect to \( \Gamma_{m,i} \) and \( \lambda_i^2 \) are equal to \( \lambda_i^2 \) and \( \lambda_i^2 \), respectively. Thus, we can factor \( X \) in two ways:
\[
X = \begin{bmatrix}
1 & \beta_{m,i} & \beta_{m,i} \\
0 & 1 & 1 \\
0 & 1 & 1
\end{bmatrix}
\tag{19}
\]
Consequently, \( \text{diag}(\lambda_i^2) \) and \( \lambda_i^2 \) are congruent and thus have the same inertia in accordance with Silvester’s law of inertia [1]. Since \( \lambda_i^2 \) and \( \lambda_i^2 \) have the same inertia. Hence, by induction, \( I_i(\Gamma_{m,i}) = 1 \) and \( L_i(\Gamma_{m,i}) = 1 \). \( \square \)

**Proposition 6** For all \( m \in \{0, 1, \ldots, M\} \) and \( i \geq 0 \), there exists \( L_{m,i} \in \mathbb{C}^{2 \times 2} \) satisfying
\[
\Gamma_{m,i} = L_{m,i} S L_{m,i}^* \tag{19}
\]
\( \square \)

\[\text{This method of checking inertia conditions is found in on p. } 717 \text{ in [1]}\]

Pro **Proof.** By Lemma 5, \( \Gamma_{m,i} \) has a negative eigenvalue \( \lambda_- = -|\lambda| \) and a positive eigenvalue \( \lambda_+ > 0 \). Since \( \Gamma_{m,i} \) is Hermitian as per Proposition 4, there exists a unitary matrix \( T_{m,i} \in \mathbb{C}^{2 \times 2} \) that factors \( \Gamma_{m,i} \) as
\[
\Gamma_{m,i} = T_{m,i} \begin{bmatrix}
-|\lambda| & 0 \\
0 & \lambda_i^2
\end{bmatrix} T_{m,i}^*
\]
Thus, we can set \( L_{m,i} = T_{m,i} \text{diag}[\sqrt{|\lambda|}, \sqrt{\lambda_+}] \).

Given the general factorization (19), we are now ready to derive an array form of Algorithm 2. For the convenience, we introduce the weighted estimation errors
\[
f_{m,i} = L_{m,i}^{-1} a_{m,i}, \quad b_{m,i} = L_{m,i}^{-1} \tilde{b}_{m,i}, \quad r_{m,i} = L_{m,i}^{-1} \tilde{r}_{m,i}
\tag{20}
\]
and the normalized reflection coefficients
\[
q_m(i) = \frac{\epsilon_m(i)}{b_m(i)}, \quad q_m(i) = \frac{\epsilon_m(i)}{b_m(i)}
\tag{21}
\]

**Algorithm 7** (QRD-based \( H^\infty \) array lattice filter)
Define the signature matrix \( J \in \mathbb{R}^{3 \times 3} \) as
\[
J = \text{diag}[1, S] = \text{diag}[1, 1, 1]
\tag{22}
\]
Then, the a priori estimation errors that result from Algorithm 2 can be equivalently computed as follows.

1. Initialization. From \( m = 0 \) to \( m = M - 1 \), set
\[
\begin{align*}
J_m & = J_{m-1} - b_{m}^* \tilde{r}_{m-1} = 0_{2 \times 1}, \quad r_{m-1} = 0 \\
l_{m} & = \begin{bmatrix}
\text{u}(i) \\
\text{u}(i)
\end{bmatrix}, \quad L_{0,i} = L
\end{align*}
\tag{23}
\]
2. For \( i \geq 0 \), repeat

(a) Set
\[
\begin{align*}
\text{b}^*_i \text{f}^*_m & = f_m^* \text{f}^*_m \text{L}^* \begin{bmatrix}
\text{u}(i-1) \\
\text{u}(i)
\end{bmatrix}, \quad L_{0,i} = L \\
r^*_i \text{f}^*_m & = \text{f}^*_m \text{f}^*_m \text{L}^* \begin{bmatrix}
\text{d}(i-1) - r(i-1) \\
\text{d}(i)
\end{bmatrix}
\end{align*}
\tag{24}
\]
(b) From \( m = 0 \) to \( m = M - 1 \), apply 3 \( \times \) 3 J-unitary rotations \( \Theta_m^{(i)}, \Theta_m^{(j)}, \text{and } \Theta_m^{(j)} \), in order to annihilate the (1,2) and (1,3) entries of the post-arrays below:
\[
\begin{bmatrix}
\text{a}^*_{m} \text{f}^*_{m} \text{u}(i-1) \\
\text{a}^*_{m} \text{f}^*_{m} \text{u}(i-1) \\
\text{a}^*_{m} \text{f}^*_{m} \text{u}(i-1)
\end{bmatrix}
\tag{25}
\]

---

Journal of Signal Processing, Vol. 25, No. 5, September 2021
\[
\begin{bmatrix}
A^{1/2}e^{f/2}(i - 1) & f_0^* \\
A^{1/2}d_0^*(i - 1) & b_0^{m-1,i}
\end{bmatrix}
\]

(2-c) Then, the a priori estimation error \(e(i)\) and the a posteriori estimation error \(r(i)\) are obtained as

\[
e(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} L_{k,j}^f r_{k,j}^f, \quad r(i) = \begin{bmatrix} 0 & 1 \end{bmatrix} R_{M,t} S r_{k,t}^f
\]

**Proof.** By Eqs. (19), (20), and (21), the equations in initialization, and steps (2-a) and (2-c) reduce to those of Algorithm 2. Regarding step (2-b), the equivalence can be verified by simply squaring both sides of each equation with the weighting matrix \(J\) and comparing the terms. For instance, by comparing the squared arrays of the second equation in step (2-b), we see that the (1,1) and (3,3) block entries coincide with Eqs. (6) and (13), respectively. Also, by equating (2,1) block entries, we have

\[
\xi_m^h(i,\kappa_m(i)) = A \xi_m^h(i - 1,\kappa_m(i - 1)) + \beta_m \Gamma_m e_{m+1,i}^h i, f
\]

By Eqs. (6), (9), and (10), Eq. (23) can be verified as

\[
\xi_m^h(i,\kappa_m(i)) = \xi_m^h(i - 1) \left[ \kappa_m(i - 1) + \beta_m \Gamma_m e_{m+1,i}^h i, f \right] + \beta_m \Gamma_m \left[ \kappa_m(i) - \kappa_m(i - 1) \right] + \beta_m \Gamma_m e_{m+1,i}^h i, f
\]

Other entries can be verified in the same way. \(\Box\)

**Note 8** In step (2-b), since the third diagonal component of \(J\) is +1, we use a Givens rotation to annihilate the (1,3) entry in the post-array. On the other hand, we must use a hyperbolic Givens rotation to annihilate the (1,2) entry, because the second diagonal component of \(J\) is not +1 but −1 (see Ch. 36 in [1]). Hence, each \(J\)-unitary matrix can be implemented as a sequence of a Givens rotation and a hyperbolic Givens rotation.

### 4. Numerical Examples

In this section, we evaluate the performance of the proposed algorithm by simulation. To begin with, let us note that as far as Problem 1 is concerned, the practical range of \(\gamma_f\) is roughly limited to \(1.01 \leq \gamma_f \leq 1.5\); outside this region, the performance of the \(H^\infty\) adaptive filter rapidly approaches that of LMS or RLS as \(\gamma_f \rightarrow 1.0\) or \(\rightarrow 2.0\), respectively [9]. Hence, in the following numerical experiments, we set \(\gamma_f = 1.01, 1.2\) or 1.5 so that the performance can be distinguishable in each region.

#### 4.1 Finite-precision performance

Since the array lattice \(H^\infty\) filter (Algorithm 7) is theoretically equivalent to its standard form (c.f., e.g., Algorithm 2 in [8]), they should behave identically as long as the infinite-precision arithmetic is assumed. In other words, it is only under the finite-precision condition that their difference becomes apparent, reflecting the numerical stability of each form. For this reason, to compare the performance of the array lattice form with that of the standard form, we perform simulation under the finite-precision condition where \(B\) denotes the number of bits including the sign bit. In the simulation, each filter estimates an unknown impulse response of length 10; the signal-to-noise ratio (SNR) is 30 dB and other parameters are \(\lambda = 0.995\) and \(\eta = 10^2\). First, let us set \(\gamma_f = 1.2\). Then, the learning curves averaged over 50 experiments for each value of \(B\) are plotted in Figs. 1 and 2. In Fig. 1, we see that the standard form converges at \(B = 40\) bits and diverges at \(B = 32\) and 16 bits. In contrast, in Fig. 2, the array lattice form consistently converges for all values of \(B\). Next, decreasing \(\gamma_f\) to 1.01, the learning curves resulting from each form again plotted in Figs 3 and 4. In these figures, we find the same tendency as observed in the case of \(\gamma_f = 1.2\); the standard form diverges at \(B = 32\) and 16 bits, while the array lattice form converges. Also, the results in the case of \(\gamma_f = 1.5\) are almost the same as those of \(\gamma_f = 1.2\), and thus, the figures are not shown here.
4.2 Tracking performance

We next demonstrate the tracking performance of the proposed array lattice algorithm by presenting two examples in which we use floating-point arithmetic. The first example is in a nonstationary environment, in which the true weight vector \( w^o \) varies in accordance with the first-order random-walk model:

\[
    w^o_i = w^o_{i-1} + q_i \quad (i \geq 0),
\]

where \( q_i \in \mathbb{C}^{M \times 1} \) denotes a zero-mean random vector whose variance is given by \( \sigma_q^2 \). Setting \( M = 30 \), \( \sigma_q^2 = 10^{-4} \), \( \lambda = 0.995 \), and \( \eta = 10^5 \), we plot in Fig. 5 ensemble averaged (50 runs) learning curves of array lattice \( H^\infty \) filters with \( \gamma_f = 1.01, 1.2, \) and 1.5 and the RLS filter, which can be approximately regarded as an \( H^\infty \) filter with \( \gamma_f = 2.0 \). The second example is in a suddenly time-varying environment, where an impulsive noise of unit amplitude added at iteration 200. The learning curves are plotted in Fig. 6 using the same set of filters as in the case of Fig 5. These figures show how the choice of \( \gamma_f \) improves the robustness to disturbances: as \( \gamma_f \) decreases, the divergence in mean-square error is gradually mitigated in

---

\(^2\)Under the floating-point conditions, as theoretically expected, the array lattice algorithm performs identically with the standard form and other variants, of which simulation results are presented in [8, 9].
Appendix 1 Correction to Appendix A3 of [8]

In Appendix A3 of [8], there was an error in the process of deriving the time-update expression for \( \kappa_m(i) \), where the square-root factor of an indefinite matrix \( \Gamma_{m,i} \) was invalidly introduced. Although the conclusion remains unchanged, we present the corrected derivation below.

Let us first recall that \( \kappa_m(i) \) is defined by

\[
\kappa_m(i) = \frac{\rho^*_m(i)}{\zeta^*_m(i)}
\]

where \( \rho_m(i) \) and \( \zeta^*_m(i) \) satisfy the recursions

\[
\rho_m(i) = \lambda \rho_m(i-1) + \zeta^*_{m,i} \Gamma_{m,i} \beta_{m,i} \tag{25}
\]

\[
\zeta^*_m(i) = \lambda \zeta^*_m(i-1) + \rho^*_m(i) \Gamma_{m,i} \beta_{m,i} \tag{26}
\]

with initial conditions \( \rho_m(-1) = 0 \) and \( \zeta^*_m(-1) = \eta^{-1} \lambda^{-m-2} \), respectively. Substituting Eq. (19) into (25) and (26),

\[
\rho_m(i) = \lambda \rho_m(i-1) + \zeta^*_{m,i} \Lambda_m \beta_{m,i} \tag{27}
\]

\[
\zeta^*_m(i) = \lambda \zeta^*_m(i-1) + \rho^*_m(i) \Lambda_m \beta_{m,i} \tag{28}
\]

Noting \( \rho_m(-1) = 0 \), Eq. (27) can be written in the closed-form expression

\[
\rho_m(i) = \sum_{k=0}^{i} \lambda^{i-k} \zeta^*_{m,k} \Lambda_m \beta_{m,k} = \tilde{r}_{m,i} \Lambda_i \tilde{b}_{m,i} \tag{29}
\]

where we defined

\[
\Lambda_i = \text{diag} \{ \lambda^0S, \ldots, \lambda^iS, S \} \tag{30}
\]

In the same way, from Eq. (28), we obtain

\[
\zeta^*_m(i) = \bar{\eta}_m \lambda^i + \tilde{b}_{m,i} \Lambda_i \tilde{b}_{m,i} ^{-1} \tilde{r}_{m,i} \tag{31}
\]

Then, substituting Eqs. (29) and (32) into Eq. (24),

\[
\kappa_m(i) = \left| \bar{\eta}_m \lambda^i + \tilde{b}_{m,i} \Lambda_i \tilde{b}_{m,i} ^{-1} \right| \left( \left| \lambda^i \tilde{r}_{m,i} \right| + \left| \tilde{b}_{m,i} \right| \kappa_m(i-1) \right) ^2 \tag{32}
\]

where \( \bar{\eta}_m := \eta \lambda^{m+2} \). Note that Eq. (33) characterizes \( \kappa_m(i) \) as the solution to a scalar-valued indefinite least-squares minimization problem, namely,

\[
\kappa_m(i) = \arg \min_{\kappa \in \mathbb{C}} \left\{ \left| \bar{\eta}_m \lambda^i + \tilde{b}_{m,i} \Lambda_i \tilde{b}_{m,i} ^{-1} \right| \left| \lambda^i \tilde{r}_{m,i} \right| + \left| \tilde{b}_{m,i} \right| \kappa \left| \kappa \right|^2 \right\} \tag{33}
\]

Then, applying the recursive minimization algorithm (see p. 712 in [1]) to Eq. (34) and using Eqs. (9) and (19), we obtain the time-update relation Eq. (10) as follows:

\[
\kappa_m(i) \left| \kappa \right|^2 = \kappa_m(i-1) + \frac{L_{m,i} \beta_{m,i} ^*}{\bar{\eta}_m \lambda^i + \tilde{b}_{m,i} \Lambda_i \tilde{b}_{m,i}} \left( \left| \lambda^i \tilde{r}_{m,i} \right| + \left| \tilde{b}_{m,i} \right| \kappa_m(i-1) \right) \tag{34}
\]

The time-update relations for \( \kappa'_m(i) \) and \( \kappa''_m(i) \) can be derived similarly.

Hidenori Matsuzaki received his B.S. degree in physics from Ibaraki University in 1986 and his Dr. Eng. degree in control theory from Shinsyu University in 2009. His research interests include control theory and its application to adaptive signal processing.

(Received November 30, 2020; revised March 15, 2021)