Square-Root Information Filtering and Fixed-Interval Smoothing with Singularities

Mark L. Psiaki[*]

Cornell University, Ithaca, New York 14853-7501

Abstract

The square-root information filter and smoother algorithms have been generalized to handle singular state transition matrices and perfect measurements. This has been done to allow the use of SRIF techniques for problems with delays and state constraints. The generalized algorithms use complete QR factorization to isolate deterministically known parts of the state and nonsingular parts of the state-transition and disturbance-influence matrices. These factorizations and the corresponding changes of coordinates are used to solve the recursive least-squares problems that are basic to the SRIF technique.

1. Introduction

1.1 Expanding the Capabilities of Square-Root Information Filters and Smoothers. Square-Root Information Filters and Smoothers (SRIF&S) are special solution techniques for, respectively, the discrete-time Kalman filter problem and the related smoothing problem [1]. Square root formulations increase numerical computation accuracy by guaranteeing positive-definiteness of the associated covariances and by decreasing the condition numbers of the manipulated matrices. The present paper generalizes the SRIF&S techniques to deal with singularities that can occur in some problems.

Suppose the system dynamics and measurement models are:

$$\boldsymbol{x}_{(j+1)} = \boldsymbol{F}_{(j)}\boldsymbol{x}_{(j)} + \boldsymbol{G}_{(j)}\boldsymbol{w}_{(j)} \tag{1a}$$

$$\boldsymbol{z}_{a(j)} = \boldsymbol{A}_{(j)}\boldsymbol{x}_{(j)} + \boldsymbol{n}_{(j)} \tag{1b}$$

$$\boldsymbol{z}_{b(j)} = \boldsymbol{B}_{(j)}\boldsymbol{x}_{(j)} \tag{1c}$$

where $\boldsymbol{x}_{(j)}$ is the state at stage j, $\boldsymbol{w}_{(j)}$ is the random process disturbance, $\boldsymbol{z}_{a(j)}$ is a noisy measurement, $\boldsymbol{n}_{(j)}$ is its random measurement error, $\boldsymbol{z}_{b(j)}$ is a prefect measurement, and $\boldsymbol{F}_{(j)}$, $\boldsymbol{G}_{(j)}$, $\boldsymbol{A}_{(j)}$, and $\boldsymbol{B}_{(j)}$ are matrices of appropriate dimensions.

The standard SRIF&S algorithms cannot handle a singular state transition matrix, $\boldsymbol{F}_{(j)}$. The filtering algorithm inverts $\boldsymbol{F}_{(j)}$ [1, p. 116] as does one version of the smoothing algorithm [1, p. 215]. The generalized SRIF&S algorithms presented below can handle a singular $\boldsymbol{F}_{(j)}$, such as occurs when the system model includes a pure delay.

The standard SRIF&S algorithms cannot deal with perfect measurements, as given in eq. (1c). This paper's generalized algorithms can handle such measurements. Attitude estimation with quaternions is an example problem in which perfect "measurements" arise.

[*] Associate Professor, Mech. & Aero. Engineering.

Presented as Paper No. FA05-5 at the 1998 American Control Conference, Philadelphia, PA.

1.2 Relationship to Prior Research. The smoothing algorithms of Watanabe [2] and McReynolds [3] are the only factorized fixed-interval smoothers that can handle a singular state transition matrix. Watanabe's algorithm first executes a backwards pass followed by a forwards pass; it cannot do just filtering. McReynolds' algorithm avoids inverting the state transition matrix only in the case of full-rank process noise.

Several known algorithms can deal with perfect measurements [4-6]. None of them use a square-root factorization, nor do any address the smoothing problem.

This paper generalizes the original SRIF&S algorithms in Refs. 1 and 7 to deal with a) a singular state transition matrix and process noise of any rank, b) perfect measurements, and c) perfect knowledge of a component of $\boldsymbol{x}_{(0)}$. The generalized algorithms use forward-pass filtering followed by backwards-pass smoothing.

1.3 Outline of Paper. Section 2 presents the filtering and smoothing problems, Section 3 the forward-pass filtering algorithm, and Section 4 the backwards-pass smoothing algorithm. Section 5 discusses important algorithmic properties, and Section 6 gives the conclusions.

2. Definition of Filter and Smoother Problems

2.1 Problem Model. The dynamic system and measurement models are given in eqs. (1a)-(1c). The statistics of the process noise measurement noise are:

$$\boldsymbol{R}_{ww(j)}\boldsymbol{w}_{(j)} = \boldsymbol{z}_{w(j)} - \boldsymbol{n}_{w(j)} \tag{2a}$$

$$\mathrm{E}\{\boldsymbol{n}_{w(j)}\} = \boldsymbol{0}, \qquad \mathrm{E}\{\boldsymbol{n}_{w(j)}\boldsymbol{n}_{w(k)}^T\} = \boldsymbol{I}\,d_{jk} \tag{2b}$$

$$\mathrm{E}\{\boldsymbol{n}_{(j)}\} = \boldsymbol{0}, \qquad \mathrm{E}\{\boldsymbol{n}_{(j)}\boldsymbol{n}_{(k)}^T\} = \boldsymbol{I}\,d_{jk} \tag{2c}$$

$$\mathrm{E}\{\boldsymbol{n}_{(j)}\boldsymbol{n}_{w(k)}^T\} = \boldsymbol{0} \tag{2d}$$

where $[\boldsymbol{R}_{ww(j)}, \boldsymbol{z}_{w(j)}]$ is the *a priori* information array for $\boldsymbol{w}_{(j)}$, with $\boldsymbol{R}_{ww(j)}$ square and nonsingular. The vector sequences $\boldsymbol{n}_{w(j)}$ and $\boldsymbol{n}_{(j)}$ are Gaussian white-noise processes.

The *a priori* information about the initial state vector $\boldsymbol{x}_{(0)}$ is assumed to take the form:

$$\begin{bmatrix} \boldsymbol{x}_{a(j)} \\ \boldsymbol{x}_{b(j)} \end{bmatrix} = \tilde{\boldsymbol{Q}}_{ab(j)}\boldsymbol{x}_{(j)} \qquad \text{for } j = 0 \tag{3a}$$

$$\boldsymbol{x}_{a(j)} \text{ given and } \tilde{\boldsymbol{R}}_{bb(j)}\boldsymbol{x}_{b(j)} = \tilde{\boldsymbol{z}}_{b(j)} - \tilde{\boldsymbol{n}}_{b(j)} \qquad \text{for } j = 0 \tag{3b}$$

where $\boldsymbol{x}_{a(0)}$ is deterministic and $\boldsymbol{x}_{b(0)}$ is random and defined by the Gaussian random vector $\tilde{\boldsymbol{n}}_{b(0)} = \mathrm{N}(\boldsymbol{0},\boldsymbol{I})$ and the information array $[\tilde{\boldsymbol{R}}_{bb(0)}, \tilde{\boldsymbol{z}}_{b(0)}]$. $\tilde{\boldsymbol{Q}}_{ab(0)}$ is an orthogonal transformation matrix. The matrix $\tilde{\boldsymbol{R}}_{bb(0)}$ must be square and nonsingular.

2.2 Filtering and Smoothing Problem Statements. The filtering problem is to find the best estimate of the state at stage N conditioned on the measurements up to and including stage N: $\hat{x}_{(N)} = E\{x_{(N)} | z_{a(0)}, z_{b(0)}, ..., z_{a(N)}, z_{b(N)}\}$. The fixed-interval smoothing problem is to find the best estimate of the state time history for stages 0 to N conditioned on the measurements for the entire interval: $x^*_{(j)} = E\{x_{(j)} | z_{a(0)}, z_{b(0)}, ..., z_{a(N)}, z_{b(N)}\}$ for $j = 0, 1, 2, ..., N$.

3. Forward-Pass Filter Algorithm

3.1 Overview of Filtering Procedure. The filtering problem can be solved by a least-squares technique that determines the maximum-likelihood solution [1]. This technique finds $\tilde{n}_{b(0)}$ and the $x_{(j)}$, $w_{(j)}$, $n_{(j)}$, and $n_{w(j)}$ sequences that satisfy constraint eqs. (1a)-(1c), (2a), (3a) and (3b) and minimize

$$J = \frac{1}{2}\left\{\tilde{n}^T_{b(0)}\tilde{n}_{b(0)} + \sum_{j=0}^{N} n^T_{(j)} n_{(j)} + \sum_{j=0}^{N-1} n^T_{w(j)} n_{w(j)}\right\} \tag{4}$$

The algorithm works in an recursive manner analogous to the original SRIF algorithm, [1,7]. It starts at stage j with *a priori* state information in the eq. (3a)-(3b) form. The filter first performs a measurement update to combine its *a priori* state data with the measurement equations for that stage, eqs. (1b) and (1c). It treats the perfect measurements as equality constraints that exactly define a component of the state. Otherwise, the measurement update uses standard SRIF least-squares methods [1, pp. 69-76].

The propagation phase of the generalized SRIF computes the *a priori* state information at stage $j+1$ using the *a posteriori* state information at stage j in conjunction with eqs. (1a) and (2a) for stage j. The principal of the new mapping algorithm is the same as that of the original algorithm: the dynamic model is used to eliminate some stage-j variables from the information equations, and then they are QR-factorized to get an information equation for $x_{(j+1)}$ that is independent of all remaining variables. Complete QR factorizations of matrices are needed in the generalized algorithm in order to deal with singularities.

3.2 Review of Complete QR Factorization. The complete QR factorization of any matrix B is as follows:

$$Q_L^T \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} Q_R = B \tag{5}$$

where Q_L and Q_R are orthogonal and R is square, upper-triangular, and nonsingular [8].

3.3 Detailed Description of Measurement Update. The update procedure for stage j starts with *a priori* state data in the form given in eqs. (3a) and (3b). This is true by assumption for stage 0 and will be enforced by the filtering process for all later stages.

First, the update transforms $A_{(j)}$ and $B_{(j)}$ into components that multiply $x_{a(j)}$ and $x_{b(j)}$ in the measurement equations:

$$\begin{bmatrix} A_{a(j)} & A_{b(j)} \\ B_{a(j)} & B_{b(j)} \end{bmatrix} = \begin{bmatrix} A_{(j)} \\ B_{(j)} \end{bmatrix} \tilde{Q}^T_{ab(j)} \tag{6}$$

Next, it right QR factorizes $B_{b(j)}$, and transforms $x_{b(j)}$:

$$\begin{bmatrix} R_{cc(j)} & 0 \end{bmatrix} Q_{cd(j)} = B_{b(j)} \qquad \begin{bmatrix} x_{c(j)} \\ x_{d(j)} \end{bmatrix} = Q_{cd(j)} x_{b(j)} \tag{7}$$

where $Q_{cd(j)}$ is orthogonal and $R_{cc(j)}$ is square and nonsingular. If a nonsingular $R_{cc(j)}$ is unachievable, then the exact measurements are ill-posed and cannot be satisfied.

The vector $x_{c(j)}$ is determined from the exact measurements by using eqs. (1c), (3a), (6), and (7):

$$x_{c(j)} = R^{-1}_{cc(j)}(z_{b(j)} - B_{a(j)} x_{a(j)}) \tag{8}$$

Next, the update transforms $\tilde{R}_{bb(j)}$ and $A_{b(j)}$ into factors that multiply $x_{c(j)}$ and $x_{d(j)}$ in the noisy measurements and in $x_{b(j)}$'s *a priori* information equation:

$$\begin{bmatrix} \tilde{R}_{bc(j)} & \tilde{R}_{bd(j)} \\ A_{c(j)} & A_{d(j)} \end{bmatrix} = \begin{bmatrix} \tilde{R}_{bb(j)} \\ A_{b(j)} \end{bmatrix} Q^T_{cd(j)} \tag{9}$$

Equations (3b) and (1b) then become:

$$\tilde{R}_{bd(j)} x_{d(j)} = \tilde{z}_{d(j)} - \tilde{n}_{b(j)} \tag{10a}$$

$$A_{d(j)} x_{d(j)} = z_{d(j)} - n_{(j)} \tag{10b}$$

where $\tilde{z}_{d(j)} = \tilde{z}_{b(j)} - \tilde{R}_{bc(j)} x_{c(j)}$ and $z_{d(j)} = z_{a(j)} - A_{a(j)} x_{a(j)} - A_{c(j)} x_{c(j)}$.

The update process finishes by combining eqs. (10a) and (10b) into a single *a posteriori* information equation for $x_{d(j)}$ using a left QR factorization as in [1, p. 71]:

$$T_{(j)} \begin{bmatrix} \tilde{R}_{bd(j)} & \tilde{z}_{d(j)} \\ A_{d(j)} & z_{d(j)} \end{bmatrix} = \begin{bmatrix} \hat{R}_{dd(j)} & \hat{z}_{d(j)} \\ 0 & e_{(j)} \end{bmatrix} \tag{11}$$

where $T_{(j)}$ is orthogonal and $\hat{R}_{dd(j)}$ is square, nonsingular, and upper-triangular. $e_{(j)}$ is the residual measurement error, and the *a posteriori* information equation is:

$$\hat{R}_{dd(j)} x_{d(j)} = \hat{z}_{d(j)} - \hat{n}_{d(j)} \tag{12}$$

The state $x_{(j)}$ is composed of deterministic parts, $x_{a(j)}$ and $x_{c(j)}$, and a stochastic part, $x_{d(j)}$:

$$x_{(j)} = \hat{Q}^T_{acd(j)} \begin{bmatrix} x_{a(j)} \\ x_{c(j)} \\ x_{d(j)} \end{bmatrix} \quad \text{where } \hat{Q}_{acd(j)} = \begin{bmatrix} I & 0 \\ 0 & Q_{cd(j)} \end{bmatrix} \tilde{Q}_{ab(j)} \tag{13}$$

The *a posteriori* state estimate and covariance are:

$$\hat{x}_{(j)} = \hat{Q}^T_{acd(j)} \begin{bmatrix} x_{a(j)} \\ x_{c(j)} \\ \hat{x}_{d(j)} \end{bmatrix} \qquad \hat{P}_{(j)} = \hat{Q}^T_{acd(j)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \hat{P}_{dd(j)} \end{bmatrix} \hat{Q}_{acd(j)} \tag{14}$$

where $\hat{x}_{d(j)} = \hat{R}^{-1}_{dd(j)} \hat{z}_{d(j)}$ is derived by taking the expectation of eq. (12). At stage $j = N$ eq. (14) gives the solution to the filtering problem.

3.4 Detailed Description of Mapping with Process Noise. The propagation phase uses the *a posteriori* information about $\boldsymbol{x}_{(j)}$, eqs. (3b), (8), (12) and (13), the *a priori* information about $\boldsymbol{w}_{(j)}$, eq. (2a), and the difference equation, eq. (1a), to compute *a priori* information about $\boldsymbol{x}_{(j+1)}$ in the form of eqs. (3a)-(3b).

First, the mapping process transforms the state transition matrix into the coefficients of $\boldsymbol{x}_{a(j)}$, $\boldsymbol{x}_{c(j)}$, and $\boldsymbol{x}_{d(j)}$ in the state difference equation:

$$\begin{bmatrix}\boldsymbol{F}_{a(j)} & \boldsymbol{F}_{c(j)} & \boldsymbol{F}_{d(j)}\end{bmatrix} = \boldsymbol{F}_{(j)}\hat{\boldsymbol{Q}}^T_{acd(j)} \tag{15}$$

A deterministic, nonhomogeneous component of the difference equation is then computed, $\boldsymbol{p}_{(j)} = \boldsymbol{F}_{a(j)}\,\boldsymbol{x}_{a(j)} + \boldsymbol{F}_{c(j)}\,\boldsymbol{x}_{c(j)}$, and eq. (1a) becomes:

$$\boldsymbol{x}_{(j+1)} = \boldsymbol{F}_{d(j)}\boldsymbol{x}_{d(j)} + \boldsymbol{G}_{(j)}\boldsymbol{w}_{(j)} + \boldsymbol{p}_{(j)} \tag{16}$$

Next, the mapping process completely QR factorizes $\boldsymbol{F}_{d(j)}$ and makes corresponding changes to $\boldsymbol{G}_{(j)}$ and $\boldsymbol{p}_{(j)}$:

$$\boldsymbol{Q}^T_{gh(j+1)}\begin{bmatrix}\boldsymbol{F}_{ge(j)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\boldsymbol{Q}_{ef(j)} = \boldsymbol{F}_{d(j)} \tag{17a}$$

$$\begin{bmatrix}\boldsymbol{G}_{g(j)}\\ \boldsymbol{G}_{h(j)}\end{bmatrix} = \boldsymbol{Q}_{gh(j+1)}\boldsymbol{G}_{(j)} \text{ and } \begin{bmatrix}\boldsymbol{p}_{g(j)}\\ \boldsymbol{p}_{h(j)}\end{bmatrix} = \boldsymbol{Q}_{gh(j+1)}\boldsymbol{p}_{(j)} \tag{17b}$$

This factorization orthogonally transforms $\boldsymbol{x}_{d(j)}$ and $\boldsymbol{x}_{(j+1)}$:

$$\begin{bmatrix}\boldsymbol{x}_{e(j)}\\ \boldsymbol{x}_{f(j)}\end{bmatrix} = \boldsymbol{Q}_{ef(j)}\boldsymbol{x}_{d(j)} \text{ and } \begin{bmatrix}\boldsymbol{x}_{g(j+1)}\\ \boldsymbol{x}_{h(j+1)}\end{bmatrix} = \boldsymbol{Q}_{gh(j+1)}\boldsymbol{x}_{(j+1)} \tag{18}$$

and isolates a square, nonsingular part of the state transition matrix, $\boldsymbol{F}_{ge(j)}$. The dynamic map in eq. (16) becomes:

$$\begin{bmatrix}\boldsymbol{x}_{g(j+1)}\\ \boldsymbol{x}_{h(j+1)}\end{bmatrix} = \begin{bmatrix}\boldsymbol{F}_{ge(j)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{e(j)}\\ \boldsymbol{x}_{f(j)}\end{bmatrix} + \begin{bmatrix}\boldsymbol{G}_{g(j)}\\ \boldsymbol{G}_{h(j)}\end{bmatrix}\boldsymbol{w}_{(j)} + \begin{bmatrix}\boldsymbol{p}_{g(j)}\\ \boldsymbol{p}_{h(j)}\end{bmatrix} \tag{19}$$

Note that $\boldsymbol{x}_{e(j)}$ and $\boldsymbol{x}_{g(j+1)}$ have the same dimension, and $\boldsymbol{x}_{e(j)}$ affects $\boldsymbol{x}_{g(j+1)}$ through $\boldsymbol{F}_{ge(j)}$. $\boldsymbol{x}_{f(j)}$ does not affect the state at stage $j+1$, and $\boldsymbol{x}_{h(j+1)}$ is not affected by the state at stage j.

The next operation is a complete QR factorization of $\boldsymbol{G}_{h(j)}$

$$\boldsymbol{Q}^T_{ia(j+1)}\begin{bmatrix}\boldsymbol{G}_{iwa(j)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\boldsymbol{Q}_{wawb(j)} = \boldsymbol{G}_{h(j)} \tag{20}$$

with corresponding changes to $\boldsymbol{G}_{g(j)}$ and $\boldsymbol{p}_{h(j)}$:

$$\begin{bmatrix}\boldsymbol{G}_{gwa(j)} & \boldsymbol{G}_{gwb(j)}\end{bmatrix} = \boldsymbol{G}_{g(j)}\boldsymbol{Q}^T_{wawb(j)} \tag{21a}$$

$$\begin{bmatrix}\boldsymbol{p}_{i(j)}\\ \boldsymbol{p}_{a(j)}\end{bmatrix} = \boldsymbol{Q}_{ia(j+1)}\boldsymbol{p}_{h(j)} \tag{21b}$$

This factorization transforms $\boldsymbol{w}_{(j)}$ and $\boldsymbol{x}_{h(j+1)}$:

$$\begin{bmatrix}\boldsymbol{w}_{a(j)}\\ \boldsymbol{w}_{b(j)}\end{bmatrix} = \boldsymbol{Q}_{wawb(j)}\boldsymbol{w}_{(j)} \text{ and } \begin{bmatrix}\boldsymbol{x}_{i(j+1)}\\ \boldsymbol{x}_{a(j+1)}\end{bmatrix} = \boldsymbol{Q}_{ia(j+1)}\boldsymbol{x}_{h(j+1)} \tag{22}$$

and the difference equation in eq. (19) becomes:

$$\begin{bmatrix}\boldsymbol{x}_{g(j+1)}\\ \boldsymbol{x}_{i(j+1)}\\ \boldsymbol{x}_{a(j+1)}\end{bmatrix} = \begin{bmatrix}\boldsymbol{F}_{ge(j)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{e(j)}\\ \boldsymbol{x}_{f(j)}\end{bmatrix} + \begin{bmatrix}\boldsymbol{G}_{gwa(j)} & \boldsymbol{G}_{gwb(j)}\\ \boldsymbol{G}_{iwa(j)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_{a(j)}\\ \boldsymbol{w}_{b(j)}\end{bmatrix} + \begin{bmatrix}\boldsymbol{p}_{g(j)}\\ \boldsymbol{p}_{i(j)}\\ \boldsymbol{p}_{a(j)}\end{bmatrix} \tag{23}$$

Note that $\boldsymbol{w}_{a(j)}$ and $\boldsymbol{x}_{i(j+1)}$ have the same dimension, and $\boldsymbol{w}_{a(j)}$ affects $\boldsymbol{x}_{i(j+1)}$ through $\boldsymbol{G}_{iwa(j)}$, which is a square, nonsingular matrix.

The vector $\boldsymbol{x}_{a(j+1)}$ is deterministically known:

$$\boldsymbol{x}_{a(j+1)} = \boldsymbol{p}_{a(j)} \tag{24}$$

which gives part of the *a priori* information at stage $j+1$.

To finish, the propagation algorithm combines eqs. (2a), (12), (18), (22), and (23) to eliminate $\boldsymbol{x}_{d(j)}$, $\boldsymbol{x}_{e(j)}$, $\boldsymbol{x}_{f(j)}$, $\boldsymbol{w}_{(j)}$, $\boldsymbol{w}_{a(j)}$, and $\boldsymbol{w}_{b(j)}$ and deduce an independent information equation for

$$\boldsymbol{x}_{b(j+1)} = \begin{bmatrix}\boldsymbol{x}_{g(j+1)}\\ \boldsymbol{x}_{i(j+1)}\end{bmatrix} \tag{25}$$

The next step is to left QR factorize information equation (12) after using eq. (18) to replace $\boldsymbol{x}_{d(j)}$ by $\boldsymbol{x}_{e(j)}$ and $\boldsymbol{x}_{f(j)}$. In terms of information arrays this yields:

$$\boldsymbol{T}_{ef(j)}\begin{bmatrix}(\hat{\boldsymbol{R}}_{dd(j)}\boldsymbol{Q}^T_{ef(j)}) & \hat{\boldsymbol{z}}_{d(j)}\end{bmatrix} = \begin{bmatrix}\hat{\boldsymbol{R}}_{ee(j)} & \boldsymbol{0} & \hat{\boldsymbol{z}}_{e(j)}\\ \hat{\boldsymbol{R}}_{fe(j)} & \hat{\boldsymbol{R}}_{ff(j)} & \hat{\boldsymbol{z}}_{f(j)}\end{bmatrix} \tag{26}$$

where $\boldsymbol{T}_{ef(j)}$ is an orthogonal matrix, and $\hat{\boldsymbol{R}}_{ee(j)}$ and $\hat{\boldsymbol{R}}_{ff(j)}$ are both square, nonsingular matrices. The corresponding information equations are:

$$\hat{\boldsymbol{R}}_{ee(j)}\,\boldsymbol{x}_{e(j)} = \hat{\boldsymbol{z}}_{e(j)} - \hat{\boldsymbol{n}}_{e(j)} \tag{27a}$$

$$\hat{\boldsymbol{R}}_{fe(j)}\,\boldsymbol{x}_{e(j)} + \hat{\boldsymbol{R}}_{ff(j)}\,\boldsymbol{x}_{f(j)} = \hat{\boldsymbol{z}}_{f(j)} - \hat{\boldsymbol{n}}_{f(j)} \tag{27b}$$

Equation (2a) must be re-expressed in terms of $\boldsymbol{w}_{a(j)}$ and $\boldsymbol{w}_{b(j)}$. Their respective coefficients in the modified equation are $[\boldsymbol{R}_{wwa(j)}, \boldsymbol{R}_{wwb(j)}] = \boldsymbol{R}_{ww(j)}\boldsymbol{Q}_{wawb(j)}{}^T$. The resulting information equation is:

$$\boldsymbol{R}_{wwa(j)}\boldsymbol{w}_{a(j)} + \boldsymbol{R}_{wwb(j)}\boldsymbol{w}_{b(j)} = \boldsymbol{z}_{w(j)} - \boldsymbol{n}_{w(j)} \tag{28}$$

A combined information equation for $\boldsymbol{x}_{g(j+1)}$, $\boldsymbol{x}_{i(j+1)}$, and $\boldsymbol{w}_{b(j)}$ can be constructed from eqs. (27a) and (28). One must first use the first 2 rows in eq. (23) to solve for $\boldsymbol{x}_{e(j)}$ and $\boldsymbol{w}_{a(j)}$ in terms of $\boldsymbol{x}_{g(j+1)}$, $\boldsymbol{x}_{i(j+1)}$, and $\boldsymbol{w}_{b(j)}$:

$$\boldsymbol{x}_{e(j)} = \boldsymbol{F}^{-1}_{ge(j)}\Big\{\boldsymbol{x}_{g(j+1)} - \boldsymbol{G}_{gwa(j)}\boldsymbol{G}^{-1}_{iwa(j)}(\boldsymbol{x}_{i(j+1)} - \boldsymbol{p}_{i(j)}) - \boldsymbol{G}_{gwb(j)}\boldsymbol{w}_{b(j)} - \boldsymbol{p}_{g(j)}\Big\} \tag{29a}$$

$$\boldsymbol{w}_{a(j)} = \boldsymbol{G}^{-1}_{iwa(j)}(\boldsymbol{x}_{i(j+1)} - \boldsymbol{p}_{i(j)}) \tag{29b}$$

Substitution of these results into eqs. (27a) and (28) yields the following information equation:

$$\begin{bmatrix} (\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}) & -(\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\boldsymbol{G}_{gwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}) & -(\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\boldsymbol{G}_{gwb(j)}) \\ \boldsymbol{0} & (\boldsymbol{R}_{wwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}) & \boldsymbol{R}_{wwb(j)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{g(j+1)} \\ \boldsymbol{x}_{i(j+1)} \\ \boldsymbol{w}_{b(j)} \end{bmatrix} =$$

$$\begin{bmatrix} \left\{\hat{\boldsymbol{z}}_{e(j)} + \hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\left(-\boldsymbol{G}_{gwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}\boldsymbol{p}_{i(j)} + \boldsymbol{p}_{g(j)}\right)\right\} \\ \left\{\boldsymbol{z}_{w(j)} + \boldsymbol{R}_{wwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}\boldsymbol{p}_{i(j)}\right\} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{n}}_{e(j)} \\ \boldsymbol{n}_{w(j)} \end{bmatrix} \tag{30}$$

The final propagation step is a left QR factorization of eq. (30) to make the $\boldsymbol{x}_{g(j+1)}$ and $\boldsymbol{x}_{i(j+1)}$ information equations independent of $\boldsymbol{w}_{b(j)}$ as in [1, p. 117]:

$$\tilde{\boldsymbol{T}}_{(j+1)} \begin{bmatrix} (\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}) & -(\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\boldsymbol{G}_{gwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}) & -(\hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\boldsymbol{G}_{gwb(j)}) \\ \boldsymbol{0} & (\boldsymbol{R}_{wwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}) & \boldsymbol{R}_{wwb(j)} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{R}}_{gg(j+1)} & \boldsymbol{0} & \boldsymbol{0} \\ \tilde{\boldsymbol{R}}_{ig(j+1)} & \tilde{\boldsymbol{R}}_{ii(j+1)} & \boldsymbol{0} \\ \tilde{\boldsymbol{R}}_{wbg(j+1)} & \tilde{\boldsymbol{R}}_{wbi(j+1)} & \tilde{\boldsymbol{R}}_{wbwb(j)} \end{bmatrix} \tag{31a}$$

$$\tilde{\boldsymbol{T}}_{(j+1)} \begin{bmatrix} \left\{\hat{\boldsymbol{z}}_{e(j)} + \hat{\boldsymbol{R}}_{ee(j)}\boldsymbol{F}_{ge(j)}^{-1}\left(-\boldsymbol{G}_{gwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}\boldsymbol{p}_{i(j)} + \boldsymbol{p}_{g(j)}\right)\right\} \\ \left\{\boldsymbol{z}_{w(j)} + \boldsymbol{R}_{wwa(j)}\boldsymbol{G}_{iwa(j)}^{-1}\boldsymbol{p}_{i(j)}\right\} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{z}}_{g(j+1)} \\ \tilde{\boldsymbol{z}}_{i(j+1)} \\ \tilde{\boldsymbol{z}}_{wb(j)} \end{bmatrix} \tag{31b}$$

where $\tilde{\boldsymbol{T}}_{(j+1)}$ is an orthogonal matrix and $\tilde{\boldsymbol{R}}_{gg(j+1)}$, $\tilde{\boldsymbol{R}}_{ii(j+1)}$, and $\tilde{\boldsymbol{R}}_{wbwb(j)}$ are all square, nonsingular matrices.

Given the definition of $\boldsymbol{x}_{b(j+1)}$ in eq. (25), the *a priori* information for stage *j+1*, eqs. (3a) and (3b) for *j+1*, can be completed:

$$\tilde{\boldsymbol{R}}_{bb(j+1)} = \begin{bmatrix} \tilde{\boldsymbol{R}}_{gg(j+1)} & \boldsymbol{0} \\ \tilde{\boldsymbol{R}}_{ig(j+1)} & \tilde{\boldsymbol{R}}_{ii(j+1)} \end{bmatrix}, \quad \tilde{\boldsymbol{z}}_{b(j+1)} = \begin{bmatrix} \tilde{\boldsymbol{z}}_{g(j+1)} \\ \tilde{\boldsymbol{z}}_{i(j+1)} \end{bmatrix} \tag{32a}$$

$$\tilde{\boldsymbol{Q}}_{ab(j+1)} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} & \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \\ [\boldsymbol{I} & \boldsymbol{0}] \end{bmatrix} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_{ia(j+1)} \end{bmatrix} \boldsymbol{Q}_{gh(j+1)} \tag{32b}$$

This is the end of the recursion. The data computed in eqs. (24), (32a), and (32b) provide the inputs to eqs. (3a)-(3b) for stage *j+1*. Thus, the algorithm can repeat itself at the next stage.

4. Backward-Pass Fixed-Interval Smoother Algorithm

4.1 Overview of Smoothing Procedure. The smoother uses data from the filtering solution to execute a recursive backwards pass. Each iteration of this recursion starts with the smoothed state estimate at stage *j+1*, $\boldsymbol{x}^*_{(j+1)}$. It uses an information equation and a part of the difference equation to determine the smoothed process disturbance estimate $\boldsymbol{w}^*_{(j)}$. This process noise, $\boldsymbol{x}^*_{(j+1)}$, and the difference equation determine a component of the smoothed state at stage *j*, $\boldsymbol{x}^*_{e(j)}$. $\boldsymbol{x}^*_{e(j)}$ is used in another information equation to determine another component of $\boldsymbol{x}^*_{(j)}$, $\boldsymbol{x}^*_{f(j)}$. Finally, the smoothed and deterministic components are assembled and transformed to compute $\boldsymbol{x}^*_{(j)}$, which completes the recursion.

This algorithm is a generalization of the Dyer-McReynolds covariance smoother [1, pp. 214-217].

4.2 Detailed Backwards Recursion of the Smoothed State with Calculation of the Smoothed Process Disturbance. The smoother begins at stage *j+1 = N*. The smoothed state at this stage is: $\boldsymbol{x}^*_{(N)} = \hat{\boldsymbol{x}}_{(N)}$, from eq. (14).

First, the backwards pass transforms $\boldsymbol{x}^*_{(j+1)}$ to determine the components $\boldsymbol{x}^*_{g(j+1)}$ and $\boldsymbol{x}^*_{i(j+1)}$:

$$\begin{bmatrix} \boldsymbol{x}^*_{a(j+1)} \\ \boldsymbol{x}^*_{g(j+1)} \\ \boldsymbol{x}^*_{i(j+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^*_{a(j+1)} \\ \boldsymbol{x}^*_{b(j+1)} \end{bmatrix} = \tilde{\boldsymbol{Q}}_{ab(j+1)}\boldsymbol{x}^*_{(j+1)} \tag{33}$$

This follows from eqs. (3a) and (25).

The next step determines $\boldsymbol{w}^*_{a(j)}$ from eq. (29b), $\boldsymbol{w}^*_{b(j)}$ from the expectation value of the 3rd row of the information equation associated with eqs. (31a) and (31b):

$$\boldsymbol{w}^*_{b(j)} = \tilde{\boldsymbol{R}}^{-1}_{wbwb(j)}\left[\tilde{\boldsymbol{z}}_{wb(j)} - \tilde{\boldsymbol{R}}_{wbg(j+1)}\boldsymbol{x}^*_{g(j+1)} - \tilde{\boldsymbol{R}}_{wbi(j+1)}\boldsymbol{x}^*_{i(j+1)}\right] \tag{34}$$

and $\boldsymbol{w}^*_{(j)}$ from eq. (22):

$$\boldsymbol{w}^*_{(j)} = \boldsymbol{Q}^T_{wawb(j)} \begin{bmatrix} \boldsymbol{w}^*_{a(j)} \\ \boldsymbol{w}^*_{b(j)} \end{bmatrix} \tag{35}$$

Next, $\boldsymbol{x}^*_{e(j)}$ is computed from eq. (29a), $\boldsymbol{x}^*_{f(j)}$ from the expectation value of eq. (27b):

$$\boldsymbol{x}^*_{f(j)} = \hat{\boldsymbol{R}}^{-1}_{ff(j)}\,[\,\hat{\boldsymbol{z}}_{f(j)} - \hat{\boldsymbol{R}}_{fe(j)}\;\boldsymbol{x}^*_{e(j)}\,] \tag{36}$$

and $\boldsymbol{x}^*_{d(j)}$ from inversion of the transformation in eq. (18):

$$\boldsymbol{x}^*_{d(j)} = \boldsymbol{Q}^T_{ef(j)} \begin{bmatrix} \boldsymbol{x}^*_{e(j)} \\ \boldsymbol{x}^*_{f(j)} \end{bmatrix} \tag{37}$$

The backwards iteration to stage j is completed by assembling state components and making a transformation. The deterministic state components, $\boldsymbol{x}_{a(j)}$ and $\boldsymbol{x}_{c(j)}$, come from eqs. (3b) and (8), respectively. These components together with $\boldsymbol{x}^*_{d(j)}$ are substituted into the transformation in eq. (13) to determine $\boldsymbol{x}^*_{(j)}$. If $j > 0$, then the backwards recursion is repeated, starting at eq. (33) with a decremented value of j.

4.3 Backwards Recursion for the Smoothed State Covariance. A backwards covariance recursion can be developed by using the equations of the backwards state recursion, the definition of covariance, and the expectation operation. The recursion begins at stage $j+1 = N$, where the smoothed state covariance is known because it is the same as the filtered state covariance from eq. (14): $\boldsymbol{P}^*_{(N)} = \hat{\boldsymbol{P}}_{(N)}$. The lengthy details of this procedure, although tedious and algebraically intensive, are straightforward. They have been omitted for the sake of brevity.

5. Algorithm Characteristics

The present algorithms inherit the numerical stability of the original SRIF&S algorithms. Numerical stability means that the effects of small computer round-off errors do not grow excessively during the calculations. The good numerical properties of the original algorithms and of the new algorithms derive from their use of square roots of information matrices, which reduces condition number and ensures positive-semi-definiteness of the associated covariances [1]. The use of dynamically stable covariance and state propagation equations, i.e., sweep methods [9], also enhances numerical stability. Numerical stability of the new algorithms has been experimentally demonstrated by comparison of their solutions with solutions generated by a numerically stable batch algorithm.

The operation counts of the present algorithms and of the original SRIF&S algorithms have been compared for a representative case. Execution time varies linearly with operation count. The comparison assumes that dim($\boldsymbol{x}_{(j)}$) = n, dim($\boldsymbol{w}_{(j)}$) = m, dim($\boldsymbol{z}_{a(j)}$) = p_a, dim($\boldsymbol{z}_{b(j)}$) = p_b, and dim($\boldsymbol{x}_{a(j)}$) = 0, with $(p_a + p_b) \leq n$ and $p_b \leq m$. It also assumes that the standard SRIF processes $(p_a + p_b)$ noisy measurements and no perfect measurements, that it can invert its state transition matrix, and that it uses QR factorization to compute the transition matrix inverse. The operation counts have been totaled only for multiplications and divisions that occur during matrix factorizations and matrix-matrix multiplications. Only the dominant (highest power) terms have been included in these counts.

The computation count comparison yields the following leading terms per stage of filtering:

Standard SRIF:	$2.5n^3 + 3n^2m + 2nm^2 + (2/3)m^3 + n^2p_a + n^2p_b$
Generalized SRIF:	$6.5n^3 + 4n^2m + 3nm^2 + (4/3)m^3 + 3n^2p_a - 2.5n^2p_b + 3np_b^2 - 2np_ap_b + p_ap_b^2 + (5/6)p_b^3 + 2mp_b^2 - 2nmp_b$

There are no additional dominant terms for smoothing unless the smoothed covariance is to be calculated. The new algorithms are slower than the old algorithms by a factor of about 3.

The computation time scales linearly with the number of stages, N. That is much better than the only competing algorithm that can exactly solve the present problem, a batch least-squares algorithm. The computation time for a dense-matrix batch algorithm scales as N^3. The new algorithms have been compared with a batch algorithm for a problem with $N = 100$ stages and $n = 4$ states. Running in MATLAB the new algorithms executed 95 times faster than the batch algorithm, even though the latter has an advantage in MATLAB.

Matrix storage requirements dictate computer RAM size requirements. Storage requirements are significant when smoothing is being done because matrices that get computed during the forward filtering pass must be saved for re-use during the backwards smoothing pass. Using the same dimensions as for the comparison of operation counts, the following is a comparison of matrix storage requirements between the standard SRIF&S and the new algorithms:

Standard SRIF&S:	$2n^2 + nm + m^2$
Generalized SRIF&S:	$4n^2 + 2nm + m^2 - 4np_b - 3mp_b + 3p_b^2$

The new algorithms use roughly twice the storage of the old ones. This is much more efficient than a dense-matrix batch algorithm, which requires order N^2 storage for an N-stage problem, as opposed to the order N storage requirement for the present algorithms.

6. Conclusions

This paper has presented a generalization of the square-root information filter and smoother algorithms. The generalized algorithms have been designed for use on linear time-varying discrete-time problems. The new smoother algorithm is the only known factorized smoother that can handle the general case of a singular state transition matrix while also executing as a forwards filtering pass followed by a backwards smoothing pass. These are also the only known factorized filter and smoother algorithms that can handle perfect measurements. These algorithms will be useful when performing filtering or smoothing for systems with delays or for systems with state constraints or perfect measurements.

Acknowledgments

Ya'akov Oshman and Stephen McReynolds both provided helpful suggestions concerning the literature search for this paper.

References

1. Bierman, G.J., **Factorization Methods for Discrete Sequential Estimation**, Academic Press, (New York, 1977).
2. Watanabe, K., “A New Forward-Pass Fixed-Interval Smoother Using the U-D Information Matrix Factorization,” **Automatica**, Vol. 22, No. 4, 1986, pp. 465-475.
3. McReynolds, S.R., “Covariance Factorization Algorithms for Fixed-Interval Smoothing of Linear Discrete Dynamic Systems,” **IEEE Transactions on Automatic Control**, Vol. 35, No. 10, 1990, pp. 1181-1183.
4. Tse, E. and Athans, M., “Optimal Minimal-Order Observer-Estimators for Discrete Linear Time-Varying Systems,” **IEEE Transactions on Automatic Control**, Vol. 15, No. 4, 1970, pp. 416-426.
5. Anderson, B.D.O., and Moore, J.B., **Optimal Filtering**, Prentice-Hall, (Englewood Cliffs, N.J., 1979). pp. 292-295.
6. Mendel, J.M., **Lessons in Digital Estimation Theory**, Prentice-Hall, (Englewood Cliffs, N.J., 1987), pp. 230-233.
7. Dyer, P. and McReynolds, S., “Extension of Square-Root Filtering to Include Process Noise,” **Journal of Optimization Theory and Applications**, Vol. 3, No. 6, 1969, pp. 444-458.
8. Gill, P.E., Murray, W., and Wright, M.H., **Practical Optimization**, Academic Press, (New York, 1981), pp. 37-40.
9. McReynolds, S.R., “Fixed Interval Smoothing: Revisited,” **Journal of Guidance, Control, and Dynamics**, Vol. 13, No. 5, 1990, pp. 913-921.