RECURSIVE 4SID-BASED IDENTIFICATION ALGORITHM WITH FIXED INPUT-OUTPUT DATA SIZE

KENTARO KAMEYAMA, AKIRA OHSUMI, YÜJI MATSUIURA* AND KATSUMINE SAWADA
Department of Mechanical & System Engineering, Graduate School of Science and Technology
Kyoto Institute of Technology
Matsugasaki, Sakyo, Kyoto 606-8585, Japan
{kame02, ohsumi}@ipc.kit.ac.jp

Received September 2004; revised December 2004

ABSTRACT. The 4SID method belongs to a class of identification methods which determine unknown system parameters as well as system order from input and output data. Assuming that the mathematical model of unknown system is described by a discrete/continuous-time time-invariant state space linear system, several 4SID-based algorithms have been developed up to now; however, most of these seem to point batch processing. Recursive algorithms are urgently expected from the viewpoint of reducing computational burden and storage cost. In this paper, a new recursive 4SID-based algorithm is derived by keeping the size of input and output data to be constant. Its derivation is direct and rather simple compared with any other recursive ones, and the algorithm is called the bona fide recursive algorithm. Efficiency of the algorithm is demonstrated by comparing numerical results with that obtained by existing methods.

Keywords: Subspace identification, Recursive algorithm, Identification algorithm, Stochastic system

1. Introduction. During the last two decades, subspace-based system identification (4SID) methods have attracted a great deal of interest in control community and been particularly developed (e.g., [1-3]), because they can identify system matrices of the state space model directly from the input and output data. The primary surge of interest has been concentrated mainly on the time-invariant linear systems. In reality, however, most existing systems show time-varying and/or nonlinear behavior. The nonlinear systems are sometimes treated as higher-order linear time-varying systems from the practical point of view. As well known, subspace methods are based on robust numerical tools such as LQ-factorization and singular value decomposition (SVD). Although these tools are appropriate for batch processing, they are inappropriate for on-line identification because of their computational burden and storage costs. From the standpoint of computational burden for such systems as mentioned above, on-line subspace identification consisting of recursive algorithm is intensely desirable.

So far, several challenging works have been done in deriving recursive subspace algorithms. As mentioned above the basic tools for subspace identification are the LQ-factorization and the SVD; so, in order to lighten the computational burden as well as
the memory storage, there exist mainly three possibilities: recursion of \(LQ\)-factorization, recursion of SVD, or recursion of both these two.

For instance, Verhaegen and Deprettere [4] presented a fast recursive implementation for MOESP algorithm and applied it to the identification of slowly time-varying systems. The core of the algorithm is based on the partial update of the \(LQ\)-factorization followed by a rank-one update of a SVD step. Under the assumption that the system order is a priori known, Gustafsson [5] proposed a recursive algorithm by updating directly the extended observability matrix. Further, Lovera, Gustafsson and Verhaegen [6] and Lovera [7] proposed recursive algorithms based on the instrumental variable and the subspace tracking idea used in array signal processing for the update of the SVD. Instead of direct estimation of the extended observability matrix, Takei, Imai and Wada [8] proposed a recursive algorithm for updating the SVD of the Schur complement submatrix extracted from the extended observability matrix. Oku and Kimura [9, 10] proposed an algorithm by updating the projection matrix called the compressed input-output data matrix, and they also gave a method using gradient type subspace tracking. Inspired from the propagator method, Mercère, Lecoeuche and Vasseur [11] derived also a recursive algorithm. In all algorithms using subspace tracking idea the system order seems to be assumed known a priori.

Originally, the 4SID algorithm has been developed for time-invariant systems. However, it will be expected inherently that recursive algorithm has potential ability to be used in identification of time-varying systems, because the major mission of recursive algorithm is to implement on-line. Unfortunately, most of recent researches on recursive algorithm for subspace identification have been developed mainly for time-invariant systems. It should be recalled that the basic input-output algebraic equation in 4SID (Eq. (1) below) holds approximately for time-varying systems if we assume that the system matrices change slowly with time [4, 12]. Although many researches derive recursive algorithms for time-invariant systems in theory, they are applied and tested for time-varying system identification by numerical simulations (for instance, see [5, 8]).

In order to cope with time-varying systems the idea of forgetting factor is often employed in constructing the input and output data matrices in order to fade the influence of old data to the updated input and output data. Instead, in this paper the size of the data matrices is fixed a priori and a recursive identification algorithm is derived by performing the \(LQ\)-factorization recursively. Using modest size of input and output data in recursive identification will not cause so serious disadvantage, because the recent computer technology makes rapidly progress in computational speed and in memory storage. The accuracy and properties of proposed recursive algorithm will be compared with one of other typical methods by performing some numerical simulations.

2. Input-Output Algebraic Relationship. In the subspace identification context, the mathematical model of unknown system is assumed to be given by the (continuous/discrete-time) linear state space model, and the input-output algebraic relationship is derived:

\[
Y_i = \Gamma_i X_i + H_i U_i + \Sigma_i W_i + V_i,
\]  

(1)

where \(Y_i \in R^{d \times N}\) is output data matrix which are constructed by arranging properly \(N\) (column) vectors of the output \(y \in R^d\); \(X_i \in R^{n \times N}\) is the matrix constructed by the system states \(x \in R^n\); \(U_i \in R^{m \times N}\), \(W_i \in R^{n \times N}\) and \(V_i \in R^{d \times N}\) are constructed similarly.
to $Y_i$ from the input, system and observation noises $u \in \mathbb{R}^m$, $w \in \mathbb{R}^m$, $v \in \mathbb{R}^t$ respectively; $I_i \in \mathbb{R}^{t \times n}$ is the extended observability matrix, and $H_i \in \mathbb{R}^{t \times im}$, $\Sigma_i \in \mathbb{R}^{t \times im}$ are lower block triangular matrices consisting of system matrices (for detail structures, see e.g. [1, 13]).

The input-output algebraic equation (1) can be obtained from the discrete- or continuous-time, time-invariant/varying linear system. For instance, when the system is described by the discrete-time time-invariant stochastic systems subjected to white Gaussian noises $w_k$ and $v_k$ with zero-means:

$$\Sigma_D : \begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k = Cx_k + Du_k + v_k, \end{cases}$$

(2)

then, using $(N+i−1)$ past and current data, the matrix $Y_i$ is constructed by the Hankel matrix,

$$Y_i = [y_i(k-N+1), \cdots, y_i(k-1), y_i(k)]$$

where $y_i(j) = [y^T_j, \cdots, y^T_{j-1}, y^T_j]^T (j = k-N+1, \cdots, k-1, k)$ is the $i\ell$-vector; and

$$X_i = [x_{k-N-i+2}, \cdots, x_{k-i}, x_{k-i+1}].$$

On the other hand, if we employ the distribution-based subspace identification [13], the continuous-time stochastic systems subjected to zero-mean white Gaussian random noises $w(t)$ and $v(t)$

$$\Sigma_C : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + w(t) \\ y(t) = Cx(t) + Du(t) + v(t) \end{cases}$$

(3)

lead us to (1) in which the output matrix $Y_i$ denoted by $Y_i(\varphi)$ (indicating the used $C^\infty$-class test function $\varphi(t; \cdot)$) is constructed by

$$Y_i(\varphi) = [y_i(\varphi)(t_{k-N+1}), \cdots, y_i(\varphi)(t_{k-1}), y_i(\varphi)(t_k)]$$

where $y_i(\varphi)(t_j) = [y^T(\varphi)(t_j), \cdots, (-1)^{i-1}y^T(\varphi^{(i-1)})(t_j)]^T$, $y(\varphi)(t_j) := \int_{-\infty}^{\infty} y(t)\varphi(t; t_j)dt$, $(\{t_j\}_{j=k-N+1, \cdots, k-1, k})$ are (not necessarily equidistant) different time instants;

$$X_i(\varphi) = [x(\varphi)(t_{k-N+1}), \cdots, x(\varphi)(t_{k-1}), x(\varphi)(t_k)],$$

and other matrices are defined similarly.

For the purpose of deriving recursive subspace identification algorithms, let $t_k$ or $k$ be the current time instant or step for $\Sigma_C$ or $\Sigma_D$, respectively, and let the integer $N$ be fixed. Since no confusion will arise, we write (1) for both cases as follows:

$$Y_i(k|k-N+1) = I_i(k)X_i(k|k-N+1) + H_i(k)U_i(k|k-N+1) + \Sigma_i(k)W_i(k|k-N+1) + V_i(k|k-N+1),$$

(4)

where the arguments $k-N+1$ and $k$ denote the first or the last (current) time step of the $N$ column vectors, respectively; viz.

$$Y_i(k|k-N+1) = \begin{cases} [y_i(k-N+1), \cdots, y_i(k-1), y_i(k)] & \text{for } \Sigma_D \\ [y_i(\varphi)(t_{k-N+1}), \cdots, y_i(\varphi)(t_{k-1}), y_i(\varphi)(t_k)] & \text{for } \Sigma_C. \end{cases}$$

For matrices $I_i$, $H_i$, and $\Sigma_i$ the argument $k$ is also introduced in order to cope with time-varying system matrices $(A(k), B(k), C(k), D(k))$ or $(A(t), B(t), C(t), D(t)).$
Then, our problem is to derive the recursive algorithm for determining the quadruple system matrices (within a similarity transformation) and the system order \( n \).

3. Main Result: Recursive Algorithm. Let \( U_h(k|k - N + 1) \in R^{hm \times N} \) be an instrumental variable matrix different from the matrix \( U_i(k|k - N + 1) \) and constructed from the input data such as

\[
U_h(k|k - N + 1) = [u_h(k - i - N + 1), \ldots, u_h(k - i)] \quad \text{for } \Sigma_D
\]

\[
= [u_h(\psi)(t_{k-N+1}), \ldots, u_h(\psi)(t_k)] \quad \text{for } \Sigma_C.
\]

Then, perform the \( LQ \)-factorization with the following partitioning:

\[
\begin{bmatrix}
U_i(k|k - N + 1) \\
U_h(k|k - N + 1) \\
Y_i(k|k - N + 1)
\end{bmatrix} = \begin{bmatrix}
L_{11}(k) & 0 & 0 \\
L_{21}(k) & L_{22}(k) & 0 \\
L_{31}(k) & L_{32}(k) & L_{33}(k)
\end{bmatrix} \begin{bmatrix}
Q_1^T(k) \\
Q_2^T(k) \\
Q_3^T(k)
\end{bmatrix}.
\]

The assumption that the size of \( i \) and \( h \) are selected as \( i, h > n \) is made tacitly. As is well known this \( LQ \)-factorization is critical to determine the column space of the extended observability matrix \( \Gamma_i(k) \), since for sufficiently large \( N \) the relation

\[
\frac{1}{\sqrt{N}} L_{32}(k) = \Gamma_i(k) \frac{1}{\sqrt{N}} X_i(k|k - N + 1)Q_2(k)
\]

holds. So, by means of the eigenvalue decomposition of the matrix \( L_{32}(k)L_{32}^T(k) \), the pair of matrices \( (A_T, C_T) \) can be determined (where the suffix \( T \) indicates the similarity transformation by a nonsingular matrix \( T \)). Furthermore, the equation

\[
\{U_n^+(k)\}^T L_{31}(k)L_{11}^{-1}(k) = \{U_n^-(k)\}^T H_i(k),
\]

where \( U_n^+(k) \) is the orthonormal complement of the estimates of \( \Gamma_i(k) \) is derived. From the matrices \( L_{31}(k) \) and \( L_{11}(k) \) the pair of \( (B_T, D) \) can be also determined (where \( A_T = TAT^{-1}, B_T = TB, C_T = CT^{-1} \)). Thus, estimates of unknown system matrices are derived if \( L \)-factors of \( LQ \)-factorization are renewed recursively.

Given a set of new input and output data \( \{u_{k+1}, y_{k+1}\} \) or \( \{u(t_{k+1}), y(t_{k+1})\} \), construct vectors \( u_i(k + 1), u_h(k + 1) \) and \( y_i(k + 1) \) as updated data. Then, we have the relation:

\[
\begin{bmatrix}
U_i(k|k - N + 1) : u_i(k + 1) \\
U_h(k|k - N + 1) : u_h(k + 1) \\
Y_i(k|k - N + 1) : y_i(k + 1)
\end{bmatrix} = \begin{bmatrix}
u_i(k - N + 1) : U_i(k + 1|k - N + 2) \\
u_h(k - N + 1) : U_h(k + 1|k - N + 2) \\
y_i(k - N + 1) : Y_i(k + 1|k - N + 2)
\end{bmatrix}.
\]

This is the key to derive recursive identification algorithm in this paper.

Now, note that the \( LQ \)-factorization of the \((im + hm + il) \times N\) data matrix at the step \( k + 1 \) is given by

\[
\begin{bmatrix}
U_i(k + 1|k - N + 2) \\
U_h(k + 1|k - N + 2) \\
Y_i(k + 1|k - N + 2)
\end{bmatrix} = \begin{bmatrix}
L_{11}(k + 1) & 0 & 0 \\
L_{21}(k + 1) & L_{22}(k + 1) & 0 \\
L_{31}(k + 1) & L_{32}(k + 1) & L_{33}(k + 1)
\end{bmatrix} \begin{bmatrix}
Q_1^T(k + 1) \\
Q_2^T(k + 1) \\
Q_3^T(k + 1)
\end{bmatrix}.
\]
Substituting the $LQ$-factorizations of (5) and (9) into the left- and right-hand sides of (8) respectively, we have

$$\begin{bmatrix}
L_{11}(k)Q_1^T(k) & : u_i(k + 1) \\
L_{21}(k)Q_1^T(k) + L_{22}(k)Q_2^T(k) & : u_h(k + 1) \\
L_{31}(k)Q_1^T(k) + L_{32}(k)Q_2^T(k) + L_{33}(k)Q_3^T(k) & : y_i(k + 1)
\end{bmatrix} = \begin{bmatrix}
u_i(k - N + 1) \\
u_h(k - N + 1) \\
y_i(k - N + 1)
\end{bmatrix}$$

$$L_{11}(k + 1)Q_1^T(k + 1) + L_{21}(k + 1)Q_1^T(k + 1) + L_{22}(k + 1)Q_2^T(k + 1) + L_{31}(k + 1)Q_1^T(k + 1) + L_{32}(k + 1)Q_2^T(k + 1) + L_{33}(k + 1)Q_3^T(k + 1). \quad (10)$$

Regard (10) as $[A:b] = [c:D]$. Then the relation $[A:b][A:b]^T = [c:D][c:D]^T$ gives $AA^T + bb^T = cc^T + DD^T$. So, noting that $Q_i^T(\cdot)Q_j(\cdot) = I\delta_{ij}$ ($\delta_{ij}$ : Kronecker delta), we have for the L.H.S.

$$\begin{bmatrix}
L_{11}(k)Q_1^T(k) & : u_i(k + 1) \\
L_{21}(k)Q_1^T(k) + L_{22}(k)Q_2^T(k) & : u_h(k + 1) \\
L_{31}(k)Q_1^T(k) + L_{32}(k)Q_2^T(k) + L_{33}(k)Q_3^T(k) & : y_i(k + 1)
\end{bmatrix}^T$$

$$= \begin{bmatrix}
L_{11}(k)L_{11}^T(k) \\
L_{21}(k)L_{11}^T(k) + L_{22}(k)L_{21}^T(k) + L_{22}(k)L_{22}^T(k) \\
L_{31}(k)L_{11}^T(k) + L_{32}(k)L_{21}^T(k) + L_{33}(k)L_{31}^T(k) + L_{32}(k)L_{32}^T(k) + L_{33}(k)L_{33}^T(k)
\end{bmatrix}$$

$$+ \begin{bmatrix}
u_i(k + 1)u_i^T(k + 1) \\
u_h(k + 1)u_h^T(k + 1) \\
y_i(k + 1)y_i^T(k + 1)
\end{bmatrix} \quad (11)$$

Similarly, we have for the R.H.S. that

$$\begin{bmatrix}
u_i(k-N+1)u_i^T(k-N+1) \\
u_h(k-N+1)u_h^T(k-N+1) \\
y_i(k-N+1)y_i^T(k-N+1)
\end{bmatrix}$$

$$\begin{bmatrix}
L_{11}(k + 1)L_{11}^T(k + 1) \\
L_{21}(k + 1)L_{11}^T(k + 1) + L_{22}(k + 1)L_{21}^T(k + 1) + L_{22}(k + 1)L_{22}^T(k + 1) \\
L_{31}(k + 1)L_{11}^T(k + 1) + L_{32}(k + 1)L_{21}^T(k + 1) + L_{33}(k + 1)L_{31}^T(k + 1) + L_{32}(k + 1)L_{32}^T(k + 1) + L_{33}(k + 1)L_{33}^T(k + 1)
\end{bmatrix} \quad (12)$$
Similarly, using (13) and (14), we obtain from (16) the following equation:
\[
L_3(k) = L_{21}(k)L_{21}^T(k) + L_{22}(k)L_{22}^T(k)
\]
\[
L_4(k) = L_{31}(k)L_{21}^T(k) + L_{32}(k)L_{22}^T(k).
\]

Then, we have the following relations:
\[
\hat{L}_0(k+1) = \hat{L}_0(k) + u_i(k+1)u_i^T(k+1) - u_i(k - N + 1)u_i^T(k - N + 1)
\]
\[
\hat{L}_1(k+1) = \hat{L}_1(k) + u_h(k+1)u_h^T(k+1) - u_h(k - N + 1)u_h^T(k - N + 1)
\]
\[
\hat{L}_2(k+1) = \hat{L}_2(k) + y_i(k+1)u_i^T(k+1) - y_i(k - N + 1)u_i^T(k - N + 1)
\]
\[
\hat{L}_3(k+1) = \hat{L}_3(k) + u_h(k+1)u_h^T(k+1) - u_h(k - N + 1)u_h^T(k - N + 1)
\]
\[
\hat{L}_4(k+1) = \hat{L}_4(k) + y_i(k+1)u_i^T(k+1) - y_i(k - N + 1)u_i^T(k - N + 1).
\]

As mentioned before, in order to identify the system quadruple and the order, the matrix products \(L_{32}(k)L_{32}^T(k)\) and \(L_{31}(k)L_{11}^{-1}(k)\) are required. First, from (17) we have:
\[
L_{32}(k) = \{\hat{L}_4(k) - L_{31}(k)L_{21}^T(k)\}\{L_{22}^T(k)\}^{-1}.
\]

Extracting \(L_{31}(k)\) and \(L_{21}(k)\) from (15) and (14) respectively, we obtain
\[
L_{31}(k)L_{21}^T(k) = \hat{L}_2(k)\hat{L}_0^{-1}(k)\hat{L}_1^T(k).
\]

Similarly, using (13) and (14), we obtain from (16) the following equation:
\[
L_{22}(k)L_{22}^T(k) = L_3(k) - \hat{L}_1(k)\hat{L}_0^{-1}(k)\hat{L}_1^T(k).
\]

Therefore, constructing the product \(L_{32}(k)L_{32}^T(k)\) from (23), and then using (24) and (25), we have
\[
L_{32}(k)L_{32}^T(k) = \left\{\hat{L}_4(k) - \hat{L}_2(k)\hat{L}_0^{-1}(k)\hat{L}_1^T(k)\right\}
\]
\[
\cdot \left\{\hat{L}_3(k) - \hat{L}_1(k)\hat{L}_0^{-1}(k)\hat{L}_1^T(k)\right\}^{-1} \left\{\hat{L}_4(k) - \hat{L}_2(k)\hat{L}_0^{-1}(k)\hat{L}_1^T(k)\right\}^T.
\]

Furthermore, the product \(L_{31}(k)L_{11}^{-1}(k)\) can be readily obtained from (15) and the definition of \(\hat{L}_0(k)\) as
\[
L_{31}(k)L_{11}^{-1}(k) = \hat{L}_2(k)\hat{L}_0^{-1}(k).
\]

Consequently, the recursive identification algorithm is given as follows:

**Recursive subspace identification algorithm**

**Step 0:** (Preassignment) Fix the integers \(i, h\) and \(N\) appropriately (as user-defined parameters). After observing enough amount of data to construct data matrices \(U_i(0 \mid -N+1), U_h(0 \mid -N+1)\) and \(Y_i(0 \mid -N+1)\), perform once the \(LQ\)-factorization as (5) to obtain \(\{\hat{L}_j(0), j = 0, 1, \cdots, 4\}\).

**Step 1:** With newly constructed input and output vectors \(u_i(k+1), u_h(k+1)\) and \(y_i(k+1)\) at \((k + 1)\)th step, revise \(\{\hat{L}_j(k), j = 0, 1, \cdots, 4\}\) which is obtained in the previous step by (18)-(22).
Step 2: Compute $L_{32}(k+1)L_{32}^T(k+1)$ from (26).

Step 3: Perform the eigenvalue decomposition of $L_{32}(k+1)L_{32}^T(k+1)$ obtained in Step 2 as follows:

\[
L_{32}(k+1)L_{32}^T(k+1) = \begin{bmatrix} U_n(k+1) & U_n^+(k+1) \\ S_1^2(k+1) & 0 \\ 0 & S_2^2(k+1) \end{bmatrix} \begin{bmatrix} U_n^T(k+1) \\ \{U_n^+(k+1)\}^T \end{bmatrix}.
\] (28)

This gives the information about the system order $n$.

Step 4: From the knowledge of $U_n(k+1) \in R^{d \times n}$ obtained in Step 3, compute the pair $(A_T, C_T)$.

Step 5: Compute $L_{31}(k+1)L_{31}^T(k+1)$ from (27) and obtain the pair $(B_T, D)$ from this and $U_n^+(k+1)$.

Repeating Steps 1-5 as time goes along with $k = 0, 1, 2, \cdots$, estimates of the quadruple matrices for $(A_T, B_T, C_T, D)$ and the system order $n$ are updated recursively.

At this stage it should be emphasized that the proposed algorithm has the following properties: (i) The LQ-factorization is required only once at the beginning $k = 0$ (Step 0); (ii) after that (for $k = 1, 2, \cdots$) matrices $\{\tilde{L}_j(k), j = 0, 1, \cdots, 4\}$ are revised by recursion formulas; and (iii) the dimension of the decomposed matrix required at each time step (Step 3) to perform the eigenvalue decomposition (28) is $i \ell \times i \ell$ and this can be so small as compared with the data matrix size $i \ell \times N$.

The basic idea of the method proposed in this paper is to update and down date a couple of input and output data in order to keep the data size to be constant. Such an idea was originally used by Cho, Xu and Kailath [14] and Lovera, Gustafsson and Verhaegen [6]. The algorithm in [14] needs the updates of projection matrix of output data as well as observation noise covariance, and further requires the decomposition of these matrices into generator vectors and the rearrangement of these vectors, using the fast subspace decomposition developed by them. As a result, the algorithm is so intricate that the whole procedure is complex throughout. On the other hand, the algorithm in [6] is based on the use of subspace tracking for the update of the SVD, and still complicated.

As compared with these algorithms, that proposed in this paper is derived directly from the key equation (8) and works by the direct recursion of L-factors of the LQ-factorization. The derivation is so intuitive and direct that we call it bona fide recursive algorithm.

4. Discussions. (i) Adaptation to Time-varying System Identification. As mentioned in Introduction, it will be needless to say that the major mission of the recursive algorithm is to cope with the time-varying systems. Indeed, the algorithm proposed in Section 3 can cope with such systems under the local stationarity assumption. “Time-varying” means in this paper that all system matrices $(A(t), B(t), C(t), D(t))$ for $\Sigma_C$ or $(A(k), B(k), C(k), D(k))$ for $\Sigma_D$ change slowly with time. Qualitatively speaking, the word “slowly” implies that each matrix changes slowly and continuously, and does not abrupt or randomly. Indeed, consider an interval $I_k$ around the time step $k$, and assume that the system matrix $A(t)$ or $A(k)$ behaves to be approximately time-invariant during this interval, i.e., $A(t) = A(t_k) = A_k$, or $A(k) = A_k$. Under this local stationarity assumption [12], (4) is derived as the input-output algebraic relation in which matrices $\Gamma_i(k)$, $H_i(k)$ and $\Gamma_i(k)$ are given in terms of $A_k$, $B_k$, $C_k$ and $D_k$. 
Different from ours, Verhaegen and Deprettere [4] proposed a method of recursive identification under the quasi-stationarity assumption by introducing forgetting factor to multiply the bygone input and output data. Same idea of introducing forgetting factor is used in several papers (e.g. [5-8, 10, 11, 15]). The scenario of these papers is to weaken the influence of the old input and output data.

(ii) Algorithm Using Cholesky Factorization. The most primitive and simplest approach to derive recursive algorithm will be the use of Cholesky factorization in the data matrix [16]. To fix the idea, write the relation (8) as

\[
\begin{bmatrix}
H_0(k) : h_0(k+1) \\
\end{bmatrix} = \begin{bmatrix}
h_0(k - N + 1) : H_0(k+1) \\
\end{bmatrix}.
\]

Post-multiplying the left-hand side of (29) by its transpose, we have

\[
H_0(k+1)H_0^T(k+1) = H_0(k)H_0^T(k) + h_0(k+1)h_0^T(k+1)
\]

\[- h_0(k - N + 1)h_0^T(k - N + 1). \tag{30}\]

This gives the recursive form for \(H_0(k)H_0^T(k)\). Performing the Cholesky factorization of this matrix-product, we have the relation

\[
H_0(k)H_0^T(k) = L(k)L^T(k), \tag{31}\]

where \(L(k)\) is a lower triangular matrix.

This approach is simple, however the matrix \(H_0(k)H_0^T(k)\) may become rank-deficient whenever the data are free from the observation noise or the noise intensity is so small, so that the Cholesky factorization may fail [17].

(iii) Alternative Algorithm with Unfixed Data Size. The recursive algorithm derived in Section 3 is based on the key relation (8), so that in the algorithm the total amount of input and output data employed in the current identification is kept constant. Instead, alternative algorithm is still available if we allow the increase of the amount of input and output data employed in current identification. Such algorithm may be rather preferable for time-invariant systems because the increase in the amount of data implies inevitably the higher accuracy in the identification results.

Let \(k_0\) be a fixed time step and let the output data matrix \(Y_i(k|k_0)\) be constructed from the observation data obtained up to the current step \(k\) from \(k_0\) as (for \(\Sigma_D\))

\[
Y_i(k|k_0) = [y_i(k_0), \ldots, y_i(k-1), y_i(k)], \tag{32}\]

where \(k = k_0, k_0 + 1, \ldots\). This time, the relation

\[
\begin{bmatrix}
U_i(k+1|k_0) \\
U_h(k+1|k_0) \\
Y_i(k+1|k_0)
\end{bmatrix} = \begin{bmatrix}
U_i(k|k_0) : u_i(k+1) \\
U_h(k|k_0) : u_h(k+1) \\
Y_i(k|k_0) : y_i(k+1)
\end{bmatrix} \tag{33}\]

becomes a clue to derive recursive algorithm. Similar procedure to one proposed in Section 3 is applicable [12]. In this algorithm the number of columns of each data matrix increases according to \(N = k - k_0 + 1 (=: N(k))\) as a function of \(k\).
5. **Simulation Studies.** In order to provide the reader with a feeling of how the *bona fide* recursive algorithm produces good results, a small but representative subset of simulation experiments are given here.

**A. Discrete-time time-varying system**

Consider the second-order discrete-time system (2) with system matrices:

\[
A = \begin{bmatrix} a_{11}(k) & 0.4 \\ -0.4 & a_{22}(k) \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ -1 \end{bmatrix}
\]

\[
C = \begin{bmatrix} 1 & 2 \\ -1 & 2 \end{bmatrix}, \quad D = \begin{bmatrix} 1.5 & 3 \\ 2 & 1 \end{bmatrix},
\]

where

\[
a_{11}(k) = 0.1 + 0.5 \exp \{-0.5k/500\},
\]

\[
a_{22}(k) = 0.5 + 0.6 \exp \{-0.5k/500\}.
\]

The random noises \(w_k\) and \(v_k\) are mutually independent and have common covariance 
\(\mathbb{E}\{w(k)w^T(j)\} = \mathbb{E}\{v(k)v^T(j)\} = 0.1^2 I\delta_{kj}\) (\(I\): unit matrix). The user-defined parameters are set: \(i = h = 10, N = 500\) or \(N = 250\).

The 100 Monte Carlo experiments were conducted. Figures 1 and 2 depict time-evolutions of sample-averaged means and variances of the real and imaginary parts of estimated pole for \(N = 500\), in which estimated and true ones are depicted by solid and dashed curves, respectively. Figure 3 illustrates the feature of eigenvalues estimated by the recursive algorithm applied to a typical set of input and output data. Same simulations were performed for \(N = 250\) as shown in Figs.4-6. In the simulation studies the stationarity interval \(I_k\) mentioned in Section 4 was taken as \(I_k = [k - N + 1, k]\) and the identified values at the center time step are plotted in these figures. From these three figures it can be seen that the proposed *bona fide* algorithm tracks well the true values for both cases of \(N = 500\) and \(N = 250\).

Recursive algorithm developed by Mercère, et al. [11] is one of the most recent algorithms based on the subspace tracking idea. Although this algorithm is derived for time-invariant systems, we select this as a competitor to compare the results obtained above. Their algorithm is summarized briefly as follows.

Let

\[
z_i(k) := y_i(k) - \Phi_i u_i(k) = \Gamma_i x_i(k) + b_i(k),
\]

where \(\Gamma_i\) is the observability matrix; \(\Phi_i\) is the lower triangular Toeplitz matrix corresponding to our \(H_i\); and \(b_i(k)\) is the total noise vector. Divide the vector \(z_i(k)\) into two parts:

\[
z_i(k) = \begin{bmatrix} z_{\alpha_1}(k) \\ z_{\alpha_2}(k) \end{bmatrix} = \begin{bmatrix} z_{\alpha_1}(k) \\ P^T z_{\alpha_1}(k) \end{bmatrix} = \begin{bmatrix} I \\ P^T \end{bmatrix} z_{\alpha_1}(k)
\]

in which \(P\) is the matrix called propagator and the partition is performed according to the system order (which is assumed known in the paper). Introducing an instrumental variable \(\xi(k)\), the estimation of the propagator \(P\) is obtained by minimizing the quadratic criterion given by
\[ J(\hat{P}) = \sum_{j=1}^{k} \lambda^{k-j} \left\| z_{\alpha_2}(j)\xi^T(j) - \hat{P}^T(k)z_{\alpha_1}(j)\xi^T(j) \right\|^2; \] (36)

where $\lambda (0 < \lambda < 1)$ is a forgetting factor. The estimate $\hat{P}(k)$ is revised by the following recursion algorithm:

\[ K(k) = \frac{\xi^T(k)R(k-1)}{\lambda + \xi^T(k)R(k-1)z_{\alpha_1}(k)} \] (37)

\[ \hat{P}^T(k) = \hat{P}^T(k-1) + \left[ z_{\alpha_1}(k) - \hat{P}^T(k-1)z_{\alpha_1}(k) \right]K(k) \] (38)

\[ R(k) = \frac{1}{\lambda} \left[ R(k-1) - R(k-1)z_{\alpha_1}(k)K(k) \right]; \] (39)

where $R(k) = [E\{z_{\alpha_1}(k)\xi^T(k)\}]^{-1}$. For details, refer [11]. Here, it should be noted that the forgetting factor $\lambda$ appears as a reciprocal, so this algorithm may be so sensitive in selecting it.

Eight sets of experiments were performed by changing the forgetting factor for $\lambda = 0.999, 0.998, 0.995, 0.99, 0.98, 0.95, 0.9$ and 0.8 because we encountered difficulty how to determine it. Through these experiments we found that their algorithm is so sensitive in choosing it. Figures 7-10 depict representative results for $\lambda = 0.998$ and $\lambda = 0.95$. From these experiments the authors have found the tendency that in their algorithm, the smaller
the forgetting factor is selected, the finer the sample-averaged mean of the estimated real part of the pole becomes; while both sample-averaged variances of the estimated real and imaginary parts become fluctuating intensively. Recalling that the role of forgetting factor is to weaken the influence of the past data in identification, we see that the comparison of two top figures in Figs.4 and 9 implies that the forgetting factor $\lambda = 0.95$ corresponds
to our data size of $N = 250$. Although these two sample-averaged mean of the real part of the system pole seem equivalent with each other; however their features of sample-averaged variance are quite different. Same thing is said for the estimate of imaginary part of the pole.

Figure 4. Time-varying features of the sample-averaged mean (top) and variance (bottom) of the real part of estimated pole by the proposed algorithm ($N = 250$: fixed)

Through these simulation studies it may fairly be said that our proposed algorithm is stable in estimating the system order and further the errors in estimating both real and imaginary parts of the system pole are smaller as shown by the (sample-averaged) variances than that obtained by recursive algorithm by Mercère, et al. [11].

As for the computation time, their algorithm takes 16.75 s for each trial, while ours 19.25 s; hence the former algorithm is somewhat superior to ours in computation time.

B. Sudden change system

Although the recursive algorithm was derived under the local stationarity assumption for system matrices, let us examine our algorithm to check whether it can track the system’s change well or not. To do this, consider the same second-order system as above, except the components

$$a_{11}(k) = \begin{cases} -0.2 & (0 < k < 700) \\ 0.2 & (k \geq 700) \end{cases}, \quad a_{22}(k) = 0.2.$$ 

Figure 11 shows the result of 100 Monte Carlo experiments where the data size was fixed as $N = 500$. As can be seen, even in this sudden change case the proposed recursive algorithm tracks the system fairly well.
Figure 5. Time-varying features of the sample-averaged mean (top) and variance (bottom) of the imaginary part of estimated pole by the proposed algorithm ($N = 250$: fixed)

Figure 6. Estimated eigenvalues by the proposed algorithm ($N = 250$: fixed)

C. Other cases
Simulations for continuous-time, time-invariant and time-varying systems are in part examined in [18].

6. Conclusions. In this paper, a recursive subspace identification algorithm has been proposed by updating $L$-factors of $LQ$-factorization, keeping the size of input and output
Figure 7. Sample-averaged mean (top) and variance (bottom) of the real part of the pole by Mercère, et al. (2003) ($\lambda = 0.998$)

data matrix constant as one of user-defined parameters. Its derivation is simple and direct. By simulation studies for time-varying systems its performance has been proved by comparing with another method using forgetting factor.

Acknowledgment. This work is partially supported by the Japan Society for the Promotion of Science (JSPS) under Grant-in-Aid for Scientific Research (B)-16360046. The authors are grateful to Professor K. Wada at Kyushu University for his valuable comment made on the recursive algorithm in our earlier version which made it clearer as in the present form.

REFERENCES

Figure 8. Sample-averaged mean (top) and variance (bottom) of the imaginary part of the pole by Mercère, et al. (2003) ($\lambda = 0.998$)


Figure 9. Sample-averaged mean (top) and variance (bottom) of the real part of the pole by Mercère, et al. (2003) ($\lambda = 0.95$)

Figure 10. Sample-averaged mean (top) and variance (bottom) of the imaginary part of the pole by Mercère, et al. (2003) ($\lambda = 0.95$)

Figure 11. Sample-averaged mean (top) and variance (bottom) of the estimated pole ($N = 500$: fixed)