SIAM J. SCI. STAT. COMPUT.
Vol. 9, No. 3, May 1988

008

CONDENSED FORMS FOR EFFICIENT TIME-INVARIANT KALMAN FILTERING*

PAUL VAN DOOREN† AND MICHEL VERHAEGEN‡

Abstract. In this paper, new numerical implementations are developed for several "classical" types of Kalman filters. These implementations are based on the choice of an initial state transformation which "condenses" the original model and are therefore mainly meant for time-invariant systems. Since unitary transformations are used to generate these condensed forms, no loss of accuracy is thereby incurred. The use of these forms may lead to a complexity reduction of a factor of 7 in the subsequent recursions to compute the stationary solution of the discrete Riccati equation. It is shown that therefore these new implementations become competitive with the so-called "fast Chandrasekhar" implementation.

Key words. Kalman filter, square root filter, Chandrasekhar filter, condensed forms

AMS(MOS) subject classifications. 65F30, 65U05, 93E11, 93E25

1. Introduction. Since the appearance of Kalman's 1960 paper [8], the so-called Kalman Filter (KF) has been applied successfully to many practical problems, especially in aeronautical and aerospace applications. As applications became more numerous, some pitfalls of the KF were discovered such as the problem of divergence due to the lack of reliability of the numerical algorithm [6], [3]. Later on, more reliable KF implementations were described such as the square root filters (SRF) proposed by Potter and Stern in 1963 [13]. For these filters the reliability of the filter estimates is expected to be better because of the use of numerically stable orthogonal transformations for each recursion step [12]. In terms of number of operations, these implementations can be made as efficient as the conventional KF, or for the Chandrasekhar SRF even more efficient for some special initial conditions.

In this paper the issue of complexity is addressed for the following filter types: the Conventional Kalman Filter (CKF), the Square Root Covariance Filter (SRCF), the Chandrasekhar Square Root Filter (CSRF), and the Square Root Information Filter SRIF). New variants of the CKF, the SRCF, and the SRIF are presented that are mainly restricted to the time-invariant case. These implementations are based on an appropriate choice of coordinate system (for the state space) which "condenses" the model. Three types of such forms are given: the Schur form, the observer-Hessenberg form, and the controller-Hessenberg form. These "condensed forms" are shown to yield considerable savings in computations during the subsequent filter recursion. Moreover, since unitary transformations can be used to obtain the initial condensed form, *no loss of accuracy* is induced by the original coordinate transformation.

In a connected paper [16], the numerical properties of the different filters are analyzed using a detailed error analysis and in [17] these methods are also tested on the realistic problem of flight path reconstruction. There it is shown that the Chandrasekhar SRF is in fact unstable, while the others are stable in a certain sense, which also holds for the new condensed versions presented here in this paper. This makes the condensed SRCF and SRIF algorithms particularly appealing since they are then the fastest reliable KF implementations presently available.

* Received by the editors January 22, 1986; accepted for publication (in revised form) May 12, 1987.
† Philips Research Laboratory, B-1170 Brussels, Belgium.
‡ National Aeronautics and Space Administration-Ames Research Center, Moffett Field, California 94035.

2. Notation and preliminaries. In this section we introduce our notation and list the different Kalman filter types that are discussed in the paper. We consider 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,

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

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 (with nonsingularity of A_k required for the SRIF). 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 definite* matrices. Now let the linear discrete-time system (1), (2) be given and the system matrices $\{A_k, B_k, C_k, D_k\}$ and covariance matrices $\{Q_k, R_k\}$ be known, then the problem 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.* The above problem is restricted here to these two types of estimates except for a few comments in the concluding remarks.

Kalman filtering is a *recursive* method to solve 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, this for $k = 1, 2, \cdots$. Thereby 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 recursive solution can be computed by the Conventional Kalman Filter (CKF) equations, summarized in the following "covariance form" [1]:

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

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

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

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

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

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

These equations have been implemented in various forms (see, e.g., [1]). An efficient implementation that exploits the symmetry of the different matrices in (6)-(11) yields the following operation count for each recursion step:

$$\frac{3}{2}n^3+n^2\left(3p+\frac{m}{2}\right)+n\left(\frac{3p^2}{2}+m^2\right)+\frac{p^3}{6}\text{ "flops,"} \tag{12}$$

where 1 flop = 1 multiplication + 1 addition. The details of the operation count (exploiting symmetry) are given in Table 1. By exploiting the symmetry of the matrices in (6)-(11) one in fact saves $(n^3/2+n^2m/2+np^2/2)$ flops over a straightforward implementation that one would obtain when, e.g., implementing (6)-(11) on a digital computer by using *standard* matrix multiplication routines.

TABLE 1
Efficient implementation of the CKF.

	Mathematical expression	Number of flops
CP	$C_k \cdot P_{k\|k-1}$	n^2p
$R_{e,k}$	$CP \cdot [C_k]' + R_k$	$np^2/2$
K_k	$A_k \cdot CP' \cdot R_{e,k}^{-1}$	$p^3/6+np^2+n^2p$
$P_{k+1\|k}$	$(A_k \cdot P_{k\|k-1} - K_k \cdot CP) \cdot A_k' + B_k \cdot Q_k \cdot B_k'$	$1/2n^2(3n+2p+m)+nm^2$
$\hat{x}_{k+1\|k}$	$A_k \cdot \hat{x}_{k\|k-1} - K_k \cdot (C_k \cdot \hat{x}_{k\|k-1} - y_k) + D_k \cdot u_k$	$n^2+2np+nr$

When the system matrices are *time invariant*, i.e., $\{A, B, C, D\}$, more considerable savings can be obtained using so-called *condensed forms*. 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 (for the QR- and QZ-algorithms this is the preliminary reduction to Hessenberg form). These zeros are then exploited in the subsequent iterative scheme in order to reduce substantially their operation count.

In this paper, we use a similar technique to reduce the operation count of various implementations of the Kalman filter recursion. This reduction is obtained either during the multiplication of a condensed matrix with a full one or during the construction of a (QR-) decomposition of a condensed matrix. In both cases one obtains savings that are comparable to the percentage of zeros in the condensed form. This is of course obtained at *each stage* of the Kalman filter recursion, while the construction of the condensed form is done only *once*. How much these savings exactly are and how they are actually obtained depends on both the choice of condensed form and on the choice of implementation of the Kalman filter recursion. In the next section we give a number of possible condensed forms of state space models. In § 4, we then give various possible KF implementation and in § 5 we show how savings can be obtained by using the different condensed state space models. We also give some actual computer timings in order to illustrate the predicted savings based on operations counts.

3. Condensed state space models. If the system model (1)-(2), defined by the triplet $\{A, B, C\}$, is *time invariant* one often prefers to transform it (via *similarity transformations*) into a special system structure (canonical forms, etc.) in order to gain insight into different system properties (e.g., asymptotic stability) and to reduce the number of parameters necessary to describe the system dynamics. From a numerical point of view such similarity transformations may not be very attractive since the accuracy of the system parameters may *decrease* when the transformations are ill-conditioned. When

restricting oneself to unitary state-space transformations to define a new state vector, $x_t = U \cdot x$, one *generally* (see also § 6) does not alter the sensitivity of the problem [15] and does not modify the relative precision of the system model $\{UAU^*, UB, CU^*\}$, where the * superscript denotes the conjugate transpose. One can now always choose U such that the new system model $\{UAU^*, UB, CU^*\}$ is in one of the following forms:

- The Schur form, where UAU^* is in upper Schur form;
- The observer-Hessenberg form, where the compound matrix

$$\left[\begin{array}{c} UAU^* \\ \hline CU^* \end{array}\right]$$

is upper trapezoidal;

- The controller-Hessenberg form, where the compound matrix $[UB|UAU^*]$ is upper trapezoidal.

These forms are illustrated below for $n = 6$, $m = 3$, and $p = 2$:

$$\left[\begin{array}{c|c} B_s & A_s \\ \hline & C_s \end{array}\right] = \left[\begin{array}{ccc|cccccc} x & x & x & x & x & x & x & x & x \\ x & x & x & 0 & x & x & x & x & x \\ x & x & x & 0 & 0 & x & x & x & x \\ x & x & x & 0 & 0 & 0 & x & x & x \\ x & x & x & 0 & 0 & 0 & 0 & x & x \\ x & x & x & 0 & 0 & 0 & 0 & 0 & x \\ \hline & & & x & x & x & x & x & x \\ & & & x & x & x & x & x & x \end{array}\right], \tag{13}$$

$$\left[\begin{array}{c|c} B_{oh} & A_{oh} \\ \hline & C_{oh} \end{array}\right] = \left[\begin{array}{ccc|cccccc} x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x \\ x & x & x & 0 & x & x & x & x & x \\ x & x & x & 0 & 0 & x & x & x & x \\ x & x & x & 0 & 0 & 0 & x & x & x \\ \hline & & & 0 & 0 & 0 & 0 & x & x \\ & & & 0 & 0 & 0 & 0 & 0 & x \end{array}\right], \tag{14}$$

$$\left[\begin{array}{c|c} B_{ch} & A_{ch} \\ \hline & C_{ch} \end{array}\right] = \left[\begin{array}{ccc|cccccc} x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 & x & x & x & x \\ \hline & & & x & x & x & x & x & x \\ & & & x & x & x & x & x & x \end{array}\right]. \tag{15}$$

These three forms are easily obtained using standard linear algebra methods applied to the system model $\{A, B, C\}$. For each case the matrix U in fact consists of a sequence of Givens transformations or Householder reflections and appropriate algorithm descriptions can be found in the literature [18], [15].

For the form (13), one merely uses the (iterative) QR-algorithm that puts $A_s = UAU^*$ in the so-called "Schur form" [18]. The method is numerically stable and

requires roughly $n^2(5kn+p+n)$ flops for computing it. Here k is the average number of QR-steps required for yielding this form and is usually between 1 and 2 [18].

Remark 1. Notice that when real arithmetic is used, one cannot completely triangularize A_s in general, but some 2×2 blocks corresponding to the pairs of complex conjugate eigenvalues of A_s, will remain on diagonal [18].

The other two forms are in fact variants of the so-called "staircase form" [14] and can be obtained in a numerically stable way by (successive) use of Householder transformations [14], [15], [17] in an analogous fashion to their use in the construction of the Hessenberg form of a general matrix [18]. The number of flops required to construct the forms (14) or (15) using these Householder transformations is roughly $n^2(3n+m+p)$.

Remark 2. Notice that the algorithm computing the forms (14), (15) is *recursive*, while constructing the Schur form (13) is *iterative*. Considering also the facts that the Schur form is usually more expensive in flops and does not provide as many zeros as the other two forms in the real case, the "Hessenberg"-forms (14), (15) ought to be preferred if no other considerations are of importance.

Remark 3. Note also that for each of these "upper" forms there are corresponding "lower" forms which can be obtained via dual algorithms. A survey of the use of these forms for other applications in system and control theory can be found in [15].

The main idea now is to exploit the additional zeros created in these condensed forms in order to obtain more efficient (i.e., faster) algorithm implementations. We show in the sequel how to do this for the different Kalman filtering implementations given in the next section. With these applications in mind, we will therefore only consider real matrices and transformations in the sequel.

4. Classical KF implementations. The original algorithm for computing the KF recursion is the one that implements straightforwardly formulas (6)-(11). Other implementations that became very popular are the so-called Square Root Filters (SRFs). The SRF algorithms use the Choleski factors of the covariance matrices or their inverse in order to solve the optimal filtering problem. Since the process noise covariance matrix Q_k and the measurement noise covariance matrix R_k are assumed to be positive definite, the following Choleski factorizations exist:

$$Q_k = Q_k^{1/2}[Q_k^{1/2}]', \qquad R_k = R_k^{1/2}[R_k^{1/2}]', \tag{16}$$

where the factors $Q_k^{1/2}$ and $R_k^{1/2}$ may be chosen *upper* or *lower* triangular. This freedom of choice is exploited in the development of the fast KF implementations presented in § 5. Notice that historically $Q_k^{1/2}$ and $R_k^{1/2}$ have erroneously been called "square roots" instead of "Choleski factors." However, we will maintain the adjective "square root" as far as the names of the filters are concerned because of the familiarity that they have acquired.

4.1. The square root covariance filter (SRCF). Square root covariance filters propagate the Choleski factors of the error covariance matrix $P_{k|k-1}$:

$$P_{k|k-1} = S_k . S_k', \tag{17}$$

where S_k is chosen to be lower triangular. The computational method is summarized by the following scheme [1]:

$$\underbrace{\begin{bmatrix} R_k^{1/2} & C_kS_k & 0 \\ 0 & A_kS_k & B_kQ_k^{1/2} \end{bmatrix}}_{\text{(pre-array)}} . U_1 \doteq \underbrace{\begin{bmatrix} R_{e,k}^{1/2} & 0 & 0 \\ G_k & S_{k+1} & 0 \end{bmatrix}}_{\text{(post-array)}}, \tag{18}$$

$$\hat{x}_{k+1/k} = A_k\hat{x}_{k|k-1} - G_kR_{e,k}^{-1/2}(C_k\hat{x}_{k|k-1} - y_k) + D_ku_k, \tag{19}$$

where U_1 is an orthogonal transformation that triangularizes the pre-array. Such a triangularization can, for example, be obtained using Householder transformations [4]. This recursion is now initiated with $\hat{x}_{0|-1}$ and the Choleski factor S_0 of $P_{0|-1}$ as defined in (17). The number of flops needed for (18) and (19) is given in Table 2. The total number of flops for one recursion step is here:

$$\frac{7}{6}n^3+n^2\left(\frac{5p}{2}+m\right)+n\left(p^2+\frac{m^2}{2}\right). \tag{20}$$

In order to reduce the amount of work, we only computed the diagonal elements of the covariance matrix $P_{k+1|k}$, since usually diag $\{P_{k+1|k}\}$ carries enough information about the estimate $\hat{x}_{k+1|k}$. For this reason our operation counts differ, for example, from those of [9].

TABLE 2
Numerical implementation of the SRCF.

	Mathematical expression	Number of flops
M_{23}	$B_kQ_k^{1/2}$	$nm^2/2$
$\begin{bmatrix} M_{12} \\ M_{22} \end{bmatrix}$	$\begin{bmatrix} C_k \\ A_k \end{bmatrix} S_k$	$n^2(p+n)/2$
	Annihilation M_{12} (Householder)	$2n^2p+np^2$
	Further reduction (Householder)	$2/3n^3+n^2m$
diag $\{P_{k+1|k}\}$	diag $\{S_{k-1}\cdot S'_{k+1}\}$	$n^2/2$
$\hat{x}_{k+1|k}$	$A_k\cdot\hat{x}_{k|k-1}-G_k\cdot R_{e,k}^{-1/2}\cdot(C_k\cdot\hat{x}_{k|k-1}-y_k)+D_k\cdot u_k$	$n^2+2np+p^2/2+nr$

4.2. The Chandrasekhar square root filter (CSRF). If the system model (1)–(2) is *time invariant*, the SRCF described in § 4.1 may be simplified to the Chandrasekhar square root filter, described in [10]. The Chandrasekhar filter [7] formulates recursions for the *increment* of the covariance matrix, defined as

$$\text{inc } P_k \doteq P_{k+1|k}-P_{k|k-1}. \tag{21}$$

In general this matrix can be factored as

$$\text{inc } P_k \doteq L_k\cdot\underbrace{\begin{bmatrix} I_{n_1} & 0 \\ 0 & -I_{n_2} \end{bmatrix}}_{\Sigma}\cdot L'_k, \tag{22}$$

where the rank of inc P_k is n_1+n_2 and Σ is called its signature matrix. The CSRF propagates recursions for L_k, through the following scheme [10]:

$$\underbrace{\begin{bmatrix} R_{e,k-1}^{1/2} & CL_{k-1} \\ G_{k-1} & AL_{k-1} \end{bmatrix}}_{\text{(pre-array)}}\cdot U_2 \doteq \underbrace{\begin{bmatrix} R_{e,k}^{1/2} & 0 \\ G_k & L_k \end{bmatrix}}_{\text{(post-array)}}, \tag{23}$$

with $L_0\Sigma L'_0=P_{1|0}-P_{0|-1}$. Here U_2 is a Σ_p-unitary transformation, i.e., $U_2\Sigma_pU'_2=\Sigma_p$, with

$$\Sigma_p=\begin{bmatrix} I_p & 0 \\ 0 & \Sigma \end{bmatrix}. \tag{24}$$

Such transformations are easily constructed using "skew Householder" transformations (using the indefinite Σ-norm) and require as many operations as the classical Householder transformations [10]. However, numerical errors may be proportional to the

condition number $\kappa(U_2)$ of the transformation that is applied and this may become very large for some problems. The estimate $\hat{x}_{k+1|k}$ is again computed from its previous value via

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} - G_k R_{e,k}^{-1/2}(C\hat{x}_{k|k-1} - y_k) + Du_k. \tag{25}$$

For this implementation the operation count is summarized in Table 3. The total number of flops for one recursion step is

$$(n_1+n_2)(n^2+3np+p^2). \tag{26}$$

TABLE 3
Numerical implementation of the CSRF.

	Mathematical expression	Number of flops
$\begin{bmatrix} M_{12} \\ M_{22} \end{bmatrix}$	$\begin{bmatrix} C \\ A \end{bmatrix} L_{k-1}$	$(n_1+n_2)n(p+n)$
	Annihilation M_{12} (skew Householder)	$(n_1+n_2)(p^2+2np)$
diag $\{P_{k+1\|k}\}$	diag $\{P_{k\|k-1} + L_k \cdot \Sigma \cdot L_k'\}$	$(n_1+n_2)n$
$\hat{x}_{k+1\|k}$	$A \cdot \hat{x}_{k\|k-1} - G_k \cdot R_{e,k}^{-1/2} \cdot (C \cdot \hat{x}_{k\|k-1} - y_k) + D \cdot u_k$	$n^2+2np+p^2/2+nr$

4.3. The square root information filter (SRIF). The information filter accentuates the recursive least squares nature of filtering [2]. The SRIF propagates the Choleski factor of $P_{k|k}^{-1}$ using the Choleski factor of the inverses of the process- and measurement noise covariance matrices:

$$P_{k|k}^{-1} = T_k' \cdot T_k, \tag{27}$$

$$Q_k^{-1} = [Q_k^{-1/2}]' \cdot Q_k^{-1/2}, \tag{28}$$

$$R_k^{-1} = [R_k^{-1/2}]' \cdot R_k^{-1/2}, \tag{29}$$

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

$$U_3 . \underbrace{\begin{bmatrix} Q_k^{-1/2} & 0 & 0 \\ T_k A_k^{-1} B_k & T_k A_k^{-1} & T_k \hat{x}_{k|k} \\ 0 & R_{k+1}^{-1/2} C_{k+1} & R_{k+1}^{-1/2} y_{k+1} \end{bmatrix}}_{\text{(pre-array)}} \doteq \underbrace{\begin{bmatrix} Q_{e,k+1}^{-1/2} & * & * \\ 0 & T_{k+1} & \hat{\xi}_{k+1|k+1} \\ 0 & 0 & r_{k+1} \end{bmatrix}}_{\text{(post-array)}}. \tag{30}$$

The operation counts are given in Table 4 and the total number of flops of one recursion step is

$$\frac{7}{6}n^3 + n^2\left(p + \frac{7m}{2}\right) + n\left(\frac{p^2}{2} + m^2\right). \tag{31}$$

TABLE 4
Numerical implementation of the SRIF.

	Mathematical expression	Number of flops
$[M_{21}\|M_{22}]$	$T_k[A_k^{-1}B_k\|A_k^{-1}]$	$n^2(3m+n)/2$
M_{32}	$R_{k+1}^{-1/2}C_{k+1}$	$np^2/2$
M_{23}	$T_k\hat{x}_{k\|k}$	$n^2/2$
M_{33}	$R_{k+1}^{-1/2}y_{k+1}$	$p^2/2$
	Annihilation M_{21} (Householder)	$2n^2m+nm^2$
	Further reduction (Householder)	$2/3n^3+n^2p$
$\hat{x}_{k+1\|k+1}$	$T_{k+1}^{-1} \cdot \hat{\xi}_{k+1\|k+1} + D_k \cdot u_k$	$n^2/2+nr$

Here we did *not* count the operations needed for the inversion and/or factorization of Q_k, R_k and A_k (for the time invariant case, e.g., these are computed only once) and the filtered state estimate is computed via

$$\hat{x}_{k+1|k+1} = T_{k+1}^{-1}\hat{\xi}_{k+1|k+1} + D_k u_k. \tag{32}$$

5. New efficient KF implementations. The use of condensed forms in the different KF implementations leads to a considerable speedup when the number of inputs m and/or outputs p is significantly smaller than the number of states n, a reasonable condition that we tacitly assume to hold when comparing the various implementations. The computations that are required in the CKF, (6)–(11) are mainly matrix multiplications. This is also true for the construction of the pre-array in the different SRFs. However, beside these multiplications the SRF also involve unitary or Σ-unitary transformations. In the execution of the matrix multiplication it is easily seen that a lot of profit may be gained from the use of condensed forms. For the conventional KF this leads to a reduction of the most time-consuming operation, namely the product $A \cdot P_{k|k-1} \cdot A'$, roughly by a factor of 2 if the (lower or upper) Schur form is used. About the same improvement is obtained when using the other condensed forms since A is essentially triangular in each of these cases. These operation counts for the CKF are listed at the end of this section in Table 9.

The reduction of the number of computations by the use of condensed forms is now outlined for the SRCF. First, we consider the *lower* Schur form, where the number of operations required to construct the pre-array (18) is given in Table 5 (we only mention the *differences* with Table 2). If we choose additionally that the factors $R^{1/2}$, $Q^{1/2}$, and S_k are lower triangular, the pre-array has the following special structure (illustrated again for the system dimensions $n=6$, $m=3$, $p=2$):

$$\left[\begin{array}{cc|cccccc|ccc}
\times & 0 & x_{11} & x_9 & x_7 & x_5 & x_3 & x_1 & 0 & 0 & 0 \\
x & \times & x_{12} & x_{10} & x_8 & x_6 & x_4 & x_2 & 0 & 0 & 0 \\
\hline
0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & x_{13} & x_{13} & x_{13} \\
0 & 0 & x & \times & 0 & 0 & 0 & 0 & x_{14} & x_{14} & x_{14} \\
0 & 0 & x & x & \times & 0 & 0 & 0 & x_{15} & x_{15} & x_{15} \\
0 & 0 & x & x & x & \times & 0 & 0 & x_{16} & x_{16} & x_{16} \\
0 & 0 & x & x & x & x & \times & 0 & x_{17} & x_{17} & x_{17} \\
0 & 0 & x & x & x & x & x & \times & x_{18} & x_{18} & x_{18}
\end{array}\right], \tag{33}$$

from which we see that many elements that need to be annihilated are already zero. This particular structure of the pre-array is exploited to derive an efficient triangularization, which is briefly outlined for the example. The annihilation of the elements x_1

TABLE 5
Lower Schur implementation of the SRCF.

	Mathematical expression	Number of flops
M_{23}	$BQ^{1/2}$	–
$\begin{bmatrix} M_{12} \\ M_{22} \end{bmatrix}$	$\begin{bmatrix} C \\ A \end{bmatrix} S_k$	$n^3/6+n^2p/2$
	Annihilation M_{12} using Givens	$2np^2+2n^2p$
	Further reduction using Householder	n^2m
$\hat{x}_{k+1\|k}$	$U'\hat{x}_{t,k+1\|k}$	n^2

to x_{12} of the (1, 2) block in the pre-array (18) is done with *Givens transformations* in the order denoted by the subscripts and using the × signs in the corresponding row as pivots. In this way the triangular shape of the (2, 2) block is preserved. Further elimination of all elements x_{13} to x_{18} is then done by using *Householder reflections*, again using the × signs in the corresponding rows as pivots. The total cost of these transformations is given below (for the details see Table 5):

$$\frac{1}{6}n^3+n^2\left(\frac{5p}{2}+m\right)+n(2p^2). \tag{34}$$

Similar arguments also lead to an efficient implementation of the SRIF algorithm when using an *upper* Schur form for the model and upper triangular Choleski factors for T_k, $R^{-1/2}$, and $Q^{-1/2}$. Assuming A^{-1} (which is also upper triangular), $A^{-1}B$, and $R^{-1/2}C$ to be computed once and for all, then the work for constructing the pre-array (30) is given in Table 6 (where again we only noted the differences with Table 4). The triangularization is dual to that of the SRCF, since both arrays look like each other's transpose, the pattern of zeros included. We illustrate this array below for the case $n=6$, $m=2$, $p=3$:

$$\left[\begin{array}{cc|cccccc|c}
\times & x & x & x & x & x & x & x & 0 \\
0 & \times & x & x & x & x & x & x & 0 \\
\hline
x_{11} & x_{12} & \times & x & x & x & x & x & x \\
x_9 & x_{10} & 0 & \times & x & x & x & x & x \\
x_7 & x_8 & 0 & 0 & \times & x & x & x & x \\
x_5 & x_6 & 0 & 0 & 0 & \times & x & x & x \\
x_3 & x_4 & 0 & 0 & 0 & 0 & \times & x & x \\
x_1 & x_2 & 0 & 0 & 0 & 0 & 0 & \times & x \\
\hline
0 & 0 & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x \\
0 & 0 & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x \\
0 & 0 & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x
\end{array}\right]. \tag{35}$$

The operation counts are thus similar to Table 5 (when replacing p by m) and are summarized in Table 6. The total number of flops is

$$\frac{1}{6}n^3+n^2\left(p+\frac{5m}{2}\right)+n(2m^2). \tag{36}$$

Remark 4. Note that for real matrices the (2, 2) block in the pre-arrays (33) and (35) is not exactly triangular but contains a few 2×2 diagonal blocks. Coping with these few additional off-diagonal elements is quite simple and only requires minor modifications in the algorithm.

TABLE 6
Upper Schur implementation of the SRIF.

	Mathematical expression	Number of flops
M_{32}	$R^{-1/2}C$	-
$[M_{21}\|M_{22}]$	$T_k[A^{-1}B\|A^{-1}]$	$n^3/6+n^2m/2$
	Annihilation M_{21} using Givens	$2nm^2+2n^2m$
	Further reduction using Householder	n^2p
$\hat{x}_{k+1\|k+1}$	$U'\hat{x}_{t,k+1\|k+1}$	n^2

A further improvement can now be obtained for the SRCF using a *lower* observer-Hessenberg form for the model since now Householder transformations can be used throughout instead of (slower) Givens transformations. The pattern of zeros after a permutation (i.e., a unitary transformation) of the second and third block columns of the pre-array (18) now indeed looks like (for the same choice of dimensions as above)

$$
(37)\qquad \left[\begin{array}{cc|ccc|ccccc}
\times & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 \\
x & \times & 0 & 0 & 0 & x_2 & x_2 & 0 & 0 & 0 \\
\hline
0 & 0 & \times & x_3 & x_3 & x_3 & x_3 & x_3 & 0 & 0 \\
0 & 0 & x & \times & x_4 & x_4 & x_4 & x_4 & x_4 & 0 \\
0 & 0 & x & x & \times & x_5 & x_5 & x_5 & x_5 & x_5 \\
0 & 0 & x & x & x & \times & x_6 & x_6 & x_6 & x_6 \\
0 & 0 & x & x & x & x & \times & x_7 & x_7 & x_7 \\
0 & 0 & x & x & x & x & x & \times & x_8 & x_8
\end{array}\right].
$$

and all the elements x_1 up to x_8 can be eliminated using Householder reflections only (using the $\times$ signs in the corresponding rows as pivots). The operation count for this is given in Table 7 (only the differences with Table 2 are listed) and the total computational cost is

$$
(38)\qquad \frac{1}{6}n^3+n^2\left(\frac{3p}{2}+m\right)+n(2p^2)+\frac{2p^3}{3}.
$$

Similar significant savings with the controller-Hessenberg form can apparently *not* be obtained for the SRIF, for the reason that A^{-1} and $A^{-1}B$ are now full matrices despite the sparsity of A and B. Yet one can construct a unitary transformation that condenses the pair $(A^{-1}, A^{-1}B)$ to controller-Hessenberg form instead of the pair (A, B). This of course requires the additional work to construct the matrices A^{-1} and $A^{-1}B$ before condensing them to the (lower) form (15), 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 pre-array (30), it has the form

$$
(39)\qquad \left[\begin{array}{cc|cccccc|c}
\times & x & x & x & x & x & x & x & 0 \\
0 & \times & x & x & x & x & x & x & 0 \\
\hline
0 & 0 & \times & x & x & x & x & x & x \\
0 & 0 & x_3 & \times & x & x & x & x & x \\
0 & 0 & x_3 & x_4 & \times & x & x & x & x \\
\hline
x_1 & x_2 & x_3 & x_4 & x_5 & \times & x & x & x \\
0 & x_2 & x_3 & x_4 & x_5 & x_6 & \times & x & x \\
0 & 0 & x_3 & x_4 & x_5 & x_6 & x_7 & \times & x \\
0 & 0 & 0 & x_4 & x_5 & x_6 & x_7 & x_8 & x \\
0 & 0 & 0 & 0 & x_5 & x_6 & x_7 & x_8 & x \\
0 & 0 & 0 & 0 & 0 & x_6 & x_7 & x_8 & x
\end{array}\right].
$$

This then clearly leads to an analogue of the previous method for the SRIF, which per step costs (details are given in Table 8, where again we only list the differences with Table 4)

$$
(40)\qquad \frac{1}{6}n^3+n^2\left(p+\frac{5m}{2}\right)+n(2m^2).
$$

TABLE 7
Lower observer-Hessenberg implementation of the SRCF.

	Mathematical expression	Number of flops
M_{23}	$BQ^{1/2}$	-
$\begin{bmatrix} M_{13} \\ M_{23} \end{bmatrix}$	$\begin{bmatrix} C \\ A \end{bmatrix} S_k$	$n^3/6+pn^2/2$
	Further reduction using Householder	$n^2(m+p)+2np^2+2/3p^3$
$\hat{x}_{k+1\|k}$	$U'\hat{x}_{t,k+1\|k}$	n^2

Finally, it can be checked that condensed forms do not yield any substantial savings in operations for the CSRF because the L_k matrices are usually full. The only savings are indeed obtained in the construction of the products AL_{k-1} and CL_{k-1}, whose complexity is reduced by a factor 2 (for both the Schur form and the observer-Hessenberg form).

A complete list of the total operations counts for the different methods is given in Table 9 for comparison. Here we have not included second-order terms which e.g., come from the calculation of the estimates $\hat{x}_{k+1|k}$ or $\hat{x}_{k+1|k+1}$ in these different methods or from the back transformation $\hat{x} = U'\hat{x}_t$ to the original coordinate system.

Remark 5. When the input/output dimensions are large, i.e., $m, p \geqq n$, more efficient SRCF and SRIF implementations are obtained when combining these condensed forms with "sequential processing" that is used in [2] for the "full" system

TABLE 8
Controller-Hessenberg implementation of the SRIF.

	Mathematical expression	Number of flops
M_{32}	$R^{-1/2}C$	-
$[M_{21}\|M_{22}]$	$T_k[A^{-1}B\|A^{-1}]$	$n^3/6+n^2m/2$
	Further reduction using Householder	$n^2(p+m)+2nm^2+2/3m^3$
$\hat{x}_{k+1\|k+1}$	$U'\hat{x}_{t,k+1\|k+1}$	n^2

TABLE 9
Operation counts for the different KFs.

Filter	Type	Complexity
CKF	full	$(3/2)n^3+n^2(3p+m/2)+n(3p^2/2+m^2)+p^3/6$
	Schur	$(3/4)n^3+n^2(5p/2+m/2)+n(3p^2/2+m^2)+p^3/6$
	o-Hess.	$(3/4)n^3+n^2(7p/2+m/2)+n(2p^2+m^2)+p^3/6$
SRCF	full	$(7/6)n^3+n^2(5p/2+m)+n(p^2+m^2/2)$
	Schur	$(1/6)n^3+n^2(5p/2+m)+n(2p^2)$
	o-Hess.	$(1/6)n^3+n^2(3p/2+m)+n(2p^2)+2p^3/3$
SRIF	full	$(7/6)n^3+n^2(p+7m/2)+n(p^2/2+m^2)$
	Schur	$(1/6)n^3+n^2(p+5m/2)+n(2m^2)$
	c-Hess.	$(1/6)n^3+n^2(3m/2+p)+n(2m^2)+2m^3/3$
CSRF	full	$(n_1+n_2)(n^2+3np+p^2)$
	Schur	$(n_1+n_2)(n^2/2+3np+p^2)$
	o-Hess.	$(n_1+n_2)(n^2/2+3np+p^2)$

models. Since there are no essential differences with the savings obtained when applying sequential processing to full or condensed models we do not treat these techniques here, but refer to [2]. Moreover, in practice the input/output dimensions are much smaller than the state dimension n and these additional savings are then not so significant.

The operation counts derived above for the different KF implementations demonstrate the reduction in computational complexity of the CKF, the SRCF, the SRIF and the CSRF. For $n \gg m$ and p, the dominant terms in n of Table 9 indicate that the condensed implementations lead to improvements of roughly a factor 7 for the SRCF and the SRIF and a factor 2 for the CKF and the CSRF.

Where these reductions are coming from is easily understood in terms of basic linear algebra operations. The most time-consuming substep in the CKF- and the CSRF-recursion is the multiplication of $n \times n$ matrices (respectively, $A \cdot P_{k|k+1} \cdot A'$ and $A \cdot L_k$). If A is now in condensed form, this step is performed in roughly half the number of operations. For the SRCF- and the SRIF-recursion the situation is different. There the bulk of the operations lies in *two* substeps: in the multiplication of $n \times n$ matrices (respectively, $A \cdot S_k$ and $T_k \cdot A^{-1}$) *and* in the QR-decomposition of an $O(n) \times O(n)$ matrix (respectively, $[M_{22} | M_{23}]$ and $[M_{22} / M_{32}]$). If one now uses condensed forms the multiplication is again performed in roughly half the number of operations, but the QR-decomposition *becomes an order of magnitude in n cheaper* because the considered matrices are already nearly triangular. Details of this can easily be followed by comparing the tables 1 to 8.

That these factors are not unrealistic in actual run tests is illustrated in Fig. 1. The two curbs (full lines) shown in this figure are the theoretical ratios a/b between the complexity of the full (a) and Schur (b) versions of the SRCF and CSRF, respectively, as derived from Table 9 for the value $m = 3, p = 2$ and a variable value for the state dimension n. As explained above, these curbs approach the limits 7 and 2 (dotted lines), respectively, as n becomes larger. This is of course only a theoretical

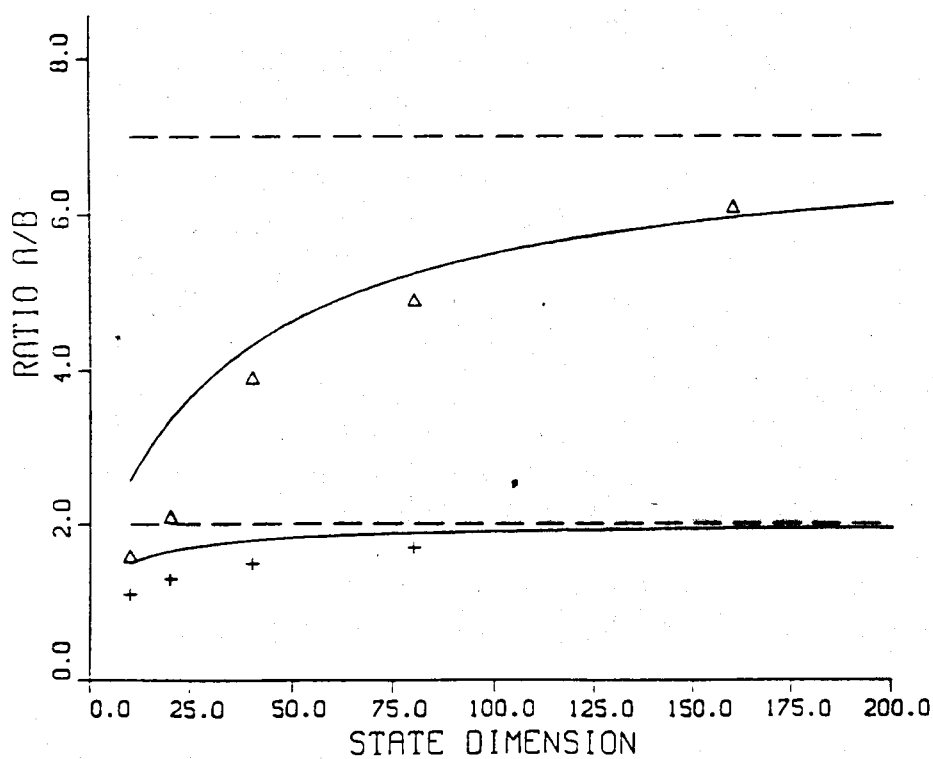

FIG. 1. *Improvement factor of condensed forms.*

ratio since it is based on the operation counts of Table 9. But on the same figure we also plotted *actually measured* ratios a/b. We ran indeed 100 steps of the SRCF recursion for matrices of dimension $m = 3$, $p = 2$, and $n = 10, 20, 40, 80, 160$ and plotted the observed ratios between full and condensed implementations as plusses (+) in Fig. 1. The same was done for the CSRF, this for values $n = 10, 20, 40, 80$, and was plotted as triangles ($\triangle$). The real observed ratios marked on this figure are clearly close to their theoretical values (full curbs) which means that overhead costs are not very significant here. Similar results were also observed for the other two filters and for ratios involving the Hessenberg forms.

Bearing in mind that the operation counts of Table 9 can be considered to be realistic, one can now recommend, as far as computational complexity is concerned, the choice of SRCF/SRIF and CSRF implementations when the state dimension n is considerably larger than the input/output dimensions (which is often the case in practice). The CSRF becomes particularly attractive with the initial condition $P_{0|-1} = 0$ because then $(n_1 + n_2)$ is of the order of p and thus also much smaller than n [10]. For general initial conditions, though, $(n_1 + n_2) = n$ and the condensed SRCF/SRIF become more economical.

As far as the numerical stability is concerned, it is shown in [16], [17] that the CSRF is numerically less reliable and should be avoided when a large number of iterations is requested. The propagation of numerical errors from one recursion step to another can indeed not be bounded in general and eventually divergence may occur between the *computed* recursive solution and the *true* one. In contrast to this, the CKF, the SRCF, and the SRIF in general do *not* suffer from this divergence phenomenon. Indeed, bounds for the accumulated errors where derived in [16], [17], showing that they remain independent of the number of recursion steps, with a certain disadvantage for the CKF and the SRIF since for these filters the bounds of the residual errors in each step are worse. These differences were also observed in practice [16], [17].

As a consequence, we recommend in general the use of the condensed SRCF for the computations of the time-invariant KF. Between the two possible condensed forms the observer-Hessenberg form is to be preferred because of a lower complexity in both the initial reduction and the subsequent recursions (see Table 9). Moreover, it does not require any special handling in the case of real arithmetic (see Remark 4).

6. Concluding remarks. In this paper we have analyzed four different KF algorithms and some new variants for their computational efficiency. The use of "condensed forms" for the implementation of different KFs was derived here and shown to yield very efficient algorithms for the problems considered in this paper.

It should be warned here that these techniques are not always advantageous. Indeed, when the model $\{A, B, C\}$ is already *sparse*, a unitary transformation is very likely to increase dramatically the fill-in, and the complexity of the resulting algorithm would probably increase a lot. Here it is more indicated to use techniques for sparse matrix multiplications and decompositions in order to obtain cost-effective algorithms.

Also, for badly scaled models $\{A, B, C\}$ it is not indicated to use *directly* unitary transformations. Indeed, these would "smear out" the larger size elements over the matrices and the higher accuracy with which the smaller elements were originally known, would get lost. Here, it is rather indicated to first "rescale" the model via diagonal state space transformation (in order to obtain a well-scaled model) and after that apply unitary transformations to condense the model.

The assumption that Q_k is nonsingular does not restrict the generality of the system description, since for the case of singular Q_k, the linear dependent components in w_k

can always be removed first [1]. On the other hand, the regularity of R_k rules out the possibility of including perfect measurements not corrupted by noise. The particular case of perfect measurements leads to special adaptations of the different KF implementations, such as the use of the Moore-Penrose inverse [1]. These special implementations were not considered here.

In [11], the Kalman filter recursions are extended to the problem of computing other estimates $\hat{x}_{k|j}$ (e.g., for smoothing) which can probably be implemented using condensed forms as well. Here the author also introduces mixed representations of covariances. These have the advantage to allow for singular covariance matrices Q_k, R_k, or $P_{k|k-1}$ and even for singular information matrices $I_{k|k} = P_{k|k}^{-1}$, thereby avoiding any use of generalized inverses. It would be worthwhile to investigate the possible use of condensed forms in these extensions.

Other extensions are obtained by the incorporation of sequential processing into these condensed algorithms [2] (see Remark 5) or by using these algorithms also for time-varying models which yet can be condensed by a time-invariant transformation. That this is possible is easily understood for system models of the type $\{A, B_k, C\}$ or $\{A, B_k, C_k\}$, where only one or two of the matrices is time varying. It is clear that these can still be condensed to observer-Hessenberg form and Schur form, respectively. But in practice one even encounters models $\{A_k, B_k, C_k\}$ which can be condensed by a time-invariant transformation. These typically occur in models where the parameters have some physical meaning and where time-varying parameters are related to each other (see [17] for a real example coming from aeronautics).

Finally, it is also shown in [17] that similar techniques can be extended to provide an efficient treatment of the so-called "biases" that are often included in the system model for which Kalman filtering is used.

REFERENCES

[1] B. B. O. ANDERSON AND J. B. MOORE, *Optimal Filtering*, Information and System Sciences Series, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[2] G. J. BIERMAN, *Factorization Methods For Discrete Sequential Estimation*, Academic Press, New York, 1976.

[3] A. E. BRYSON, *Kalman filter divergence and aircraft motion estimators*, Internat. J. Guidance and Control, 1 (1978), pp. 71–79.

[4] G. H. GOLUB, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[5] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1983.

[6] A. H. JAZWINSKI, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[7] T. KAILATH, *Some new algorithms for recursive estimation in constant linear systems*, IEEE Trans. Inform. Theory, IT-19 (1973), pp. 750–760.

[8] R. E. KALMAN, *A new approach to linear filtering and prediction problems*, Trans. ASME (J. Basic Engrg.), 82D (1960), pp. 34–45.

[9] P. G. KAMINSKI, A. E. BRYSON, AND S. F. SCHMIDT, *Discrete square root filtering. A survey of current techniques*, IEEE Trans. Automat. Control, AC-16 (1971), pp. 727–736.

[10] M. MORF AND T. KAILATH, *Square-root algorithms for least squares estimation*, IEEE Trans. Automat. Control, AC-20 (1975), pp. 487–497.

[11] C. C. PAIGE, *Covariance matrix representation in linear filtering*, in the Special Issue of Contemporary Mathematics on Linear Algebra and its Role in Systems Theory, American Mathematical Society, Providence, RI, 1985.

[12] C. C. PAIGE AND M. SAUNDERS, *Least squares estimation of discrete linear dynamic systems using orthogonal transformations*, SIAM J. Numer. Anal., 14 (1977), pp. 180–193.

[13] J. E. POTTER AND R. G. STERN, *Statistical filtering of space navigation measurements*, Proc. 1963 AIAA Guidance and Control Conference, 1963.

[14] P. VAN DOOREN, *The generalized eigenstructure problem in linear system theory*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 111-129.

[15] P. VAN DOOREN AND M. VERHAEGEN, *On the use of unitary state-space transformations*, in the Special Issue of Contemporary Mathematics on Linear Algebra and its Role in Systems Theory, American Mathematical Association, Providence, RI, 1985.

[16] M. VERHAEGEN AND P. VAN DOOREN, *Numerical aspects of different Kalman filter implementations*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 907-917.

[17] M. VERHAEGEN, *A new class of algorithms in linear system theory with application to real-time aircraft model identification*, Ph.D. dissertation, Catholic University of Leuven, Belgium, 1985.

[18] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.