ALGORITHM 675
FORTRAN Subroutines for Computing the Square Root Covariance Filter and Square Root Information Filter in Dense or Hessenberg Forms

MICHEL VANBEGIN and PAUL VAN DOOREN
Philips Research Laboratory, Brussels
and
MICHEL VERHAEGEN
NASA Ames Research Center

In this paper, codes are provided for two of the most popular square root filters: the Square Root Covariance Filter and the Square Root Information Filter. We also give efficient implementations for the time invariant case based on so-called condensed forms. All routines make extensive use of BLAS routines.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems*; G.4 [**Mathematical Software**]—*algorithm analysis*

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Condensed forms, Kalman filtering, numerical algorithms

1. INTRODUCTION

Kalman filtering is one of the most frequently used techniques for estimating or filtering a stochastic process $\{x_k\}$ on which linear observations $\{y_k\}$ are performed. The technique supposes that the processes $\{x_k\}$ and $\{y_k\}$ obey the discrete time-varying linear system

$$x_{k+1} = A_k x_k + B_k w_k + D_k u_k \tag{1}$$

and the linear observation process

$$y_k = C_k x_k + v_k \tag{2}$$

Authors' addresses: M. Vanbegin and P. Van Dooren, Philips Research Laboratory, Avenue Van Becelaere 2, Box 8, B-1170, Brussels, Belgium; M. Verhaegen, NASA Ames, Moffett Field, CA 94035.

where x_k, u_k, and y_k are, respectively, the state vector to be estimated (dimension n), the deterministic input vector (dimension r), and the measurement vector (dimension p), where w_k and v_k are the process noise (dimension m) and the measurement noise (dimension p) of the system, and, finally, where A_k, B_k, C_k, and D_k are *known* matrices of appropriate dimensions. The process noise and measurement noise sequences are assumed zero mean and uncorrelated:

$$E\{w_k\} = 0, \qquad E\{v_k\} = 0, \qquad E\{w_k v_j'\} = 0, \tag{3}$$

with covariances

$$E\{w_j w_k'\} = Q_k \delta_{jk}, \qquad E\{v_j v_k'\} = R_k \delta_{jk}, \tag{4}$$

where $E\{\cdot\}$ denotes the mathematical expectation, $'$ denotes the transpose, and Q_k and R_k are *positive semidefinite* matrices. Now let the linear discrete-time system (1 and 2) be given and the system matrices $\{A_k, B_k, C_k, D_k\}$ and covariance matrices $\{Q_k, R_k\}$ be known. The problem then is to compute the *minimum variance estimate* of the stochastic variable x_k, provided y_1 up to y_j have been measured:

$$\hat{x}_{k|j} = \hat{x}_{k|y_1,\ldots,y_j}. \tag{5}$$

When $j = k$, this estimate is called the *filtered estimate*, and for $j = k - 1$ it is referred to as the one-step predicted or, shortly, the *predicted estimate*.

Kalman filtering is a *recursive* method for solving this problem. This is done by computing the variances $P_{k|k}$ and/or $P_{k|k-1}$ and the estimates $\hat{x}_{k|k}$ and/or $\hat{x}_{k|k-1}$ from their previous values for $k = 1, 2, \ldots$. One assumes $P_{0|-1}$ (i.e., the variance of the initial state x_0) and $\hat{x}_{0|-1}$ (i.e., the mean of the initial state x_0) to be given. The solution can be computed by the following recurrence relations (see, e.g., [1] for a derivation):

$$R_{inov,k} = R_k + C_k P_{k|k-1} C_k', \tag{6}$$

$$K_k = A_k P_{k|k-1} C_k' R_{inov,k}^{-1}, \tag{7}$$

$$P_{k|k} = [I - P_{k|k-1} C_k' R_{inov,k}^{-1} C_k] P_{k|k-1}, \tag{8}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} - P_{k|k-1} C_k' R_{inov,k}^{-1} [C_k \hat{x}_{k|k-1} - y_k], \tag{9}$$

$$P_{k+1|k} = A_k P_{k|k} A_k' + B_k Q_k B_k', \tag{10}$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + D_k u_k. \tag{11}$$

The recursion clearly revolves around the covariance matrices $P_{k|k-1}$ and $P_{k|k}$. In the sequel, we drop the term $D_k u_k$ in (11) since it does not really enter the iteration of $P_{k|k}$ and $P_{k|k-1}$. These matrices are positive semidefinite and can thus be factorized into Cholesky factors (often called "square roots" in the filtering literature). These factorizations are the basis of a number of efficient algorithms. Four of them are given below.

2. SQUARE ROOT FILTERING

Here we recall two of the most popular "square root filters" for computing the recursion (6–11). The *Square Root Covariance Filter* (SRCF) propagates the Cholesky factor of $P_{k|k-1}$ using the Cholesky factors of the process and measurement noise covariance matrices:

$$P_{k|k-1} = S_k S_k', \tag{12}$$

$$Q_k = Q_k^L [Q_k^L]', \tag{13}$$

$$R_k = R_k^L [R_k^L]', \tag{14}$$

where the left factors are all chosen to be lower triangular. The computational method is summarized by the following scheme [1] (here $\doteq$ means "equals by definition"):

$$\underbrace{\begin{bmatrix} R_k^L & C_k S_k & 0 \\ 0 & A_k S_k & B_k Q_k^L \end{bmatrix}}_{\text{(prearray)}} U_1 \doteq \underbrace{\begin{bmatrix} R_{innov,k}^L & 0 & 0 \\ G_k & S_{k+1} & 0 \end{bmatrix}}_{\text{(postarray)}}, \tag{15}$$

$$K_k = G_k [R_{innov,k}^L]^{-1}, \tag{16}$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} - K_k (C_k \hat{x}_{k|k-1} - y_k), \tag{17}$$

where U_1 is an orthogonal transformation that (lower) triangularizes the prearray.

The *Square Root Information Filter* (SRIF) [2] propagates the Cholesky factor of $P_{k|k}^{-1}$ using the Cholesky factor of the inverses of the process and measurement noise covariance matrices (assumed to be invertible here):

$$P_{k|k}^{-1} = T_k' T_k, \tag{18}$$

$$Q_k^{-1} = [Q_k^U]' Q_k^U, \tag{19}$$

$$R_k^{-1} = [R_k^U]' R_k^U, \tag{20}$$

where the right factors are all chosen to be upper triangular. One recursion of the SRIF algorithm is given by [1] and [2]:

$$U_2 \underbrace{\begin{bmatrix} Q_k^U & 0 & Q_k^U \bar{w}_k \\ T_k A_k^{-1} B_k & T_k A_k^{-1} & T_k \hat{x}_{k|k} \\ 0 & R_{k+1}^U C_{k+1} & R_{k+1}^U y_{k+1} \end{bmatrix}}_{\text{(prearray)}} \doteq \underbrace{\begin{bmatrix} Q_{innov,k+1}^U & * & * \\ 0 & T_{k+1} & \hat{\xi}_{k+1|k+1} \\ 0 & 0 & e_{k+1} \end{bmatrix}}_{\text{(postarray)}}. \tag{21}$$

Note that the effect of the mean value $\bar{w}_k$ of the process noise was included in the prearray. This was done here because the SRIF algorithm also works when the process noise w_k in (1) is *not* zero mean, whereas this is not the case for the SRCF. A disadvantage of the SRIF approach is the need for inverses in this formulation (see [8] for a discussion and [6] for an alternative formulation). The new filtered state estimate is then computed via

$$\hat{x}_{k+1|k+1} = T_{k+1}^{-1} \hat{\xi}_{k+1|k+1}. \tag{22}$$

3. CONDENSED FORMS

When the system matrices $\{A, B, C\}$ are *time invariant*, considerable savings can be obtained using so-called *condensed forms* [7]. The idea of using these forms comes from standard linear algebra techniques such as those used in the *QR* and *QZ* algorithms. In these algorithms one performs *preliminary unitary transformations* on the given matrices in order to "condense" them or, in other words, create as many zeros as possible. These zeros are exploited in the subsequent iterative scheme in order to substantially reduce their operation count. (Note that condensed forms can also be used in the time-varying case, but then without any resulting savings since they have to be recomputed at each step k.) The following two forms have been shown to be particularly appropriate for the Kalman filter recursion [7, 8]:

—the controller Hessenberg form, where the compound matrix $[UB \mid UAU']$ is upper trapezoidal or where $[UAU' \mid UB]$ is lower trapezoidal;

—the observer Hessenberg form, where the compound matrix $\left[\frac{UAU'}{CU'}\right]$ is upper trapezoidal or where $\left[\frac{CU'}{UAU'}\right]$ is lower trapezoidal.

An appropriate use of these forms in the implementation of the SRCF and SRIF algorithms yields savings up to a factor 7 if $n \gg m, p$ [8]. Two of these forms are illustrated below for $n = 6$, $m = 3$, and $p = 2$. The lower observer Hessenberg form is

$$\left[\begin{array}{c|c} & C_{oh} \\ \hline B_{oh} & A_{oh} \end{array}\right] = \left[\begin{array}{ccc|cccccc} & & & x & 0 & 0 & 0 & 0 & 0 \\ & & & x & x & 0 & 0 & 0 & 0 \\ \hline x & x & x & x & x & x & 0 & 0 & 0 \\ x & x & x & x & x & x & x & 0 & 0 \\ x & x & x & x & x & x & x & x & 0 \\ x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x \end{array}\right], \tag{23}$$

and the upper controller Hessenberg form is

$$\left[\begin{array}{c|c} & C_{ch} \\ \hline B_{ch} & A_{ch} \end{array}\right] = \left[\begin{array}{ccc|cccccc} & & & x & x & x & x & x & x \\ & & & x & x & x & x & x & x \\ \hline x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 & x & x & x & x \end{array}\right]. \tag{24}$$

These two forms are easily obtained using standard linear algebra methods applied to the system model $\{A, B, C\}$. For each case, the matrix U consists of a sequence of Householder reflections [8].

4. TIME INVARIANT SQUARE ROOT FILTERS

For time invariant systems the complexity of the SRCF can significantly be reduced using the lower observer Hessenberg form (23) for the model (namely, $n^3/g + 3n^2p/2 + n^2m$ instead of $7n^3/6 + 5n^2p/2 + n^2m$ per iteration step). The pattern of zeros after a permutation (i.e., a unitary transformation) of the second and third block columns of the prearray (15) looks like (for the same choice of dimensions as above):

$$\left[\begin{array}{cc|ccc|cccccc} \mathrm{x} & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 \\ x & \mathrm{x} & 0 & 0 & 0 & x_2 & x_2 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathrm{x} & x_3 & x_3 & x_3 & x_3 & x_3 & 0 & 0 & 0 \\ 0 & 0 & x & \mathrm{x} & x_4 & x_4 & x_4 & x_4 & x_4 & 0 & 0 \\ 0 & 0 & x & x & \mathrm{x} & x_5 & x_5 & x_5 & x_5 & x_5 & 0 \\ 0 & 0 & x & x & x & \mathrm{x} & x_6 & x_6 & x_6 & x_6 & x_6 \\ 0 & 0 & x & x & x & x & \mathrm{x} & x_7 & x_7 & x_7 & x_7 \\ 0 & 0 & x & x & x & x & x & \mathrm{x} & x_8 & x_8 & x_8 \end{array}\right], \tag{25}$$

and all the elements x_1 up to x_8 can be eliminated using Householder reflections only (using the x's in the corresponding rows as pivots). Similar significant savings with the upper controller Hessenberg form (24) can be obtained for the SRIF by constructing a unitary transformation that condenses the pair $(A^{-1}, A^{-1}B)$ to controller Hessenberg form (namely, $n^3/6 + 3n^2m/2 + n^2p$ instead of $7n^3/6 + 7n^2m/2 + n^2p$ per iteration step). This, of course, requires the additional construction of the matrices A^{-1} and $A^{-1}B$ before condensing them to the form (24), but since this is done only once it is not a real drawback. The numerical stability of the method, on the other hand, can suffer from the inversion of the matrix A needed in this approach. Here again, after a permutation of the second and third block row of the prearray (21), it has the form:

$$\left[\begin{array}{cc|cccccc|c} \mathrm{x} & x & 0 & 0 & 0 & 0 & 0 & 0 & x \\ 0 & \mathrm{x} & 0 & 0 & 0 & 0 & 0 & 0 & x \\ \hline 0 & 0 & \mathrm{x} & x & x & x & x & x & x \\ 0 & 0 & x_3 & \mathrm{x} & x & x & x & x & x \\ 0 & 0 & x_3 & x_4 & \mathrm{x} & x & x & x & x \\ \hline x_1 & x_2 & x_3 & x_4 & x_5 & \mathrm{x} & x & x & x \\ 0 & x_2 & x_3 & x_4 & x_5 & x_6 & \mathrm{x} & x & x \\ 0 & 0 & x_3 & x_4 & x_5 & x_6 & x_7 & \mathrm{x} & x \\ 0 & 0 & 0 & x_4 & x_5 & x_6 & x_7 & x_8 & x \\ 0 & 0 & 0 & 0 & x_5 & x_6 & x_7 & x_8 & x \\ 0 & 0 & 0 & 0 & 0 & x_6 & x_7 & x_8 & x \end{array}\right] \tag{26}$$

This clearly leads to an analogue of the previous method for the SRIF. Both methods are more elaborately discussed in [8].

5. ROUTINE CALLS AND PROGRAMMING DETAILS

SCRF and **SRIF** are both routines meant especially for systems whose parameters $\{A_k, B_k, C_k, Q_k, R_k\}$ are *time varying*. Each routine implements only *one step* of the Kalman filter recursion (6–11).

SCRF only updates the equations (15 and 16) using the model $\{A_k, B_k, C_k, Q_k^L, R_k^L\}$ as well as the square root S_k as input to the prearray, and then returns S_{k+1} and K_k. From the latter, one can compute $\hat{x}_{k|k-1}$ via (17). Since this (rather simple operation) is not always requested, it is not included in the routine. The mode parameter **WITHK** allows the computation of K_k to be suppressed and is set to **.FALSE.** if $R_{inov,k}^L$ is nearly singular (i.e., if its estimated condition number × **TOL** is larger than 1). Finally, the mode parameter **MULTBQ** allows B_k and Q_k^L to be provided separately or as their product. (For example, the latter is chosen when $Q_k^L = I_m$.) Other parameters G_k and $R_{inov,k}^L$ are not returned but may be recovered from the work array **WRK** if needed.

SRIF updates (21) and (22) using the model $\{A_k^{-1}, B_k, C_{k+1}, Q_k^U, R_{k+1}^U\}$ as well as T_k, y_{k+1}, $\hat{x}_{k|k}$, and $\bar{w}_k$ as input to the prearray and returns T_{k+1} and $\hat{x}_{k+1|k+1}$. The mode parameter **WITHX** allows the computation of $\hat{x}_{k+1|k+1}$ to be suppressed and is set to **.FALSE.** if T_{k+1} is nearly singular (i.e., if its estimated condition number × **TOL** is larger than 1). The mode parameters **MULTAB** and **MULTRC** allow the user to provide A_k^{-1} and B_k separately and R_{k+1}^U and C_{k+1} separately or their respective products. (For example, the latter is chosen when $R_{k+1}^U = I_p$.) Other parameters e_{k+1} and $Q_{inov,k}^U$ are not returned but may be recovered from the work array **WRK** if needed.

The calling sequence for **SRCF** is **SRCF(S, LDS, A, LDA, B, LDB, Q, LDQ, C, LDC, R, LDR, N, M, P, K, LDK, WRK, LDW, MULTBQ, WITHK, TOL)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***S**	A two-dimensional real array containing the lower triangular square root of the **N** × **N** state covariance matrix $P_{k|k-1}$. Upon return it contains the corresponding square root of $P_{k+1|k}$.
LDS	The first (integer) dimension of **S** as declared in the calling program.
A	A two-dimensional real array containing the **N** × **N** state transition matrix A_k of the discrete-time system.
LDA	The first (integer) dimension of **A** as declared in the calling program.
B	A two-dimensional real array containing the **N** × **M** input weight matrix B_k (or its product with **Q** if **MULTBQ** = **.TRUE.**).
LDB	The first (integer) dimension of **B** as declared in the calling program.
Q	A two-dimensional real array containing the **M** × **M** lower triangular Cholesky factor Q_k^L of the process noise covariance matrix.
LDQ	The first (integer) dimension of **Q** as declared in the calling program.

C	A two-dimensional real array containing the **P** × **N** output weight matrix C_k of the discrete-time system.
LDC	The first (integer) dimension of **C** as declared in the calling program.
R	A two-dimensional real array containing the **P** × **P** lower triangular Cholesky factor R_k^L of the measurement noise covariance matrix.
LDR	The first (integer) dimension of **R** as declared in the calling program.
N	Dimension (integer) of the state (actual value used in **SRCF**).
M	Dimension (integer) of the input (actual value used in **SRCF**).
P	Dimension (integer) of the output (actual value used in **SRCF**).
***K**	A two-dimensional real array containing the **N** × **P** Kalman gain K_k upon return (if **WITHK** = **.TRUE.**).
LDK	The first (integer) dimension of **K** as declared in the calling program.
***WRK**	A real working array of dimensions **LDW** × **NW** containing the postarray upon return. **NW** must be at least (**M** + **N** + **P**).
LDW	The first (integer) dimension of **WRK**, which must be at least (**N** + **P**).
MULTBQ	Logical variable indicating how input matrices **B** and **Q** are transferred. If **MULTBQ** = **.TRUE.**, then the product **B***Q is transferred via **B**, and **Q** is not used. If **MULTBQ** = **.FALSE.**, then **B** and **Q** are transferred via the parameters **B** and **Q**, respectively.
***WITHK**	Logical variable indicating whether or not **K** is requested. If a nearly singular matrix is encountered during the calculation of **K**, **WITHK** is set to **.FALSE.** upon return.
TOL	Real indicating the tolerance for the reciprocal condition number of the computed $R_{inov,k}^L$ when solving for K_k.

The calling sequence for **SRIF** is **SRIF(T, LDT, AINV, LDA, B, LDB, RINV, LDR, C, LDC, QINV, LDQ, X, RINVY, W, N, M, P, WRK, LDW, MULTAB, MULTRC, WITHX, TOL)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***T**	A two-dimensional real array containing the **N** × **N** upper triangular square root T_k of the inverse of the state covariance matrix $P_{k\mid k}$. Upon return, it contains the corresponding factor of $P_{k+1\mid k+1}$.
LDT	The first (integer) dimension of **T** as declared in the calling program.
AINV	A two-dimensional real array containing the inverse A_k^{-1} of the **N** × **N** state transition matrix of the discrete-time system (if **MULTAB** = **.FALSE.**).

LDA	The first (integer) dimension of **AINV** as declared in the calling program.
B	A two-dimensional real array containing the **N** × **M** input weight matrix B_k (or its product with **AINV** if **MULTAB** = **.TRUE.**) of the discrete-time system.
LDB	The first (integer) dimension of **B** as declared in the calling program.
RINV	A two-dimensional real array containing the **P** × **P** upper triangular Cholesky factor R_{k+1}^{U} of the inverse of the measurement noise covariance (if **MULTRC** = **.FALSE.**).
LDR	The first (integer) dimension of **RINV** as declared in the calling program.
C	A two-dimensional real array containing the **P** × **N** output weight matrix C_{k+1} (or its product with **RINV** if **MULTRC** = **.TRUE.**).
LDC	The first (integer) dimension of **C** as declared in the calling program.
QINV	A two-dimensional real array containing the **M** × **M** upper triangular Cholesky factor Q_k^U of the inverse of the process noise covariance.
LDQ	The first (integer) dimension of **Q** as declared in the calling program.
***X**	The real vector containing the estimated state $\hat{x}_{k\mid k}$ upon input and containing the estimated state $\hat{x}_{k+1\mid k+1}$ upon return (if **WITHX** = **.TRUE.**).
RINVY	The real product of **RINV** with the measured output y_{k+1}.
W	The real mean value of the state process noise $\bar{w}_k$.
N	Dimension (integer) of the state (actual value used in **SRIF**).
M	Dimension (integer) of the input (actual value used in **SRIF**).
P	Dimension (integer) of the output (actual value used in **SRIF**).
***WRK**	A real working array of dimensions **LDW** × **NW** containing the postarray upon return. **NW** must be at least (**M** + **N** + 1).
LDW	The first (integer) dimension of **WRK**, which must be at least (**M** + **N** + **P**).
MULTAB	Logical variable indicating how input matrices **AINV** and **B** are transferred. If **MULTAB** = **.TRUE.**, then the product **AINV***B is transferred via **B**, and **AINV** is not used. If **MULTAB** = **.FALSE.**, then **AINV** and **B** are transferred via their respective parameters.
MULTRC	Logical variable indicating how input matrices **RINV** and **C** are transferred. If **MULTRC** = **.TRUE.**, then the product **RINV***C is transferred via **C**, and **RINV** is not used. If **MULTRC** = **.FALSE.**, then **RINV** and **C** are transferred via their respective parameters.

***WITHX** Logical variable indicating whether or not **X** is requested. If a nearly singular matrix is encountered during the calculation of **X**, **WITHX** is set to **.FALSE.** upon return.

TOL Real indicating the tolerance for the reciprocal condition number of the computed T_{k+1} when solving for $\hat{x}_{k+1|k+1}$.

The routines **OBHESS** and **COHESS** construct an orthogonal state-space transformation U to reduce an arbitrary time-invariant state-space model $\{A, B, C\}$ to observer or controller Hessenberg form, respectively. Both routines have a logical mode parameter **UPPER** that allows choice between upper and lower Hessenberg forms. Both routines provide the orthogonal transformation U if requested (using the logical mode parameter **WITHU**). Each routine uses only two matrices of the state-space model (A and C for **OBHESS** and A and B for **COHESS**) since these completely determine U. The application of the transformation U to the third matrix is not implemented since this is not always requested. (Even when requested, no savings can be obtained by implementing it within the routines.)

SRCFOB and **SRIFCO** are routines implementing respectively the SRCF and SRIF algorithms for time-invariant systems $\{A, B, C\}$ with time-invariant noise covariances Q and R. They both exploit the time invariance by using the condensed forms that can be obtained by **OBHESS** (with **UPPER** = **.TRUE.**) and **COHESS** (with **UPPER** = **.FALSE.**), respectively. The assumption that the system model is in condensed form is the main difference between **SRCF** and **SRIF**, on the one hand, and **SRCFOB** and **SRIFCO**, on the other. The parameter lists of **SRCF** and **SRCFOB** are indeed identical, while those of **SRIF** and **SRIFCO** only differ in two parameters (namely, **B** and **MULTAB**). The option in **SRIF** to pass **AINV** and **B** separately (i.e., the logical **MULTAB**) is no longer appropriate in **SRIFCO**, since for time-invariant systems this product should be computed only once for all time-steps k. In **SRIFCO**, this product is passed via **AINVB** (replacing **B**), and the parameter **MULTAB** disappears.

Remark. Although **SRCFOB** and **SRIFCO** are meant for time-invariant systems $\{A, B, C\}$ with time-invariant noise covariances Q and R, they can also be applied to models where (some of) these matrices are *time varying*. The routine **SRCFOB** can already be applied using an observer Hessenberg form when only A and C are time invariant. The time dependency of the other matrices, of course, induce some extra work at each step k (such as computing UB_k), but this is largely compensated by the savings obtained in the reduction to the postarray via condensed forms. Similar comments also hold for the routine **SRIFCO** where, in principle, only A and B are required to be time invariant. We remark that this is the reason why the logical parameters **MULTBQ** and **MULTRC** were maintained in these routines: the matrices corresponding to these mode parameters are not necessarily time invariant, and one may wish to perform their products in the routines. Finally, we note that even for systems where all matrices are time varying, one may still use condensed forms if the user can supply an appropriate state-space coordinate system where the relevant matrices have the correct pattern of zeros (see [8] for additional comments).

The calling sequence for **OBHESS** is **OBHESS(A, LDA, N, C, LDC, P, U, LDU, WITHU, UPPER)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***A**	A two-dimensional real array containing the **N** × **N** state transition matrix A to be transformed.
LDA	The first (integer) dimension of **A** as declared in the calling program.
N	Dimension (integer) of the state (actual value used in **OBHESS**).
***C**	A two-dimensional real array containing the **P** × **N** input matrix C to be transformed.
LDC	The first (integer) dimension of **C** as declared in the calling program.
P	Dimension (integer) of the output (actual value used in **OBHESS**).
***U**	A two-dimensional real array which upon return is multiplied (if **WITHU** is **.TRUE.**) with the **N** × **N** state space transformation U reducing the given pair to observer Hessenberg form.
LDU	The first (integer) dimension of **U** as declared in the calling program.
WITHU	Logical variable equal to **.TRUE.** when **U** is requested and **.FALSE.** otherwise.
UPPER	Logical variable equal to **.TRUE.** when upper Hessenberg is requested and **.FALSE.** when lower Hessenberg is requested.

The calling sequence for **COHESS** is **COHESS(A, LDA, N, B, LDB, M, U, LDU, WITHU, UPPER)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***A**	A two-dimensional real array containing the **N** × **N** state transition matrix A to be transformed.
LDA	The first (integer) dimension of **A** as declared in the calling program.
N	Dimension (integer) of the state (actual value used in **COHESS**).
***B**	A two-dimensional real array containing the **N** × **M** input matrix B to be transformed.
LDB	The first (integer) dimension of **B** as declared in the calling program.
M	Dimension (integer) of the input (actual value used in **COHESS**).
***U**	A two-dimensional real array which, upon return, is multiplied (if **WITHU** is **.TRUE.**) with the **N** × **N** state space transformation U reducing the given pair to controller Hessenberg form.
LDU	The first (integer) dimension of **U** as declared in the calling program.
WITHU	Logical variable equal to **.TRUE.** when **U** is requested and **.FALSE.** otherwise.
UPPER	Logical variable equal to **.TRUE.** when upper Hessenberg is requested and **.FALSE.** when lower Hessenberg is requested.

The calling sequence for **SRCFOB** is **SRCFOB(S, LDS, A, LDA, B, LDB, Q, LDQ, C, LDC, R, LDR, N, M, P, K, LDK, WRK, LDW, MULTBQ, WITHK, TOL)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***S**	A two-dimensional real array containing the lower triangular square root of the **N** × **N** state covariance matrix $P_{k\mid k-1}$. Upon return it contains the corresponding square root of $P_{k+1\mid k}$.
LDS	The first (integer) dimension of **S** as declared in the calling program.
A	A two-dimensional real array containing the **N** × **N** state transition matrix A of the discrete-time system.
LDA	The first (integer) dimension of **A** as declared in the calling program.
B	A two-dimensional real array containing the **N** × **M** input weight matrix B (or its product with **Q** if **MULTBQ** = **.TRUE.**).
LDB	The first (integer) dimension of **B** as declared in the calling program.
Q	A two-dimensional real array containing the **M** × **M** lower triangular Cholesky factor Q^L of the process noise covariance matrix.
LDQ	The first (integer) dimension of **Q** as declared in the calling program.
C	A two-dimensional real array containing the **P** × **N** output weight matrix C of the discrete-time system.
LDC	The first (integer) dimension of **C** as declared in the calling program.
R	A two-dimensional real array containing the **P** × **P** lower triangular Cholesky factor R^L of the measurement noise covariance matrix.
LDR	The first (integer) dimension of **R** as declared in the calling program.
N	Dimension (integer) of the state (actual value used in **SRCFOB**).
M	Dimension (integer) of the input (actual value used in **SRCFOB**).
P	Dimension (integer) of the output (actual value used in **SRCFOB**).
***K**	A two-dimensional real array containing the **N** × **P** Kalman gain K_k upon return (if **WITHK** = **.TRUE.**).
LDK	The first (integer) dimension of **K** as declared in the calling program.
***WRK**	A real working array of dimensions **LDW** × **NW** containing the postarray upon return. **NW** must be at least (**M** + **N** + **P**).
LDW	The first (integer) dimension of **WRK**, which must be at least (**N** + **P**).

MULTBQ Logical variable indicating how input matrices **B** and **Q** are transferred. If **MULTBQ = .TRUE.**, then the product **B*Q** is transferred via **B**, and **Q** is not used. If **MULTBQ = .FALSE.**, then **B** and **Q** are transferred via the parameters **B** and **Q**, respectively.

***WITHK** Logical variable indicating whether or not **K** is requested. If a nearly singular matrix is encountered during the calculation of **K**, **WITHK** is set to **.FALSE.** upon return.

TOL Real indicating the tolerance for the reciprocal condition number of the computed $R^{L}_{inov,k}$, when solving for K_k.

The calling sequence for **SRIFCO** is **SRIFCO(T, LDT, AINV, LDA, AINVB, LDB, RINV, LDR, C, LDC, QINV, LDQ, X, RINVY, W, N, M, P, WRK, LDW, MULTRC, WITHX, TOL)** with parameters as follows (those preceded by an asterisk are altered by the routine):

***T** A two-dimensional real array containing the **N** × **N** upper triangular square root T_k of the inverse of the state covariance matrix $P_{k|k}$. Upon return it contains the corresponding factor of $P_{k+1|k+1}$.

LDT The first (integer) dimension of **T** as declared in the calling program.

AINV A two-dimensional real array containing the inverse A^{-1} of the **N** × **N** state transition matrix of the discrete-time system. The matrix **AINV**, together with **AINVB**, is in upper controller Hessenberg form.

LDA The first (integer) dimension of **AINV** as declared in the calling program.

AINVB A two-dimensional real array containing the **N** × **M** product of **AINV.** with the input weight matrix B of the discrete-time system. The matrix **AINVB**, together with **AINV**, is in upper controller Hessenberg form.

LDB The first (integer) dimension of **B** as declared in the calling program.

RINV A two-dimensional real array containing the **P** × **P** upper triangular Cholesky factor R^{U} of the inverse of the measurement noise covariance (if **MULTRC = .FALSE.**).

LDR The first (integer) dimension of **RINV** as declared in the calling program.

C A two-dimensional real array containing the **P** × **N** output weight matrix C (or its product with **RINV** if **MULTRC = .TRUE.**).

LDC The first (integer) dimension of **C** as declared in the calling program.

QINV A two-dimensional real array containing the **M** × **M** upper triangular Cholesky factor Q^{U} of the inverse of the process noise covariance.

LDQ	The first (integer) dimension of **Q** as declared in the calling program.
***X**	The real vector containing the estimated state $\hat{x}_{k\mid k}$ upon input and containing the estimated state $\hat{x}_{k+1\mid k+1}$ upon return (if **WITHX = .TRUE.**).
RINVY	The real product of **RINV** with the measured output y_{k+1}.
W	The real main value of the state process noise $\bar{w}_k$.
N	Dimension (integer) of the state (actual value used in **SRIFCO**).
M	Dimension (integer) of the input (actual value used in **SRIFCO**).
P	Dimension (integer) of the output (actual value used in **SRIFCO**).
***WRK**	A real working array of dimensions **LDW** × **NW** containing the postarray upon return. **NW** must be at least (**M** + **N** + 1).
LDW	The first (integer) dimension of **WRK**, which must be at least (**M** + **N** + **P**).
MULTRC	Logical variable indicating how input matrices **RINV** and **C** are transferred. If **MULTRC = .TRUE.**, then the product **RINV*C** is transferred via **C**, and **RINV** is not used. If **MULTRC = .FALSE.**, then **RINV** and **C** are transferred via their respective parameters.
***WITHX**	Logical variable indicating whether or not **X** is requested. If a nearly singular matrix is encountered during the calculation of **X**, **WITHX** is set to **.FALSE.** upon return.
TOL	Real indicating the tolerance for the reciprocal condition number of the computed T_{k+1} when solving for $\hat{x}_{k+1\mid k+1}$.

We have made extensive use of LINPACK [3], BLAS [5], and XBLAS [4] routines in order to make the code of the routines **SRCF**, **SRIF**, **OBHESS**, **COHESS**, **SRCFOB**, and **SRIFCO** more readable and also enhance performance for computers with machine-code versions of these basic routines. Some additional "BLAS-like" Numerical Algorithms Group routines (for implementing Householder transformations) have been used for the same reason.

ACKNOWLEDGMENTS

We gratefully acknowledge the help of Aloys Geurts (Technical University of Eindhoven) when testing and documenting a preliminary version of this routine and Sven Hammarling (Numerical Algorithms Group) for providing us with the BLAS-like routines F06FBF, F06FSF, and F06FUF. We also thank an anonymous reviewer for numerous details improving the paper.

REFERENCES

1. ANDERSON, B. D. O., AND MOORE, J. B. *Optimal Filtering.* Prentice-Hall, Englewood Cliffs, N.J., 1979.
2. BIERMAN, G. J. *Factorization Methods for Discrete Sequential Estimation.* Academic Press, Orlando, Fla., 1977.

3. DONGARRA, J. J., BUNCH, J. R., MOLER, C. B., AND STEWART, G. W. *LINPACK Users's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1979.
4. DONGARRA, J. J., DU CROZ, J., HAMMARLING, S., AND HANSON, R. J. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Softw. 14*, 1 (Mar. 1988), 1–17.
5. LAWSON, C. L., HANSON, R. J., KINCAID, D. R., AND KROGH, F. T. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 324–325.
6. PAIGE, C. C. Covariance matrix representation in linear filtering. In the *Special Issue of Contemporary Mathematics in Linear Algebra and its Role in Systems Theory*. American Mathematical Society, Providence, R.I., 1985.
7. VAN DOOREN, P., AND VERHAEGEN, M. On the use of unitary state-space transformations. In the *Special Issue of Contemporary Mathematics in Linear Algebra and its Role in Systems Theory*. American Mathematical Society, Providence, R.I., 1985.
8. VAN DOOREN, P., AND VERHAEGEN, M. Condensed forms for efficient time invariant Kalman filtering. *SIAM J. Sci. Stat. Comput. 9*, 3 (May 1988), 516–530.

Received July 1987; revised September 1988; accepted October 1988.