Proceedings of the 31st Conference
on Decision and Control
Tucson, Arizona • December 1992

KALMAN FILTER ALGORITHM BASED ON SINGULAR VALUE DECOMPOSITION

Liang Wang, Gaëtan Libert and Pierre Manneback
Department of Computer Science
Faculte Polytechnique de Mons, Rue de Houdain 9
7000 Mons, Belgium

Abstract: This paper develops a new algorithm for the discrete time linear filtering problem. The crucial component of this algorithm involves the computation of the singular value decomposition (SVD) of an unsymmetric matrix - without explicitly forming its left factor that has a high dimension. The presented algorithm has a good numerical stability and can handle correlated measurement noise without any additional transformation. This algorithm is formulated in the form of vector-matrix and matrix-matrix operations, so it is also useful for parallel computers. Details of the algorithm are provided and a numerical example is given.

I. INTRODUCTION

The Kalman filter [14] has been one of the most widely applied techniques in the area of modern control, signal processing, and communication applications. By using a state space model, it facilitates the estimation of the unknown state vector recursively for each new observation. It is considered an optimal estimator because it provides a minimum variance estimate. Discussion and applications on Kalman filter can be found in many literature.

The following equations define a discrete time state space system:

$$x_{k+1} = \Phi_{k+1,k}\, x_k + G_k w_k \tag{1}$$

$$z_k = H_k\, x_k + v_k \tag{2}$$

where $x_k \in \Re^n$ is the state vector, $z_k \in \Re^m$ is the measurement vector, $w_k \in \Re^s$ is the disturbance input vector, $v_k \in \Re^s$ is the measurement noise vector, $\Phi_{k+1,k} \in \Re^{n \times n}$, $G_k \in \Re^{n \times s}$ and $H_k \in \Re^{n \times s}$ are the system matrices. The disturbance w_k and noise v_k are assumed to be zero mean Gaussian white noise sequences with symmetric positive definite covariance matrices Q_k and R_k, respectively. Furthermore, sequences w_k and v_k are assumed to be statistically independent. The Kalman filter is then described by the following recursive equations under assumptions that matrices $\Phi_{k+1,k}$, G_k, H_k, Q_k, and R_k are known:

Time extrapolation:

$$\hat{x}_{k+1} = \Phi_{k+1,k}\, \hat{x}_k^+ \tag{3}$$

$$P_{k+1} = \Phi_{k+1,k} P_k^+ \Phi_{k+1,k}^T + G_k Q_k G_k^T \tag{4}$$

Measurement update:

$$\hat{x}_k^+ = \hat{x}_k + K_k (z_k - H_k \hat{x}_k) \tag{5}$$

$$P_k^+ = P_k - K_k H_k P_k \tag{6}$$

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \tag{7}$$

where P_k is the covariance of estimation uncertainty, superscript + refers to values after the measurement update, and K_k is the Kalman gain matrix.

The major disadvantage of the Kalman formulation is that the matrix subtraction in Eq.(6), representing the reduction in uncertainty due to the measurement, can yield a result P_k^+ that is computationally not positive definite (or, at least, nonnegative) -- a theoretical impossibility. To circumvent this difficulty, Potter [2] introduced the idea of using a square root of the covariance matrix in the algorithmic implementation. This is a matrix

$$S = P^{1/2} \quad such\ that\ P = SS^T$$

where S is obtained in triangular form by the well-known Cholesky decomposition. Although equivalent algebraically to the conventional Kalman filter recursion, the square root approach exhibits improved numerical precision and stability, particularly in ill-conditioned problems. The advances in square root filtering up to 1971 have been summarized by Kaminski, Bryson and Schmidt [15]. Subsequently, Carlson [6] and Bierman [5] have introduced strictly algorithmic approaches to the square root filtering. Bierman also introduced the idea of using a UDL decomposition of the covariance matrix in place of the square root decomposition. This is a decomposition of the sort

$$P = UDU^T$$

where D is a diagonal matrix and U is an upper triangular matrix with 1's along its main diagonal. This factorization does not require taking scalar square roots and is superior in most respects to the basic square root algorithm [21].

Both the square root and the UDU^T decompositions may result in numerically stable filter algorithms. But these formulations can only be used if one has single dimension measurements with uncorrelated measurement noise. Generally one does not have this in practice. To handle correlated measurement noise, additional transformations have to be used which increase the computation cost. Moreover, these formulations cannot be effectively implemented on vector processors because their designs are virtually serial in structure.

Extensions of the Potter and the Bierman methods to the multiple measurement case have been devised by several researchers [1][3][18]. Especially, in a recent paper by Hotop [13], the author gives a fresh Kalman filter formulation which is based on a special Givens orthogonal transformation. Hotop's algorithm has been shown to be very useful for parallel computers. In the sequel of this paper we will present an SVD-based Kalman filter algorithm which has the UDU^T formulation as in the Bierman method. Like Hotop's algorithm, our algorithm is also suitable for parallel computers, but it has a higher numerical stability than Hotop's and other previous algorithms.

II. SINGULAR VALUE DECOMPOSITION AND ITS COMPUTATION

One of the basic and most important tools of modern numerical analysis, particularly numerical linear algebra, is the singular value decomposition. For a survey of the theory and its many interesting

applications, see Vandewalle and De Moor [22].

The singular value decomposition of an m-by-n matrix A($m \geq n$), is a factorization of A into a product of three matrices. That is, there exist orthogonal matrices $U \in \Re^{m \times m}$ and $V \in \Re^{n \times n}$ such that

$$A = U \Lambda V^T, \quad \Lambda = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \tag{8}$$

where $\Lambda \in \Re^{m \times n}$ and $S=\text{diag}(\sigma_1,...,\sigma_r)$ with

$$\sigma_1 \geq ... \geq \sigma_r > 0 .$$

The numbers $\sigma_1,...,\sigma_r$ together with $\sigma_{r+1}=0,...,\sigma_n=0$ are called the singular values of A and they are the positive square roots of the eigenvalues of A^TA. The columns of U are called the left singular vectors of A(the orthonormal eigenvectors of AA^T) while the columns of V are called the right singular vectors of A(the orthonormal eigenvectors of A^TA).

It is known that the singular values and singular vectors of a matrix are relatively insensitive to perturbations in the entries of the matrix, and to finite precision errors [24]. Furthermore, since the σ_i's are, in fact, the eigenvalues of a symmetric matrix, they are guaranteed to be well-conditioned so that, with respect to accuracy, we are in the best of possible situations [16].

In practice, if A^TA is positive definite then (8) can be reduced to

$$A = U \begin{bmatrix} S \\ O \end{bmatrix} V^T \tag{9}$$

where S is an n-by-n diagonal matrix. Especially, if A itself is symmetric positive definite then we will have a symmetric singular value decomposition

$$A = USU^T = UD^2U^T \tag{10}$$

In our filter algorithm derivation, (9) and (10) will be of particularly real interest.

The standard method for computing (8) is the Golub-Kahan-Reinsch SVD algorithm ([9] and [10]), in which the Householder transformation is first used to bidiagonalize the given matrix and then the QR method to compute the singular values of the resultant bidiagonal form. Recently, with the advent of massively parallel computer architectures, two classical SVD computation methods, that is, Hestenes algorithm (one-sided Jocabi) [12] and Kogbetliantz algorithm (two-sided Jocabi) [17], have gained a renewed interest for their inherent parallelism and vectorizability (see a good overview written by Berry and Sameh [4] summarizing parallel algorithms for the singular value and symmetric eigenvalue problems). For illustration, Table I gives the computation flops of Golub-Kahan-Reinsch algorithm (G-K-R), row-oriented Hestenes algorithm (R-Hestenes), column-oriented Hestenes algorithm (C-Hestenes), and Kogbetliantz algorithm (Kogbetliantz) for random n-by-n matrices A whose elements are uniformly distributed in the interval (0, 1)(An initial QR step is done before the SVD procedures are applied to A). All algorithms are implemented in the MATLAB environment and ran on a PC 486 [23]. It can be seen that the Golub-Kahan-Reinsch algorithm is most computationally efficient on the sequential machine. However, this algorithm will become less attractive on a parallel processor [20], while Hestenes algorithm and Kogbetliantz algorithm will be of importance there.

Our present Kalman filter formulation is based on Golub-Kahan-Reinsch algorithm and ran on the sequential machine. In a future paper we will discuss its parallel implementation on a transputer network in which Kogbetliantz's two-sided Jocabi algorithm will be used.

III. NEW KALMAN FILTER FORMULATION

Time Extrapolation Formulation

In the covariance equation (4) of the conventional Kalman filter, assume that the singular value decomposition of P_k^+ is available for all t_k and has been propagated and updated by the filter algorithm. Thus, we have

$$P_k^+ = U_k^+ D_k^{+2} U_k^{+T}$$

Eq.(4) can therefore be written as

$$P_{k+1} = \Phi_{k+1,k} U_k^+ D_k^{+2} U_k^{+T} \Phi_{k+1,k}^T + G_k Q_k G_k^T \tag{11}$$

Our goal is to find the factors U_{k+1} and D_{k+1} from Eq.(11) such that $P_{k+1}=U_{k+1}D_{k+1}^2U_{k+1}^T$, where U factors are orthogonal and D factors are diagonal. Provided that there is no danger of numerical accuracy deterioration, one could, in a brute force fashion, compute P_{k+1} and then apply the singular value decomposition of symmetric positive definite matrix which is given by Eq.(10). However, it has been shown that this is not a good numerical exercise [10]. Instead we define the following (s+n)-by-n matrix

$$\begin{bmatrix} D_k^+ U_k^{+T} \Phi_{k+1,k}^T \\ \sqrt{Q_k}^T G_k^T \end{bmatrix}$$

and compute its singular value decomposition

$$\begin{bmatrix} D_k^+ U_k^{+T} \Phi_{k+1,k}^T \\ \sqrt{Q_k}^T G_k^T \end{bmatrix} = U_k' \begin{bmatrix} D_k' \\ 0 \end{bmatrix} V_k'^T$$

Multiplying each side on the left by its transpose, we have

$$\Phi_{k+1,k} U_k^+ D_k^{+T} D_k^+ U_k^{+T} \Phi_{k+1,k}^T + G_k \sqrt{Q_k} \sqrt{Q_k}^T G_k^T$$

$$= V_k' \begin{bmatrix} D_k'^T \mid 0 \end{bmatrix} U_k'^T U_k' \begin{bmatrix} D_k' \\ 0 \end{bmatrix} V_k'^T$$

TABLE I
AVERAGE NUMBER OF FLOPS FOR DIFFERENT SVD ALGORITHMS

n	Trials	G-K-R	R-Hestenes	C-Hestenes	Kogbetliantz
4	100	1480	2607	2918	3412
6	100	4861	10686	11839	14017
8	100	10992	27841	31061	35489
10	100	20573	57939	63972	75571
20	10	149700	537220	601350	686890
30	10	488670	1973400	2119208	2428400
40	10	1136000	4867800	5162000	5950600

That is

$$\Phi_{k+1,k} U_k^+ D_k^{+2} U_k^{+T} \Phi_{k+1,k}^T + G_k Q_k G_k^T = V_k' D_k'^2 V_k'^T \qquad (12)$$

Comparing the result with (11), we find that V_k' and D_k' are just the U_{k+1} and D_{k+1} we are looking for. Here we want to point out that the (s+n)-by-(s+n) orthogonal matrix U_k' and its transpose $U_k'^T$ are not needed directly in our algorithm and it is not necessary to store or compute them explicitly.

Measurement Update Formulation

In the conventional Kalman measurement update, substituting Eq. (7) into (6) yields

$$P_k^+ = P_k - P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} H_k P_k \qquad (13)$$

To obtain the new measurement update result, we will require use of the well-known *matrix inversion lemma*, valid for positive definite N and M

$$(N+BMB^T)^{-1} = N^{-1} - N^{-1}B(B^T N^{-1} B + M^{-1})^{-1} B^T N^{-1}$$

It follows from this and (13) that

$$(P_k^+)^{-1} = P_k^{-1} + H_k^T R_k^{-1} H_k \qquad (14)$$

Applying the singular value decomposition of symmetric positive definite matrix to P_k^+ and P_k, respectively, we may get

$$(U_k^+ D_k^{+2} U_k^{+T})^{-1} = (U_k D_k^2 U_k^T)^{-1} + H_k^T R_k^{-1} H_k$$

$$= (U_k^T)^{-1} D_k^{-2} U_k^{-1} + (U_k^T)^{-1} U_k^T H_k^T R_k^{-1} H_k U_k U_k^{-1}$$

$$= (U_k^T)^{-1} (D_k^{-2} + U_k^T H_k^T R_k^{-1} H_k U_k) U_k^{-1} \qquad (15)$$

In (15) let

$$L_k L_k^T = R_k^{-1} \qquad (16)$$

be the Cholesky decomposition of the inverse of the covariance matrix. If the inverse is available then there is no difficulty. If the covariance matrix R_k is known, but not its inverse, then the reverse Cholesky decomposition $R_k^{1/2} R_k^{1/2T} = R$, $R_k^{1/2}$ upper triangular, can be found(see, for example, [11]). It then follows that $L_k = (R_k^{1/2T})^{-1}$ is the required Cholesky decomposition in (16).

Now considering the (m+n)-by-n matrix

$$\begin{bmatrix} L_k^T H_k U_k \\ D_k^{-1} \end{bmatrix}$$

and computing its singular value decomposition, we have

$$\begin{bmatrix} L_k^T H_k U_k \\ D_k^{-1} \end{bmatrix} = U_k^* \begin{bmatrix} D_k^* \\ 0 \end{bmatrix} V_k^{*T} \qquad (17)$$

Multiplying each side on the left by its transpose yields

$$D_k^{-2} + U_k^T H_k^T L_k L_k^T H_k U_k = V_k^* D_k^{*2} V_k^{*T} \qquad (18)$$

Then Eq. (15) can be written as

$$(U_k^{+T})^{-1} (D_k^+)^{-2} (U_k^+)^{-1} = (U_k^T)^{-1} V_k^* D_k^{*2} V_k^{*T} U_k^{-1}$$

$$= [(U_k V_k^*)^T]^{-1} D_k^{*2} [U_k V_k^*]^{-1} \qquad (19)$$

Comparing two sides of Eq.(19), we get

$$U_k^+ = U_k V_k^* \qquad (20)$$

$$D_k^+ = (D_k^*)^{-1} \qquad (21)$$

In this manner, a new measurement update formulation has been obtained. The crucial component of the update, like that of time extrapolation, involves the computation of the singular value decomposition of an unsymmetric matrix - without explicitly forming its left orthogonal factor that has a high dimension.

For the Kalman gain an alternative expression may also be derived. Beginning with Eq. (7) we have

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

Insertion of $P_k^+(P_k^+)^{-1}$ and $R_k^{-1}R_k$ will not alter the gain. Thus, K_k can be written as

$$K_k = P_k^+ (P_k^+)^{-1} P_k H_k^T R_k^{-1} R_k (H_k P_k H_k^T + R_k)^{-1}$$

$$= P_k^+ (P_k^+)^{-1} P_k H_k^T R_k^{-1} (H_k P_k H_k^T R_k^{-1} + I)^{-1}$$

We now use Eq. (14) for $(P_k^+)^{-1}$ and get

$$K_k = P_k^+ H_k^T R_k^{-1}$$

$$= U_k^+ D_k^{+2} U_k^{+T} H_k^T L_k L_k^T \qquad (22)$$

There is no need to obtain a formula for the singular value decomposition of K_k.

The state vector measurement update is given by

$$\hat{x}_k^+ = \hat{x}_k + K_k (z_k - H_k \hat{x}_k) \qquad (23)$$

Together with the time extrapolation described in the above section and the measurement update of the covariance matrix and the state vector described here, a new Kalman filter algorithm is formulated, and it is summarized in Fig. 1.

Algorithmic Details

In this section we provide a few details and references for the new algorithm summarized in Fig. 1.

(i) Determine initial U_0 and D_0. In practice, the initial P_0 is generally assumed to be diagonal, in which case we set U_0=I and D_0=P_0. If P_0 is not diagonal then a symmetric QR algorithm [11] can be used to compute the U_0 and P_0, and about $9n^3$ flops are required.

(ii) Update U_k^+ and D_k^+. The key to this step is the construction of the (m+n)-by-n matrix

$$\begin{bmatrix} L_k^T H_k U_k \\ D_k^{-1} \end{bmatrix}$$

and then its SVD computation. Because of the iterative nature of the SVD algorithm it is difficult to give a reliable flop count. In terms of Golub and Van Loan's estimate, forming the explicit matrix product and doing a standard SVD(including accumulating the U and V factors) using the efficient Golub-Reinsch algorithm requires $4(m+n)^2n+8(m+n)n^2+9n^3$ flops. In our filter formulation, only the right SVD factor V is needed, so the practical flop count is $4(m+n)n^2+8n^3=4mn^2+12n^3$. Furthermore, if we notice the fact that the above matrix has the form

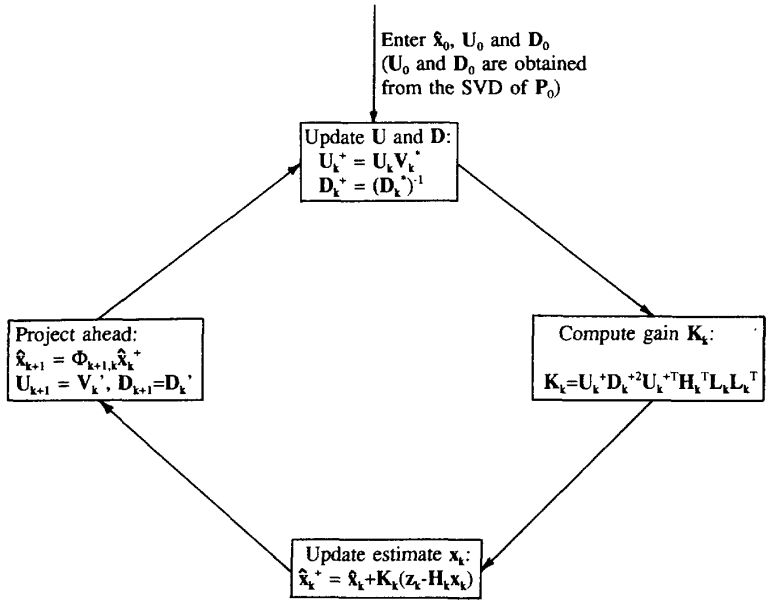

Fig.1. New Kalman filter recursive loop.

$$\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{bmatrix}$$

where * denotes the non-zero element of the matrix, then we can further reduce the computation flops to approximately $5mn^2+10n^3$ by bidiagonalizing this matrix using both Givens rotations and Householder reflections(Here we assume that m is less than n, as it often does in practical Kalman filter applications).

In the above matrix, L_k is the Cholesky decomposition of the inverse of the m-by-m covariance matrix R_k, and its computation requires 1/3 m^3 flops. If the covariance matrix R_k is known, but not its inverse, then L_k can be obtained from the reverse Cholesky decomposition, with an additional 1/3 m^3 flops for computing the inverse of the Cholesky factor[8]. Especially, if the covariance matrix R_k cannot be always guaranteed to be positive definite then the numerical reliable SVD algorithm can be used for computing its inverse(pseudo-inverse)(see [9]). In this case $L_k=U_kD^I$ where D^I is obtained from D by replacing each positive diagonal entry by its reciprocal, and approximately $9m^3$ flops are required for this process.

In summary, together with the approximate m^2n+2mn^2 flops for computing the product of $L_k^TH_kU_k$, this step requires approximately $1/3m^3+m^2n+7mn^2+10n^3$ flops (Assume that the Cholesky decomposition of the inverse of R_k has been used here).

(iii) Compute gain K_k and update estimate $\hat{x}_k^+$. Computations of K_k from (22) and $\hat{x}_k^+$ form (23) are straightforward. The essential calculation for K_k and $\hat{x}_k^+$ is the matrix-matrix and matrix-vector multiplications. Notice that $H_k^TL_k=(L_k^TH_k)^T$ can be obtained from the previous computation and L_k^T is triangular, the flop count for K_k is approximately $m^2n+2mn^2+2n^3$. Calculation of $\hat{x}_k^+$ takes on the order of O(mn) flops and can be computationally negligible.

(iv) Compute the extrapolations $\hat{x}_{k+1}$,U_{k+1}, and D_{k+1}. $\hat{x}_{k+1}$ can be directly computed from Eq. (3) with trivial flops. U_{k+1} and D_{k+1} can be obtained from the singular value decomposition of the (s+n)-by-n matrix

$$\begin{bmatrix} D_k^* U_k^{+T} \Phi_{k+1,k}^T \\ \sqrt{Q_k}^T G_k^T \end{bmatrix}$$

without explicitly forming the (n+s)-by-(n+s) left orthogonal factor, which requires about $4sn^2+12n^3$ flops. Computation of the Cholesky factor $Q_k^{1/2}$ and related matrix-matrix multiplications requires approximately $1/3s^3+s^2n+2n^3$ flops. In summary, a total of approximately $1/3s^3+s^2n+4sn^2+14n^3$ flops are needed in this step.

To summarize, one measurement update and one time extrapolation(not including the initial computations) can be computed by the above procedure in approximately $1/3m^3+2m^2n+9mn^2+1/3s^3+s^2n+4sn^2+26n^3$ flops. Here, we want to point out that the operation count is very rough, and only tribasic amount of work $O(m^3)$, $O(m^2n)$, $O(mn^2)$, $O(s^3)$, $O(s^2n)$, $O(sn^2)$, $O(n^3)$) is considered. Not too much weight should be attached to these flop count estimates. Actual running time of an algorithm depends on the specific language and compiler used, the efficiency of the source and machine code, details of the floating-point arithmetic used, I/O, and a variety of other architecture-dependent details. Extensive empirical experience with a robust software implementation on a specific computing machine is a more reliable guide for the timing associated with a particular algorithm [19].

An Example Problem

An algorithm can be said to be numerically stable if the computed result of the algorithm corresponds to an exactly computed solution to a problem that is only slightly perturbed from the original one. By this criterion, the Kalman filter is numerically unstable in the conventional formulation. Examples illustrating the numerical

TABLE II
COMPARISON OF ROUNDED SOLUTIONS FOR DIFFERENT FILTER IMPLEMENTATIONS

Filter implementation	Rounded solution
Exact value	$\frac{1}{\Delta}\begin{bmatrix} 1+2\varepsilon^2 & -(1+2\varepsilon) \\ -(1+2\varepsilon) & 2+\varepsilon^2 \end{bmatrix}$ $\Delta = 1-2\varepsilon+2\varepsilon^2(2+\varepsilon^2)$
Conventional Kalman	$\frac{1}{\Delta_r}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $\Delta_r = 1-2\varepsilon$
Potter square root	$\begin{bmatrix} (1+2\varepsilon)^{-1} & -(1-3\varepsilon)\Delta_r^{-1} \\ -(1-3\varepsilon)\Delta_r^{-1} & 2 \end{bmatrix}$*
Bierman UDU^T factorization	$\frac{1}{\Delta_r}\begin{bmatrix} 1 & -(1+\varepsilon) \\ -(1+\varepsilon) & 2 \end{bmatrix}$*
SVD formulation	$\begin{bmatrix} .9999 & -.9999 \\ -.9999 & 1.9999 \end{bmatrix}$*

* These covariances are not part of algorithm. Only the updated factors are needed.

divergence of the conventional Kalman filter algorithm can be found in [3], [5] and [15]. To appreciate and compare the performance of the new algorithm and other filter implementations, we include a simple but illuminating example which is taken from [5].

We are to estimate x_1 and x_2 from the measurements

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon \\ 1 & \varepsilon \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

where the measurements have been normalized so that the v error terms have zero mean and unity covariance. $0<\varepsilon\ll 1$ and is small enough so that in terms of machine computation $1+\varepsilon>1$, $1+\varepsilon^2\doteq 1$. The a priori estimate and covariance for x are assumed to be zero and $P=\sigma^2 I$. We can take σ large because this parameter estimation problem is well defined even if there were no a priori information about x. To keep the number of free parameters to a minimum we take $\sigma=1/\varepsilon$. The results as they would be computed are tabulated in Table II.

Here we want to point out that it is difficult for our filter formulation to give an analytic representation of ε because the SVD computation in our algorithm needs a fairly sophisticated iterative implementation. Instead we let $\varepsilon=0$ after the factors in Eq. (15) have been clearly formed, so that a purely numerical result has to be given in Table II for our algorithm (This computation was done in MATLAB on PC 486, and for convenience the printing result does not include the full fifteen decimal place output that was produced by the long precision computation on the machine). We believe that, however, such a handling does not destroy our illustration.

Note first that the conventional Kalman algorithm computed covariance has completely degenerated and if further measurements were processed the estimates would soon become completely inaccurate. As expected, our algorithm, together with Potter's and Bierman's, agrees with the rounded exact result. Remember that in this example R has been normalized, and measurement noise has been assumed to be uncorrelated. If not so additional transformations have to be done before Potter's and Bierman's algorithms can be used. In this situation, the advantage of our algorithm is clear. Furthermore, all computations have to be done in sequence in Potter's and Bierman's algorithms, and evidently this is not suitable for parallel computers, while our algorithm does not have this defect.

Finally, we want to point out that, except the conventional Kalman covariance, these covariances are not part of algorithm. They are included for comparison purposes only. In practical computations, only the updated covariance factors, such as

$$U^+ = \begin{bmatrix} .8567 & -.5257 \\ .5257 & .8507 \end{bmatrix},$$

and

$$D^+ = \begin{bmatrix} .3820 & 0 \\ 0 & 2.6178 \end{bmatrix},$$

in our algorithm, are needed.

IV. CONCLUSIONS

In this paper we developed a new Kalman filter algorithm which is based on the well-known singular value decomposition - one of the most important and fundamental working tools for the control/systems community, particularly in the area of linear systems. SVD is essential for the numerical stability, and is unsurpassed when it comes to producing a numerical solution to a nearly singular system [11][16], so it can be expected that the SVD-based new Kalman filter formulation will have the highest accuracy and stability characteristics in all existing filter algorithms.

Another advantage of the new Kalman formulation is the ability to handle correlated measurement noise without any additional transformations as might be used with the Potter and the Bierman algorithms. In many present day applications, measurements are generally designed to be uncorrelated; and this may be a large source of error. Our algorithm can efficiently overcome this defect.

Finally, the presented algorithm is also suitable for parallel computers because it is formulated in the form of matrix-matrix and matrix-vector operations. In a future paper we will discuss the parallel implementation of the new algorithm on a transputer network which will combine our recent research results on a shared memory multiprocessor [7].

REFERENCES

[1] A. Andrews, "A square root formulation of the Kalman covariance equations," AIAA J., vol. 6, pp. 1165-1166, 1968.

[2] R.H. Battin, Astronautical Guidance, New York: McGraw-Hill, 1964.

[3] J.F. Bellantoni and K.W. Dodge, "A square-root formulation of the Kalman-Schmidt filter," AIAA J., vol. 5, pp. 1309-1314, 1967.

[4] M. Berry and A. Sameh, "An overview of parallel algorithms for the singular value and symmetric eigenvalue problems," J. Comput. Appl. Math., vol. 27, pp. 191-213, 1989.

[5] G.J. Bierman, Factorization Methods for Discrete Sequential Estimation, New York: Academic Press, 1977.

[6] N.A. Carlson, "Fast triangular formulation of the square root filter," AIAA Journal, vol. 11, pp. 1259-1265, 1973.

[7] E.M. Daoudi and G. Libert, "Parallel Givens factorization on a shared memory multiprocessor," in H. Brukhart (ed.), Lecture Notes in Computer Sciences, Vol. 457, New York: Springer-Verlag, 1990, pp. 131-142.

[8] L. Fox, An Introduction to Numerical Linear Algebra, London: Clarendon Press, 1966.

[9] G.H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," SIAM J. Numer. Anal., vol. 2, pp. 205-224, 1965.

[10] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," Numer. Math., vol. 14, pp. 403-420, 1970.

[11] G.H. Golub and C.F. Van Loan, Matrix Computations, Second Edition, London: The Johns Hopkins Press Ltd., 1989.

[12] M.R. Hestenes, "Inversion of matrices by biorthogonalization and related results," J. Soc. Indust. Appl. Math., vol. 6, pp. 51-90, 1958.

[13] H.J. Hotop, "New Kalman filter algorithms based on orthogonal transformations for serial and vector computers," Parallel Computing, vol. 12, pp. 233-247, 1989.

[14] R.E. Kalman, "A new approach to linear filtering and prediction problems," Trans. ASME (J. Basic Eng.), vol. 82D, pp. 35-50, 1960.

[15] P.G. Kaminski, A.E. Bryson and S.F. Schmidt, "Discrete square root filtering: a survey of current techniques," IEEE Trans. Automat. Contr., vol. AC-16, pp. 727-735, 1971.

[16] V.C. Klema and A.J. Laub, "The singular value decomposition: its computation and some applications," IEEE Trans. Automat. Contr., vol. AC-25, pp. 165-176, 1980.

[17] E. Kogbetliantz, Solution of linear equations by diagonalization of coefficient matrices, Quart. Appl. Math., vol. 13, pp. 123-132, 1955.

[18] J.M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update and parameter estimation," Automatica, vol. 22, pp. 43-57, 1986.

[19] A.J. Laub, M.T. Heath, C.C. Paige and R.C. Ward, "Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms," IEEE Trans. Automat. Contr., vol. AC-32, pp. 115-121, 1987.

[20] F.T. Luk, "Computing the singular value decomposition on the Illiac IV," ACM Trans. Math. Software, vol. 6, pp. 524-539, 1980.

[21] P.S. Mayback, Stochastic Models, Estimation, and Control, Vol. 1. New York: Academic Process, 1979.

[22] J. Vandewalle and B. De Moor, "A variety of applications of singular value decomposition in identification and signal processing," in E.F. Deprettere (ed.), SVD and Signal Processing: Algorithms, Applications and Architectures, Amsterdam: Elsevier Science Publishers B.V., 1988, pp. 43-91.

[23] L. Wang, G. Libert and P. Manneback, "Experience with computing the singular value decomposition of a real unsymmetric matrix," Technical Report, Dept. of Computer Science, FPMs, 1992.

[24] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford: Clarendon Press, 1965.