Derivation of A Fast Algorithm of Modified H_∞ Filters

Dep. of Comp. & Info. Science, Iwate University

Kiyoshi NISHIYAMA

Abstract— **The fast Kalman filter provides very quick convergence at the computational complexity of the same order that the LMS algorithm requires. Nevertheless, its performance is still unsatisfactory in system identification because the conventional fast Kalman filter fails to track time-varying impulse responses of FIR systems. The failure of tracking is due to the absence of system noise in the state-space model to be used. However, according to the derivation of the fast Kalman filter, it is difficult to theoretically introduce the term of system noise into the algorithm. In this paper, to overcome the difficulties, a new fast filtering algorithm, called a fast H_∞ filter, is derived based on the H_∞ theory.**

Keywords— H_∞ **filter, fast algorithm, Kalman filter, robust estimation, system identification, LMS**

I. Introduction

Efficient linear estimation algorithms have been developed in the past based mainly on the minimization of the L_2-norm of the estimation error [1],[2]. In such approaches, the Kalman filter has played a central role among the minimum variance estimators, and fortunately, the fast algorithm, called the fast Kalman filter, has been developed for a specific state-space model [3],[4].

Recently, a measure which differs from the L_2-norm has been introduced for optimal estimation [5],[6]. This measure is the H_∞-norm of the operation that relates the exogenous disturbances (the initial state and noises) to the estimation error, and it has been used successfully in optimal estimation. As a solution to the suboptimal H_∞ estimation problem in which the H_∞-norm is less than a prescribed positive value, an H_∞ filter has been derived from the game theory approach which permits a consideration of finite-time H_∞ filtering problems for time-varying state-space models [7]. The H_∞ filtering problem has also been solved as linear estimation in an indefinite-metric space, called the Krein space [8]. The H_∞ filters have attracted much attention in the field of robust estimation [5]-[9], since they attempt to optimally estimate the required combination of the states against the worst disturbances. However, their computational complexity becomes intractable as the size of the state vector grows large. To solve this problem, some fast array algorithms have been developed using J-unitary transformations [8]. Unfortunately, they have been applicable only for time-invariant state-space models.

Focusing on the fast Kalman filter, it provides extremely fast convergences for system identification of FIR systems at a reasonable computational requirement, whose order is equal to that of the LMS algorithm. Nevertheless, its performance is still unsatisfactory because the standard fast Kalman filter fails to track the time-varying impulse responses. The failure of tracking in the the fast Kalman filter is due to the absence of system noise in the state-space model to be used. However, according to the derivation of the fast Kalman filter, it is essentially difficult to theoretically introduce the term of system noise into the algorithm.

In this paper, a new fast filtering algorithm, which is called a fast H_∞ filter, is derived in the sense of H_∞ optimization, which possesses the capability of successfully tracking the time-varying impulse responses on real-time with a reasonable computational cost. Furthermore, a condition for the fast H_∞ filter to exist is given, which makes it easy to use the filter in practical situations.

II. Modified H_∞ Filters

For simplicity, we consider the following specific time-varying state-space model:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{w}_k, \qquad \boldsymbol{w}_k, \boldsymbol{x}_k \in \mathcal{R}^N \tag{1}$$

$$y_k = \boldsymbol{H}_k \boldsymbol{x}_k + v_k, \qquad y_k, v_k \in \mathcal{R} \tag{2}$$

$$z_k = \boldsymbol{H}_k \boldsymbol{x}_k, \quad z_k \in \mathcal{R},\ \boldsymbol{H}_k \in \mathcal{R}^{1\times N} \tag{3}$$

which is often used for echo cancellers and system identification although it appears to be a very specialized model, where z_k is the signal to be estimated and $\boldsymbol{H}_k$ has a shift-property such that $\boldsymbol{H}_{k+1}(i+1) = \boldsymbol{H}_k(i)$, $\boldsymbol{H}_k = [\ u_k \quad u_{k-1} \quad \cdots \quad u_{k-(N-1)}\ ]$. Furthermore, to simplify the presentation, we will confine our attention in this paper to the case of real-valued data.

Theorem 1: For such a state-space model, a (level-γ_f) modified H_∞ filter to achieve

$$\sup_{x_0,\{w_i\},\{v_i\}} \cdot \frac{\sum_{i=0}^{k} \|e_{f,i}\|^2/\rho}{\|\boldsymbol{x}_0 - \check{x}_{0|-1}\|^2_{\boldsymbol{\Sigma}_0^{-1}} + \sum_{i=0}^{k}\|w_i\|^2_{\boldsymbol{\Sigma}_{w_k}} + \sum_{i=0}^{k}\|v_i\|^2/\rho} < \gamma_f^2 \tag{4}$$

is represented by

$$\check{z}_{k|k} = \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k} \tag{5}$$

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{K}_{s,k+1}(y_{k+1} - \boldsymbol{H}_{k+1}\hat{\boldsymbol{x}}_{k|k}) \tag{6}$$

The author is with the Dep. of Comp. & Info. Science, Faculty of Engineering, Iwate University, 4-3-5, Ueda, Morioka, JAPAN. Tel/Fax: +81 19-621-6475, E-mail: nisiyama@cis.iwate-u.ac.jp

$$\begin{aligned}\boldsymbol{K}_{s,k+1} &= \hat{\boldsymbol{\Sigma}}_{k+1|k}\boldsymbol{H}_{k+1}^T \\ &\cdot\ (\boldsymbol{H}_{k+1}\hat{\boldsymbol{\Sigma}}_{k+1|k}\boldsymbol{H}_{k+1}^T+\rho)^{-1} \quad (7)\\ \hat{\boldsymbol{\Sigma}}_{k+1|k} &= \hat{\boldsymbol{\Sigma}}_{k|k-1}-\hat{\boldsymbol{\Sigma}}_{k|k-1}[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T] \\ &\cdot\ \boldsymbol{R}_{e,k}^{-1}\begin{bmatrix}\boldsymbol{H}_k\\ \boldsymbol{H}_k\end{bmatrix}\hat{\boldsymbol{\Sigma}}_{k|k-1}+\boldsymbol{\Sigma}_{w_k} \quad (8)\end{aligned}$$

where

$$\begin{aligned} e_{f,i} &= \check{z}_{i|i}-\boldsymbol{H}_i\boldsymbol{x}_i\\ \boldsymbol{R}_{e,k} &= \boldsymbol{R}_k+\begin{bmatrix}\boldsymbol{H}_k\\ \boldsymbol{H}_k\end{bmatrix}\hat{\boldsymbol{\Sigma}}_{k|k-1}[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T]\\ \boldsymbol{R}_k &= \begin{bmatrix}\rho & 0\\ 0 & -\rho\gamma_f^2\end{bmatrix},\quad \boldsymbol{\Sigma}_{w_k}=\gamma_f^{-2}\hat{\boldsymbol{\Sigma}}_{k+1|k}\\ \hat{\boldsymbol{P}}_{1|0} &= \varepsilon_0\boldsymbol{I},\ \varepsilon_0>0,\ 0<\rho=1-\gamma_f^{-2}\le 1,\ \gamma_f>1 \quad (9)\end{aligned}$$

and the parameter γ_f should be sufficiently adjusted small for robustness, as long as the following existence condition is satisfied:

$$\hat{\boldsymbol{\Sigma}}_{i|i-1}^{-1} + \boldsymbol{H}_i^T\boldsymbol{H}_i>0,\quad i=0,\cdots,k. \quad (10)$$

Proof: Supposing that the weight parameter ρ is independent of γ_f, we can easily derive the above modified H_∞ filter and the existence condition using the standard H_∞ estimation scheme [8]. ■

It should be noted here that since ρ appears on the left-hand side of (4) and depends on γ_f, the above algorithm is a modified version of the ordinary central H_∞ filter, i.e., the present suboptimal H_∞ estimation problem is of a nonstandard type.

If the condition of (10) is not satisfied, then any H_∞ filter to achieve (4) no longer exists, i.e., it is not guaranteed that the estimation level (the H_∞-norm on the time interval $[0,k]$) is less than γ_f for any case. So, the determination of γ_f is very important in both existence and performance. The requirement of (10) for existence is also equivalent to the condition that the matrices $\boldsymbol{R}_k$ and $\boldsymbol{R}_{e,k}$ have the same inertia [8].

As seen in (4), the H_∞ boundness is modified in advance to theoretically introduce a nonstationary system noise with the covariance $\boldsymbol{\Sigma}_{w_k}$ into the state-space model through the parameter $\rho=1-\gamma_f^{-2}$, which can be regarded as a forgetting factor. The system noise $\boldsymbol{w}_k$, whose individual entries are not necessarily independent, makes it possible for the H_∞ filter to track the variations in the dynamic behavior of unknown systems. Note that the smaller γ_f is chosen the larger the effect of system noise becomes. Moreover, one has no use for the determination of $\boldsymbol{\Sigma}_{w_k}$. Indeed, the update of the error covariance matrix is carried out as

$$\hat{\boldsymbol{\Sigma}}_{k+1|k} = \left(\hat{\boldsymbol{\Sigma}}_{k|k-1}-\hat{\boldsymbol{\Sigma}}_{k|k-1}[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T]\cdot\boldsymbol{R}_{e,k}^{-1}\begin{bmatrix}\boldsymbol{H}_k\\ \boldsymbol{H}_k\end{bmatrix}\hat{\boldsymbol{\Sigma}}_{k|k-1}\right)/\rho. \quad (11)$$

Here, it is worth noting that the so-called γ-iteration theoretically provides an optimum value of the forgetting factor ρ, which has been empirically chosen without evidence of optimality in the conventional approaches.

The major computational burden of the H_∞ filter lies in the update of $\hat{\boldsymbol{\Sigma}}_{k|k-1}\in\boldsymbol{\mathcal{R}}^{N\times N}$ for $\boldsymbol{K}_{s,k+1}$ which requires proportional-to-N^2, i.e., $\boldsymbol{O}(N^2)$ arithmetic operations per time step, where N is the dimension of $\boldsymbol{x}_k$. Hence, on increasing the dimension of $\boldsymbol{x}_k$, the computation time required to run the H_∞ filter increases rapidly.

To overcome this drawback, we must develop a fast algorithm of the H_∞ filter.

III. Fast Algorithm of Modified H_∞ Filters

In this section, by taking into account the shifting property of the sequences of $\boldsymbol{H}_k$, the number of arithmetic operations required to compute the filter gain $\boldsymbol{K}_{s,k}$ for each time step is reduced from $\boldsymbol{O}(N^2)$ to $\boldsymbol{O}(N)$. Before the derivation of the fast algorithm, some preparations are given.

A. Preparations

An alternative framework is given to efficiently derive a fast algorithm of the modified H_∞ filter as follows. As the first step, one starts with recursively determining

$$\boldsymbol{K}_k=\boldsymbol{P}_k\boldsymbol{C}_k^T\quad\in\boldsymbol{\mathcal{R}}^{N\times 2} \quad (12)$$

instead of $\boldsymbol{K}_{s,k}\in\boldsymbol{\mathcal{R}}^{N\times 1}$ in (7), where $\boldsymbol{K}_k$ is called a gain matrix hereafter and

$$\begin{aligned}\boldsymbol{P}_k &= [\boldsymbol{\mathcal{O}}_k^T\boldsymbol{\Omega}_k\boldsymbol{\mathcal{O}}_k]^{-1}=[\sum_{i=1}^{k}\rho^{k-i}\boldsymbol{C}_i^T\boldsymbol{W}_i\boldsymbol{C}_i]^{-1}\\ \boldsymbol{\Omega}_k &= \left[\begin{array}{c|c}\rho\boldsymbol{\Omega}_{k-1} & \boldsymbol{0}\\ \hline \boldsymbol{0} & \boldsymbol{W}_k\end{array}\right],\quad \boldsymbol{\Omega}_1=\boldsymbol{W}_1\\ \boldsymbol{W}_i &= \rho\boldsymbol{R}_i^{-1}=\rho\begin{bmatrix}1 & 0\\ 0 & -\gamma_f^{-2}\end{bmatrix}\in\boldsymbol{\mathcal{R}}^{2\times 2}\\ \boldsymbol{\mathcal{O}}_k &= \begin{bmatrix}\boldsymbol{C}_1\\ \vdots\\ \boldsymbol{C}_k\end{bmatrix},\quad \boldsymbol{C}_i=\begin{bmatrix}\boldsymbol{H}_i\\ \boldsymbol{H}_i\end{bmatrix}\in\boldsymbol{\mathcal{R}}^{2\times N}. \quad (13)\end{aligned}$$

This is because $\boldsymbol{K}_k$ is more convenient than $\boldsymbol{K}_{s,k}$ to derive the fast algorithm in an iterative manner without the Riccati recursion, and moreover, $\boldsymbol{P}_k$ completely agrees with $\hat{\boldsymbol{\Sigma}}_{k+1|k}$ in (8) under a certain initial condition.

Now we can prove the following.

Lemma 1: The matrix $\boldsymbol{P}_k$ defined by (13) satisfies the Riccati recursion of (8).

Proof: Taking the inverse of $\boldsymbol{P}_k$, we have

$$\begin{aligned}\boldsymbol{P}_k^{-1} &= \rho\boldsymbol{\mathcal{O}}_{k-1}^T\boldsymbol{\Omega}_{k-1}\boldsymbol{\mathcal{O}}_{k-1}+\boldsymbol{C}_k^T\boldsymbol{W}_k\boldsymbol{C}_k\\ &= \rho\boldsymbol{P}_{k-1}^{-1}+\boldsymbol{C}_k^T\boldsymbol{W}_k\boldsymbol{C}_k. \quad (14)\end{aligned}$$

Furthermore, using the matrix inversion lemma, we obtain the recursion of $\boldsymbol{P}_k$ as

$$\begin{aligned}\boldsymbol{P}_k &= [\rho\boldsymbol{P}_{k-1}^{-1}+[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T]\boldsymbol{W}_k\begin{bmatrix}\boldsymbol{H}_k\\ \boldsymbol{H}_k\end{bmatrix}]^{-1}\\ &= \rho^{-1}\boldsymbol{P}_{k-1}-\rho^{-1}\boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T]\\ &\cdot\ (\boldsymbol{W}_k^{-1}+\begin{bmatrix}\boldsymbol{H}_k\\ \boldsymbol{H}_k\end{bmatrix}\rho^{-1}\boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T\ \boldsymbol{H}_k^T])^{-1}\end{aligned}$$

$$
\begin{aligned}
&\cdot \quad \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_k \end{bmatrix} \rho^{-1}\boldsymbol{P}_{k-1}, \\
\rho\boldsymbol{P}_k \quad &= \quad \boldsymbol{P}_{k-1} - \boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T \ \boldsymbol{H}_k^T] \\
&\cdot \quad \left(\boldsymbol{R}_k + \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_k \end{bmatrix} \boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T \ \boldsymbol{H}_k^T]\right)^{-1} \\
&\cdot \quad \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_k \end{bmatrix} \boldsymbol{P}_{k-1}, \\
\boldsymbol{P}_k \quad &= \quad \boldsymbol{P}_{k-1} - \boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T \ \boldsymbol{H}_k^T] \\
&\cdot \quad \left(\boldsymbol{R}_k + \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_k \end{bmatrix} \boldsymbol{P}_{k-1}[\boldsymbol{H}_k^T \ \boldsymbol{H}_k^T]\right)^{-1} \\
&\cdot \quad \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_k \end{bmatrix} \boldsymbol{P}_{k-1} + \gamma_f^{-2}\boldsymbol{P}_k. \qquad (15)
\end{aligned}
$$

Here, regarding $\boldsymbol{P}_k$ as $\hat{\boldsymbol{\Sigma}}_{k+1|k}$, we easily see that the above recursion is identical to the Riccati equation of (8). ■

On the other hand, the important relationship between the filter gain $\boldsymbol{K}_{s,k}$ and the gain matrix $\boldsymbol{K}_k$ is given by the following lemma.

Lemma 2: The filter gain $\boldsymbol{K}_{s,k}$ of the modified H_∞ filter is expressed in terms of $\boldsymbol{K}_k$ as

$$
\boldsymbol{K}_{s,k} \ = \ G_k^{-1}\tilde{\boldsymbol{K}}_k, \ \ G_k = \rho + \gamma_f^{-2}\boldsymbol{H}_k\tilde{\boldsymbol{K}}_k \ \ \in \mathcal{R} \quad (16)
$$

where

$$
\tilde{K}_k(i) = \rho K_k(i,1), \quad i = 1, 2, \cdots, N. \qquad (17)
$$

Proof: Taking into account that the gain matrix $\boldsymbol{K}_k$ is arranged as

$$
\begin{aligned}
\boldsymbol{K}_k &= \boldsymbol{P}_k\boldsymbol{C}_k^T \\
&= \left[\rho\boldsymbol{P}_{k-1}^{-1} + \boldsymbol{C}_k^T\boldsymbol{W}_k\boldsymbol{C}_k\right]^{-1}\boldsymbol{C}_k^T \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T - \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T \\
&\cdot \left[\boldsymbol{W}_k^{-1} + \boldsymbol{C}_k\rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\right]^{-1}\boldsymbol{C}_k\rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T \\
&- \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\left[\boldsymbol{W}_k^{-1} + \boldsymbol{C}_k\rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\right]^{-1} \\
&\cdot \left[(\boldsymbol{W}_k^{-1} + \boldsymbol{C}_k\rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T) - \boldsymbol{W}_k^{-1}\right] \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\left[\boldsymbol{I} + \boldsymbol{W}_k\boldsymbol{C}_k\rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\right]^{-1} \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\boldsymbol{W}_k \\
&\cdot \left[\boldsymbol{W}_k + \rho^{-1}\boldsymbol{W}_k\boldsymbol{C}_k\boldsymbol{P}_{k-1}\boldsymbol{C}_k^T\boldsymbol{W}_k\right]^{-1} \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\begin{bmatrix} \boldsymbol{H}_k^T & -\gamma_f^{-2}\boldsymbol{H}_k^T \end{bmatrix}\left[\begin{bmatrix} 1 & 0 \\ 0 & -\gamma_f^{-2} \end{bmatrix}\right. \\
&\left. +\rho^{-1}\begin{bmatrix} \boldsymbol{H}_k \\ -\gamma_f^{-2}\boldsymbol{H}_k \end{bmatrix}\boldsymbol{P}_{k-1}\begin{bmatrix} \boldsymbol{H}_k^T & -\gamma_f^{-2}\boldsymbol{H}_k^T \end{bmatrix}\right]^{-1} \\
&= \rho^{-1}\boldsymbol{P}_{k-1}\begin{bmatrix} \boldsymbol{H}_k^T & \boldsymbol{H}_k^T \end{bmatrix}\left(1 + \boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T\right)^{-1} \quad (18)
\end{aligned}
$$

and using $G_k = (\rho + \boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T)/(1 + \boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T)$ and $\boldsymbol{H}_k\tilde{\boldsymbol{K}}_k = \boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T/(1 + \boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T)$, we can obtain from the first block column of $\boldsymbol{K}_k$ the transformation of (16). ■

The above lemmas mean that our problem of determining the filter gain $\boldsymbol{K}_{s,k}$ is equivalent to the determination of the gain matrix $\boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{C}_k^T$.

According to (14), $\boldsymbol{P}_k^{-1}$ is also expressed by

$$
\boldsymbol{P}_k^{-1} = \frac{\rho}{\varepsilon_0}\boldsymbol{I} + \sum_{i=1}^{k}\rho^{k-i}\boldsymbol{C}_i^T\boldsymbol{W}_i\boldsymbol{C}_i \qquad (19)
$$

for $k > 0$ when it is set that $\boldsymbol{P}_0 = \varepsilon_0\boldsymbol{I}$. It follows that $\boldsymbol{P}_k$ in (14) tends to $\boldsymbol{P}_k$ in (13) when ε_0 approaches infinity ($\varepsilon_0 \to \infty$).

For further convenience, defining $\boldsymbol{Q}_k = \boldsymbol{P}_k^{-1}$ and returning to (12), we find that our objective is to recursively determine the gain matrix $\boldsymbol{K}_k$ which satisfies

$$
\boldsymbol{Q}_k\boldsymbol{K}_k \ = \ \boldsymbol{C}_k^T \qquad (20)
$$

at the computational burden of $\mathcal{O}(N)$.

In the following, the fast calculation of $\boldsymbol{K}_k$ is given using the shifting property of $\boldsymbol{C}_k^T \in \boldsymbol{\mathcal{R}}^{N\times 2}$, requiring $\mathcal{O}(N)$ arithmetic operations per time step.

B. Fast Calculation of the Gain Matrix

From *lemma 1* and *lemma 2*, our aim can be replaced by determining the gain matrix $\boldsymbol{K}_k$. The following lemma gives a solution to the problem.

Lemma 3: The gain matrix $\boldsymbol{K}_k$ is updated as

$$
\boldsymbol{K}_{k+1} \ = \ \boldsymbol{m}_k - \boldsymbol{B}_k F_k{}^{-1}\boldsymbol{\mu}_k \ \ \in \boldsymbol{\mathcal{R}}^{N\times 2} \qquad (21)
$$

where $\boldsymbol{m}_k \in \mathcal{R}^{N\times 2}$ and $\boldsymbol{\mu}_k \in \mathcal{R}^{1\times 2}$ are obtained, through the partition of $\breve{\boldsymbol{K}}_k = \breve{\boldsymbol{Q}}_k^{-1}\breve{\boldsymbol{C}}_k$, from

$$
\begin{bmatrix} \boldsymbol{m}_k \\ \boldsymbol{\mu}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{K}_k \end{bmatrix} + \begin{bmatrix} S_k^{-1} \\ \boldsymbol{A}_k S_k^{-1} \end{bmatrix}\left[\boldsymbol{c}_k^T + \boldsymbol{A}_k^T\boldsymbol{C}_k^T\right] \qquad (22)
$$

and the matrices $\boldsymbol{A}_k \in \mathcal{R}^{N\times 1}$, $S_k \in \mathcal{R}$, and $\boldsymbol{B}_k F_k^{-1} \in \mathcal{R}^{N\times 1}$ are given by *Lemma 4* and *Lemma 5*

Proof: Now, we assume that $\boldsymbol{K}_i, i = 0, \cdots, k$ has been given, and then solve the problem of determining $\boldsymbol{K}_{k+1}$, defined by

$$
\boldsymbol{Q}_{k+1}\boldsymbol{K}_{k+1} \ = \ \boldsymbol{C}_{k+1}^T. \qquad (23)
$$

To take advantage of the shifting property of $\boldsymbol{C}_k$, we introduce

$$
\breve{\boldsymbol{C}}_k^T \ = \ \begin{bmatrix} \boldsymbol{c}_k^T \\ \boldsymbol{C}_k^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{k+1}^T \\ \boldsymbol{c}_{k-N}^T \end{bmatrix} \in \boldsymbol{\mathcal{R}}^{(N+1)\times 2} \qquad (24)
$$

and

$$
\breve{\boldsymbol{Q}}_k \ = \ \sum_{i=1}^{k}\rho^{k-i}\breve{\boldsymbol{C}}_i^T\boldsymbol{W}_i\breve{\boldsymbol{C}}_i \in \boldsymbol{\mathcal{R}}^{(N+1)\times(N+1)} \qquad (25)
$$

which is expressed in recursive fashion by

$$
\breve{\boldsymbol{Q}}_k \ = \ \rho\breve{\boldsymbol{Q}}_{k-1} + \breve{\boldsymbol{C}}_k^T\boldsymbol{W}_k\breve{\boldsymbol{C}}_k. \qquad (26)
$$

Furthermore, it is partitioned as

$$
\breve{\boldsymbol{Q}}_k \ = \ \begin{bmatrix} M_k & \boldsymbol{T}_k^T \\ \boldsymbol{T}_k & \boldsymbol{Q}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_{k+1} & \underline{\boldsymbol{T}}_k^T \\ \underline{\boldsymbol{T}}_k & \underline{M}_k \end{bmatrix}. \qquad (27)
$$

With this notation, the equations (20) and (23) are contained in the following:

$$\breve{Q}_k \begin{bmatrix} \mathbf{0} \\ K_k \end{bmatrix} = \begin{bmatrix} \alpha_k^T \\ C_k^T \end{bmatrix} = \breve{C}_k^T + \begin{bmatrix} \alpha_k^T - c_k^T \\ \mathbf{0} \end{bmatrix} \quad (28)$$

$$\breve{Q}_k \begin{bmatrix} K_{k+1} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} C_{k+1}^T \\ \beta_k^T \end{bmatrix} = \breve{C}_k^T + \begin{bmatrix} \mathbf{0} \\ \beta_k^T - c_{k-N}^T \end{bmatrix} \quad (29)$$

where

$$\alpha_k^T = T_k^T K_k \in \mathcal{R}^{1\times 2}, \quad \beta_k^T = \underline{T}_k K_{k+1} \in \mathcal{R}^{1\times 2}.$$

In view of these expressions, it seems more reasonable to find $\breve{K}_k \in \mathcal{R}^{(N+1)\times 2}$ which satisfies

$$\breve{Q}_k \breve{K}_k = \breve{C}_k^T \quad (30)$$

as an intermediate step before determining K_k, where

$$\breve{K}_k = [k_{k+1}^T \ K_k^T]^T = [K_{k+1}^T \ k_{k-N}^T]^T. \quad (31)$$

T o do so, using

$$\breve{C}_k^T = \breve{Q}_k \begin{bmatrix} \mathbf{0} \\ K_k \end{bmatrix} - \begin{bmatrix} {\alpha_k}^T - {c_k}^T \\ \mathbf{0} \end{bmatrix} \quad (32)$$

which is obtained from (28), w e arrange $\breve{K}_k \in \mathcal{R}^{(N+1)\times 2}$ as

$$\begin{aligned} \breve{K}_k &= \begin{bmatrix} m_k \\ \mu_k \end{bmatrix} \\ &= \breve{Q}_k^{-1}\breve{C}_k^T = \begin{bmatrix} \mathbf{0} \\ K_k \end{bmatrix} - \breve{Q}_k^{-1} \begin{bmatrix} \alpha_k^T - c_k^T \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} \\ K_k \end{bmatrix} - \begin{bmatrix} S_k^{-1} \\ A_k S_k^{-1} \end{bmatrix} [\alpha_k^T - c_k^T] \end{aligned} \quad (33)$$

where $\breve{K}_k$ is partitioned with $m_k \in \mathcal{R}^{N\times 2}$ and $\mu_k \in \mathcal{R}^{1\times 2}$. Note that $\alpha_k^T - c_k^T = -(c_k^T + A_k^T C_k^T)$ holds. Also, $\breve{Q}_k$ is invertible, and $A_k \in \mathcal{R}^{N\times 1}$ and $S_k \in \mathcal{R}$ satisfy

$$\breve{Q}_k \begin{bmatrix} 1 \\ A_k \end{bmatrix} = \begin{bmatrix} S_k \\ \mathbf{0} \end{bmatrix} \quad \left(\begin{bmatrix} 1 \\ A_k \end{bmatrix} S_k^{-1} = \breve{Q}_k^{-1} \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \right) \quad (34)$$

whose lower block giv es $T_k + Q_k A_k = \mathbf{0}$ or $T_k^T = -A_k^T Q_k^T$.

T o eliminate μ_k in (33) without affecting the upper part of $\breve{C}_k^T$, introducing the matrices $B_k \in \mathcal{R}^{N\times 1}$ and $F_k \in \mathcal{R}$ such that

$$\breve{Q}_k \breve{B}_k = \breve{Q}_k \begin{bmatrix} B_k \\ F_k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \quad \left(\breve{B}_k = \begin{bmatrix} B_k \\ F_k \end{bmatrix} \right) \quad (35)$$

and then subtracting $\breve{B}_k {F_k}^{-1} \mu_k$ from the partition of $\breve{K}_k$ in (33), we hav e

$$\begin{aligned} \breve{K}_k - \breve{B}_k {F_k}^{-1}\mu_k &= \begin{bmatrix} m_k \\ \mu_k \end{bmatrix} - \begin{bmatrix} B_k {F_k}^{-1} \\ 1 \end{bmatrix} \mu_k \\ &= \begin{bmatrix} m_k - B_k {F_k}^{-1}\mu_k \\ \mathbf{0} \end{bmatrix}. \end{aligned} \quad (36)$$

Moreov er, multiplying the left-hand side of (36) by $\breve{Q}_k$ from the left, we can arrange it as

$$\begin{aligned} &\breve{Q}_k(\breve{K}_k - \breve{B}_k {F_k}^{-1}\mu_k) \\ &= \breve{Q}_k \breve{K}_k - \breve{Q}_k \breve{B}_k {F_k}^{-1}\mu_k = \breve{C}_k^T - \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} {F_k}^{-1}\mu_k \\ &= \breve{C}_k^T - \begin{bmatrix} \mathbf{0} \\ {F_k}^{-1}\mu_k \end{bmatrix}. \end{aligned} \quad (37)$$

By substituting (36) in to the abo veleft-hand side, the equation (30) of interest is presented by

$$\begin{aligned} &\breve{Q}_k(\breve{K}_k - \breve{B}_k {F_k}^{-1}\mu_k) \\ &\quad = \breve{C}_k^T - \begin{bmatrix} \mathbf{0} \\ F_k^{-1}\mu_k \end{bmatrix}, \\ &\begin{bmatrix} Q_{k+1} & \underline{T}_k^T \\ \underline{T}_k & \underline{M}_k \end{bmatrix} \begin{bmatrix} m_k - B_k {F_k}^{-1}\mu_k \\ \mathbf{0} \end{bmatrix} \\ &\quad = \begin{bmatrix} {C_{k+1}}^T \\ {c_{k-N}}^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -F_k^{-1}\mu_k \end{bmatrix} \end{aligned} \quad (38)$$

which is the same form as (29). The upper block of (38) leads to

$$Q_{k+1}(m_k - B_k {F_k}^{-1}\mu_k) = C_{k+1}^T. \quad (39)$$

With comparison of (23) and (39), w ereach the update equation of the gain matrix K_k. ■

C. Determination of Unknown Matrices Appear edin the Update of Gain Matrix

It now remains to determine the matrices A_k, S_k, and $D_k = B_k {F_k}^{-1}$, which are defined by (34) and (35). The determination of these matrices will be also carried out by means of induction.

L emma 4: The matrices A_k and S_k are recursively computed as

$$A_k = A_{k-1} - K_k W_k [c_k + C_k A_{k-1}] \in \mathcal{R}^{N\times 1} \quad (40)$$

$$S_k = \rho S_{k-1} + [c_k^T + A_k^T C_k^T] W_k [c_k + C_k A_{k-1}] \in \mathcal{R} \quad (41)$$

where $A_{-1} = \mathbf{0}$ and $S_{-1} = \rho/\varepsilon_0$.

Pro of: Now, recalling that the matrices A_{k-1} and S_{k-1}, by definition, satisfy

$$\breve{Q}_{k-1} \begin{bmatrix} 1 \\ A_{k-1} \end{bmatrix} = \begin{bmatrix} S_{k-1} \\ \mathbf{0} \end{bmatrix} \quad (42)$$

and using

$$\breve{Q}_k = \rho \breve{Q}_{k-1} + \breve{C}_k^T W_k \breve{C}_k \quad (43)$$

we obtain

$$\begin{aligned} &\breve{Q}_k \begin{bmatrix} 1 \\ A_{k-1} \end{bmatrix} \\ &= \rho \breve{Q}_{k-1} \begin{bmatrix} 1 \\ A_{k-1} \end{bmatrix} + \breve{C}_k^T W_k [c_k + C_k A_{k-1}] \\ &= \begin{bmatrix} \rho S_{k-1} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} c_k^T \\ C_k^T \end{bmatrix} W_k [c_k + C_k A_{k-1}]. \end{aligned} \quad (44)$$

Whereas, multiplying both sides of (28) by $\boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}]$ yields

$$\breve{\boldsymbol{Q}}_k \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{K}_k \end{bmatrix} \boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}] = \begin{bmatrix} \boldsymbol{\alpha}_k \\ \boldsymbol{C}_k^T \end{bmatrix} \boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}]. \quad (45)$$

Subtracting (45) from (44) on both sides, we find that

$$\begin{aligned} &\breve{\boldsymbol{Q}}_k \left[\begin{bmatrix} 1 \\ \boldsymbol{A}_{k-1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{K}_k \end{bmatrix} \boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}] \right] \\ &= \begin{bmatrix} \rho S_{k-1} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_k^T \\ \boldsymbol{C}_k^T \end{bmatrix} \boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}] \\ &- \begin{bmatrix} \boldsymbol{\alpha}_k^T \\ \boldsymbol{C}_k^T \end{bmatrix} \boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}], \\ &\breve{\boldsymbol{Q}}_k \begin{bmatrix} 1 \\ \boldsymbol{A}_{k-1} - \boldsymbol{K}_k\boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}] \end{bmatrix} \\ &= \begin{bmatrix} \rho S_{k-1} + [\boldsymbol{c}_k^T - \boldsymbol{\alpha}_k^T]\boldsymbol{W}_k[\boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1}] \\ \boldsymbol{0} \end{bmatrix}. \quad (46) \end{aligned}$$

Then, comparing the above equation with (34), we obtain the recursions of (40) and (41) because $\boldsymbol{\alpha}_k^T = \boldsymbol{T}_k^T\boldsymbol{K}_k = -\boldsymbol{A}_k^T\boldsymbol{C}_k^T$ holds. ■

Next, the matrix $\boldsymbol{D}_k = \boldsymbol{B}_kF_k^{-1}$ is determined instead of $\boldsymbol{B}_k$ and F_k since they are used only in such a combination.

Lemma 5: The matrix $\boldsymbol{D}_k = \boldsymbol{B}_kF_k{}^{-1}$ is recursively determined as

$$\boldsymbol{D}_k = [\boldsymbol{D}_{k-1} - \boldsymbol{m}_k\boldsymbol{W}_k\boldsymbol{\eta}_k][1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\boldsymbol{\eta}_k]^{-1} \in \mathcal{R}^{N\times 1} \quad (47)$$

and F_k is evolved by

$$F_k = F_{k-1}[1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\boldsymbol{\eta}_k]/\rho \quad \in \mathcal{R} \quad (48)$$

where $\boldsymbol{\eta}_k = \breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1} = \boldsymbol{c}_{k-N} + \boldsymbol{C}_{k+1}\boldsymbol{D}_{k-1}$ and $\boldsymbol{D}_{-1} = \boldsymbol{0}$.

Proof: To update $\boldsymbol{B}_k$ and F_k, using

$$\breve{\boldsymbol{Q}}_{k-1}\breve{\boldsymbol{B}}_{k-1} = \breve{\boldsymbol{Q}}_{k-1} \begin{bmatrix} \boldsymbol{B}_{k-1} \\ F_{k-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} \quad (49)$$

and (43), we find that

$$\begin{aligned} \breve{\boldsymbol{Q}}_k\breve{\boldsymbol{B}}_{k-1} &= \rho\breve{\boldsymbol{Q}}_{k-1}\breve{\boldsymbol{B}}_{k-1} + \breve{\boldsymbol{C}}_k^T\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1} \\ &= \rho \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} + \breve{\boldsymbol{C}}_k^T\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1}. \quad (50) \end{aligned}$$

To change the above into the same form as (49), subtracting $\breve{\boldsymbol{C}}_k^T\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1}$ from both sides of (50), we eliminate the last term on the above right-hand side as

$$\begin{aligned} &\breve{\boldsymbol{Q}}_k\breve{\boldsymbol{B}}_{k-1} - \breve{\boldsymbol{C}}_k^T\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1} \\ &= \breve{\boldsymbol{Q}}_k\breve{\boldsymbol{B}}_{k-1} - \breve{\boldsymbol{Q}}_k\breve{\boldsymbol{K}}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1} = \rho \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix}, \\ &\breve{\boldsymbol{Q}}_k[\breve{\boldsymbol{B}}_{k-1} - \breve{\boldsymbol{K}}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1}] \\ &= \rho \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix}. \quad (51) \end{aligned}$$

Then, comparing the last equation of (51) with (35), we achieve the recursion of $\breve{\boldsymbol{B}}_k$:

$$\breve{\boldsymbol{B}}_k = (\breve{\boldsymbol{B}}_{k-1} - \breve{\boldsymbol{K}}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1})/\rho \quad (52)$$

which updates $\boldsymbol{B}_k$ and F_k iteratively.

However, since the rows of $\breve{\boldsymbol{B}}_k$ are used only in the combination such that $\boldsymbol{D}_k = \boldsymbol{B}_kF_k^{-1} \in \mathcal{R}^{N\times 1}$, it is probably more convenient to rewrite (35) and (52) in terms of

$$\boldsymbol{D}_k = \boldsymbol{B}_kF_k^{-1}, \ \ \breve{\boldsymbol{D}}_k = \breve{\boldsymbol{B}}_kF_k^{-1} = \begin{bmatrix} \boldsymbol{D}_k \\ 1 \end{bmatrix} \quad (53)$$

which obeys

$$\begin{aligned} \breve{\boldsymbol{Q}}_k\breve{\boldsymbol{D}}_k &= \breve{\boldsymbol{Q}}_k\breve{\boldsymbol{B}}_kF_k^{-1} = \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} F_k^{-1}, \\ \breve{\boldsymbol{Q}}_k \begin{bmatrix} \boldsymbol{D}_k \\ 1 \end{bmatrix} &= \begin{bmatrix} \boldsymbol{0} \\ F_k^{-1} \end{bmatrix}. \quad (54) \end{aligned}$$

Then, multiplying both sides of (51) by F_{k-1}^{-1}, we have

$$\begin{aligned} &\breve{\boldsymbol{Q}}_k[\breve{\boldsymbol{B}}_{k-1}F_{k-1}^{-1} - \breve{\boldsymbol{K}}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{B}}_{k-1}F_{k-1}^{-1}] \\ &= \breve{\boldsymbol{Q}}_k[\breve{\boldsymbol{D}}_{k-1} - \breve{\boldsymbol{K}}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1}] \\ &= \begin{bmatrix} \boldsymbol{0} \\ \rho F_{k-1}^{-1} \end{bmatrix}. \quad (55) \end{aligned}$$

Furthermore, using $\breve{\boldsymbol{D}}_{k-1} = \breve{\boldsymbol{B}}_{k-1}F_{k-1}^{-1}$, it is arranged as

$$\begin{aligned} \breve{\boldsymbol{Q}}_k \left[\breve{\boldsymbol{D}}_{k-1} - \begin{bmatrix} \boldsymbol{m}_k \\ \boldsymbol{\mu}_k \end{bmatrix} \boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1} \right] &= \begin{bmatrix} \boldsymbol{0} \\ \rho F_{k-1}^{-1} \end{bmatrix}, \\ \breve{\boldsymbol{Q}}_k \begin{bmatrix} \boldsymbol{D}_{k-1} - \boldsymbol{m}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1} \\ 1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{0} \\ \rho F_{k-1}^{-1} \end{bmatrix}. \quad (56) \end{aligned}$$

Consequently, multiplying the last equation of (56) with $[1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1}]^{-1}$ from the right, we find that

$$\begin{aligned} &\breve{\boldsymbol{Q}}_k \begin{bmatrix} [\boldsymbol{D}_{k-1} - \boldsymbol{m}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1}][1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1}]^{-1} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{0} \\ \rho F_{k-1}^{-1}[1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\breve{\boldsymbol{C}}_k\breve{\boldsymbol{D}}_{k-1}]^{-1} \end{bmatrix}. \end{aligned}$$

Comparing the above equation with (54) yields the update formulas of $\boldsymbol{D}_k$ and F_k. ■

D. A Fast H_∞ *Filter*

As a conclusion, we may summarize a fast algorithm of the H_∞ filter.

Theorem 2: A fast algorithm of the modified H_∞ filter is carried out with computational complexity of $\mathcal{O}(N)$ per time step as follows.

[Step 0] Take the initial conditions for the recursions as

$$\boldsymbol{K}_0 = \boldsymbol{0}, \ \boldsymbol{A}_{-1} = \boldsymbol{0}, \ S_{-1} = \frac{\rho}{\varepsilon_0}, \ \ \boldsymbol{D}_{-1} = \boldsymbol{0}, \ \hat{\boldsymbol{x}}_{0|0} = \boldsymbol{0}$$

where ε_0 is a sufficiently large positive number.

[Step 1] Determine $\boldsymbol{A}_k$ and S_k recursively as

$$\begin{aligned}\tilde{\boldsymbol{e}}_k &= \boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_{k-1} && \in \boldsymbol{\mathcal{R}}^{2\times 1}\\ \boldsymbol{A}_k &= \boldsymbol{A}_{k-1} - \boldsymbol{K}_k\boldsymbol{W}_k\tilde{\boldsymbol{e}}_k && \in \boldsymbol{\mathcal{R}}^{N\times 1}\\ \boldsymbol{e}_k &= \boldsymbol{c}_k + \boldsymbol{C}_k\boldsymbol{A}_k && \in \boldsymbol{\mathcal{R}}^{2\times 1}\\ S_k &= \rho S_{k-1} + \boldsymbol{e}_k^T\boldsymbol{W}_k\tilde{\boldsymbol{e}}_k && \in \boldsymbol{\mathcal{R}}.\end{aligned}$$

[Step 2] Calculate $\breve{\boldsymbol{K}}_k$ as

$$\breve{\boldsymbol{K}}_k = \begin{bmatrix} S_k^{-1}\boldsymbol{e}_k^T \\ \boldsymbol{K}_k + \boldsymbol{A}_k S_k^{-1}\boldsymbol{e}_k^T \end{bmatrix} \in \boldsymbol{\mathcal{R}}^{(N+1)\times 2}.$$

[Step 3] Partition $\breve{\boldsymbol{K}}_k$ as

$$\breve{\boldsymbol{K}}_k = \begin{bmatrix} \boldsymbol{m}_k \\ \boldsymbol{\mu}_k \end{bmatrix}, \quad \boldsymbol{m}_k \in \mathcal{R}^{N\times 2}, \ \boldsymbol{\mu}_k \in \mathcal{R}^{1\times 2}.$$

[Step 4] Determine $\boldsymbol{D}_k = \boldsymbol{B}_k F_k^{-1}$ and then obtain the filter gain $\boldsymbol{K}_{s,k+1}$ at next time step $k+1$, through the gain matrix $\boldsymbol{K}_{k+1}$, as

$$\begin{aligned}\boldsymbol{\eta}_k &= \boldsymbol{c}_{k-N} + \boldsymbol{C}_{k+1}\boldsymbol{D}_{k-1}\\ \boldsymbol{D}_k &= [\boldsymbol{D}_{k-1} - \boldsymbol{m}_k\boldsymbol{W}_k\boldsymbol{\eta}_k][1 - \boldsymbol{\mu}_k\boldsymbol{W}_k\boldsymbol{\eta}_k]^{-1}\\ \boldsymbol{K}_{k+1} &= \boldsymbol{m}_k - \boldsymbol{D}_k\boldsymbol{\mu}_k\\ \tilde{K}_{k+1}(i) &= \rho K_{k+1}(i,1), \quad i = 1,\cdots,N\\ \boldsymbol{K}_{s,k+1} &= G_{k+1}^{-1}\tilde{\boldsymbol{K}}_{k+1}, \quad G_{k+1} = \rho + \gamma_f^{-2}\boldsymbol{H}_{k+1}\tilde{\boldsymbol{K}}_{k+1}\end{aligned}$$

where $\boldsymbol{\eta}_k \in \boldsymbol{\mathcal{R}}^{2\times 1}$, $\boldsymbol{D}_k \in \boldsymbol{\mathcal{R}}^{N\times 1}$, $\boldsymbol{K}_{k+1} \in \boldsymbol{\mathcal{R}}^{N\times 2}$, $\boldsymbol{K}_{s,k+1} \in \boldsymbol{\mathcal{R}}^{N\times 1}$, $0 < \rho = 1 - \gamma_f^{-2} \leq 1$, and $\gamma_f > 1$.

[Step 5] Update the filter equation of the H_∞ filter as

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{K}_{s,k+1}(y_{k+1} - \boldsymbol{H}_{k+1}\hat{\boldsymbol{x}}_{k|k}).$$

[Step 6] Increase time step k ($k \to k+1$), and return to Step 1.

Proof: It follows from *lemma 1* to *lemma 5* that the above results are immediately proved. ■

The derived fast algorithm has the following good asymptotical property.

Corollary 1: The fast algorithm when $\gamma_f = \infty$ agrees with that of the Kalman filter [3].

Proof: Setting $\gamma_f = \infty$, we find that the statement is clear from the H_∞ criterion in (4) or the algorithm described in *Theorem 2*. ■

This means that the fast algorithm derived in this section could be regarded as an extended version of the fast Kalman filter in the sense of H_∞ optimization.

E. Existence of Fast H_∞ *Filter*

In general, it requires a considerably computational cost to check the positivity condition of (10), especially for large N. Therefore, it may be more favorable in this case to use the alternative existence condition that the matrices $\boldsymbol{R}_k$ and $\boldsymbol{R}_{e,k}$ have the same inertia [8]. Here, what one means by the inertia of a matrix is the number of its positive, negative, and zero eigenvalues.

Fortunately the inertia condition for the fast H_∞ filter to exist can be reduced to a simpler form.

Lemma 6: The following condition allows a fair judgment for existence of the fast H_∞ filter at the computational requirement of $\mathcal{O}(N)$ per time step.

[Existence Condition]

$$-\varrho\hat{\Xi}_i + \rho\gamma_f^2 > 0, \quad i = 0,\cdots,k \tag{57}$$

where

$$\varrho = 1 - \gamma_f^2, \quad \hat{\Xi}_i = \frac{\boldsymbol{H}_i\tilde{\boldsymbol{K}}_i}{1 - \boldsymbol{H}_i\tilde{\boldsymbol{K}}_i} \tag{58}$$

Proof: Solving the characteristic equation $|\lambda\boldsymbol{I} - \boldsymbol{R}_{e,k}| = 0$ of the 2×2 matrix $\boldsymbol{R}_{e,k}$, we obtain the eigenvalues λ_i of $\boldsymbol{R}_{e,k}$ as

$$\lambda_i = \frac{\boldsymbol{\Phi} \pm \sqrt{\boldsymbol{\Phi}^2 - 4\rho\varrho\boldsymbol{H}_k\hat{\boldsymbol{\Sigma}}_{k|k-1}\boldsymbol{H}_k^T + 4\rho^2\gamma_f^2}}{2}$$

where $\boldsymbol{\Phi} = 2\boldsymbol{H}_k\hat{\boldsymbol{\Sigma}}_{k|k-1}\boldsymbol{H}_k^T + \rho\varrho$ and $\varrho = 1-\gamma_f^2$. If and only if $-4\rho\varrho\boldsymbol{H}_k\hat{\boldsymbol{\Sigma}}_{k|k-1}\boldsymbol{H}_k^T + 4\rho^2\gamma_f^2 > 0$, one of the two eigenvalues of $\boldsymbol{R}_{e,k}$ becomes positive and the another negative, so that $\boldsymbol{R}_k$ and $\boldsymbol{R}_{e,k}$ have the same inertia. Hence, using $\boldsymbol{H}_k\hat{\boldsymbol{\Sigma}}_{k|k-1}\boldsymbol{H}_k^T = \frac{\boldsymbol{H}_k\tilde{\boldsymbol{K}}_k}{1-\boldsymbol{H}_k\tilde{\boldsymbol{K}}_k}$, we attain the more simplified inertia condition of (57), where the calculation of $\boldsymbol{H}_k\tilde{\boldsymbol{K}}_k$ requires $\mathcal{O}(N)$ multiplications per time step. ■

IV. Conclusions

A fast algorithm of H_∞ filters with a modified H_∞ boundness has been successfully derived in a recursive fashion using the shifting property of the sequences of the observation matrix $\boldsymbol{H}_k$. In addition, it was clarified that the computational complexity of the fast H_∞ filter is of $\boldsymbol{O}(N)$ per time step which is equal to that of the fast Kalman filter and the LMS algorithm, dramatically reducing the complexity, $\boldsymbol{O}(N^2)$, of the corresponding H_∞ filter. Furthermore, a simple method to judge the validity of the fast H_∞ filter for a given γ_f was developed, which requires $\boldsymbol{O}(N)$ arithmetic operations per time step.

References

[1] A.H. Jazwinski, Stochastic Processes and Filtering Theory, Academic Press, 1970.

[2] B.D.O. Anderson and J.B. Moore, Optimal Filtering, Prentice-Hall, 1979.

[3] L. Ljung, M. Morf and D. Falconer: "Fast Calculation of Gain Matrices for Recursive Estimation Schemes", International Journal of Control, 27, 1, pp.1-19, 1978.

[4] L. Ljung: System Identification: Theory for the User, Prentice-Hall, New York, 1987.

[5] K. M. Nagpal, P.P. Khargonekar: "Filtering and Smoothing in an H_∞ Setting," IEEE Trans. Automatic Control, 36, 2, pp.152-166, 1991.

[6] U. Shaked and Y. Theodor: "H_∞-Optimal Estimation: A Tutorial," IEEE Proc. of the 31st CDC, pp.2278-2286, 1992.

[7] I. Yaesh and U. Shaked: "Game Theory Approach to Optimal Linear State Estimation and Its Relation to the Minimum H_∞-Norm Estimation," IEEE Trans. Automatic Control, 37, 6, pp.828-831, 1992.

[8] B. Hassibi, A. H. Sayed, and T. Kailath: Indefinite-Quadratic Estimation and Control, 1999.

[9] K. Nishiyama:" Robust Estimation of a Single Complex Sinusoid in White Noise: H_∞ Filtering Approach," IEEE Trans. Signal Processing, 47, 10, pp.2853-2856, 1999.