Working Paper 93-14
Statistics and Econometrics Series 12
June 1993

Departamento de Estadística y Econometría
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (341) 624-9849

A PARALLEL KALMAN FILTER VIA THE SQUARE ROOT KALMAN FILTERING

Rosario Romera and Tomas Cipra*

Abstract

A parallel algorithm for Kalman filtering with contaminated observations is developed. The parallel implementation is based on the square root version of the Kalman filter (see [3]). This represents a great improvement over serial implementations reducing drastically computational costs for each state update.

Key Words
Parallel Robust Kalman Filter; Square Root Kalman Filter.

*Romera, Departamento de Estadística y Econometría, Universidad Carlos III de Madrid; Cipra, Department of Statistics, Charles University of Prague, Czech Republic.
Classification AMS: 62M20, 60G35, 93E11.

1. Introduction

Practical implementation of the Kalman filtering with a huge number (n) of state variables (e.g. physical/technological processes) might require a expensive operation time. The order of the operational cost for the solution of the filter equations is in general $0(n^3)$ for each state update, if the implementation is serial (sequential). Cost and size of computer components have declined so sharply that parallel computers have become feasible. As a result, there is a increasing interest about designing algorithms that exploit both the parallelism inherent in the problem and that available on the computer (see [1]).

From the computational point of view, this paper extent results in [3].

In this paper we describe a parallel Kalman filter algorithm based in the square root formulation of Kalman filter. The algorithm allows to reduce the operational cost to 0(n) for each state update, like alternative approaches (see [4] y [5]) and is able to be implemented in a wide variety of commercially available parallel computers. The paper is organized as follows: section 2.1 contains a brief description of the square root Kalman filter developed in [3]. In 2.2 the triangularization matrix procedure is described and in 2.3 the algorithm to update the state is explained. In section 3 we present how to implement the triangularization procedure for parallel computation. In section 4 we extent the algorithm in 2.3 to the case of the square root Kalman filter with contaminated observations. We include a special study for the $m=1$ case (scalar observations). Concluding remarks and computational complexity of the algorithm are mentioned in section 4. Comparative results for sequential and parallel implementation are included.

2. Square root Kalman filter

2.1. Formulation

Let us consider a discrete-time dynamic linear system given by

$$x_{t+1} = F_t x_t + w_t \tag{2.1}$$

$$y_t = H_t x_t + v_t \tag{2.2}$$

where x_t is the $(n \times 1)$ state vector, and y_t is an $(m \times 1)$ measurement vector (typically $m \leq n$).

We suposse that the process noise w_t and the measurement noise v_t are mutually independent and verify

$$E(w_t) = 0$$

$$E(v_t) = 0$$

$$E(w_t, w'_t) = \delta_{tt} Q_t$$

$$E(v_t, v'_t) = \delta_{tt} R_t.$$

The matrices F_t, H_t, Q_t and R_t are time varying of appropriate dimensions, and are supposed to be known at time t.

The classical Kalman filter gives the state estimate at time t ($\hat{x}_t^t$) as a linear combination of an state estimate at time t-1 ($\hat{x}_t^{t-1}$) and the observations data at time t-1(Y^{t-1}). The minimum variance state estimator $\hat{x}_t^t = E[x_t / Y^t]$ and its covariance matrix $P_t^t = E[(x_t - \hat{x}_t^t)(x_t - \hat{x}_t^t)' / Y^t]$ where $Y^t = \{y_0, y_1, \ldots, y_t\}$ are then given by the recursive algorithm

$$\hat{x}_t^t=\hat{x}_t^{t-1}+P_t^{t-1}H_t'(H_tP_t^{t-1}H_t'+R_t)^{-1}(y_t-H_t\hat{x}_t^{t-1}) \tag{2.3}$$

$$P_t^t=P_t^{t-1}-P_t^{t-1}H_t'(H_tP_t^{t-1}H_t'+R_t)^{-1}H_tP_t^{t-1} \tag{2.4}$$

Then, the predicted state is given by the recursive formula

$$\hat{x}_{t+1}^t=F_t\,\hat{x}_t^t \tag{2.5}$$

$$P_{t+1}^t=F_tP_t^tF_t'+Q_t \tag{2.6}$$

An analogous expression for the predicted state estimate is given by the recursive relation

$$\hat{x}_{t+1}^t=F_t\hat{x}_t^{t-1}+K_t(y_t-H_t\hat{x}_t^{t-1}) \tag{2.7}$$

$$P_{t+1}^t=F_tP_t^{t-1}F_t'+Q_t-K_tR_{et}K_t' \tag{2.8}$$

where $K_t=F_tP_t^{t-1}H_t'R_{et}^{-1}$ is the Kalman gain matrix with dimension (n×m), and $R_{et}=H_tP_t^{t-1}H_t'+R_t$ is the innovations covariance matrix with dimension (m×m). These expressions are obtained by substitution of (2.3) and (2.4) into (2.5) and (2.6).

The square root Kalman filter formulation, which exploits the factorization property for a positive semidefinite matrix can be found in [4].

Then for any positive semidefinite matrix, specifically for P_t and R_{et}, we obtain the appropriate factorization by means of a unit lower triangular matrix , with units on the main diagonal, let say L_{pt} and L_{et}, and a diagonal matrix, denoted by D_{pt} and D_{et}, i.e.

$$P_t^{t-1}=L_{pt}D_{pt}L_{pt}' \tag{2.9}$$

$$R_{et}=L_{et}D_{et}L_{et}' \tag{2.10}$$

Then, the Kalman gain matrix K_t and the covariance update P^t_{t+1} are then obtained from the triangularization of a particular matrix.

2.2. Triangularization procedure

The main idea is to transform a pair of matrices (A,D) such that

$$A = \begin{bmatrix} I & HL_{pt} & 0 \\ 0 & FL_{pt} & I \end{bmatrix} \quad , \; M \times N \text{ dimensional}$$

where M = m + n and N = m + n + n, and

$$D = \begin{bmatrix} R_t & 0 & 0 \\ 0 & D_{pt} & 0 \\ 0 & 0 & Q_t \end{bmatrix} \quad , \; N \times N \text{ dimensional.}$$

into another pair (A^*, D^*) such that A^* is a unit lower triangular (M×N) dimensional matrix with units on the main diagonal

$$A^* = \begin{bmatrix} L_{et} & 0 & 0 \\ KL_{et} & L_{p\,t+1} & 0 \end{bmatrix}$$

and D^* is a (N×N) diagonal matrix

$$D^* = \begin{bmatrix} D_{et} & 0 & 0 \\ 0 & D_{p\,t+1} & 0 \\ 0 & 0 & D_a \end{bmatrix}$$

D_a is an arbitrary diagonal n×n dimensional matrix. Then, it holds that

$$A\,D\,A' = A^*\,D^*\,A^{*\prime} \tag{2.11}$$

See Appendix for the proof.

2.3. Algorithm

The basis of the square root algorithm implementation is to maintain the triangularized form during each update computation. The scheme of the algorithm for one step is as follows

Time t

Input:	F_t, H_t, R_t, Q_t
$t \geq 0$	y_t
	$\hat{x}_t^{t-1}$
	L_{pt}, D_{pt}
Output:	$\hat{x}_{t+1}^{t}$
$t > 0$	$L_{p(t+1)}$, $D_{p(t+1)}$, and P_{t+1}^{t} from (2.9)

We calculate these prediction output values by means of:

- Construct A and D matrices from input matrices F_t, H_t, R_t, D_t, L_{pt} and D_{pt}
- Obtain A^* and D^* matrices verifying (2.11)
- Compute $\hat{x}_{t+1}^{t}$ (2.7) following:
    - compute $\alpha_t = y_t - H_t \hat{x}_t^{t-1}$
    - solve $L_{et} z_t = \alpha_t$ for z_t
    - compute $K_t L_{et} z_t = \beta_t$
    - finally compute $\hat{x}_{t+1}^{t} = F_t \hat{x}_t^{t-1} + \beta_t$
- Give $\hat{x}_{t+1}^{t}$ and $L_{p(t+1)}$, $D_{p(t+1)}$ as input values for next step.

3. Triangularization procedure by parallel computation

In this section we describe in terms of parallel instructions how to obtain A^* and D^* matrices from A and D matrices. The key is the triangularization of matrix A. Basic ideas are in Palis (1989) (see [5]).

Examples of primary parallel instructions (included as primitive instruction for example in the connection Machine parallel instruction set) are the scan operations. A scan operation takes a binary associative operator $\oplus$ and an ordered set of elements $[e_1\ e_2\ e_3,\ldots]$ and computes the ordered set $[e_1,\ e_1 \oplus e_2,\ e_1 \oplus e_2 \oplus e_3,\ \ldots]$. Example of scan operation is the scan-with-add operator, where $\oplus$ is the addition. We will make extensive use of this parallel instruction.

A clear way to obtain the triangularization is to zero out the rows of A one at time starting with the first row. The idea behind the parallel algorithm is to achieve zeros in each row of A using a constant number of scan operations. Thus, the total amount of parallel operations is O(M).

For $1 \leq K \leq M$ let denote by $A^{(k)}$ and $D^{(k)}$ the corresponding matrices after zeroing out the k-th row of A. We denote the original A and D matrices by $A^{(0)}$ and $D^{(0)}$.

Let introduce $S = ADA' = A^*D^*A^{*\prime}$. We define for $k \leq i \leq M$ and $k \leq j \leq N$

$$s_{ij}^{(k-1)} = \sum_{l=k}^{j} a_{kl}^{(k-1)} a_{il}^{(k-1)} d_{ll}^{(k-1)} \tag{3.1}$$

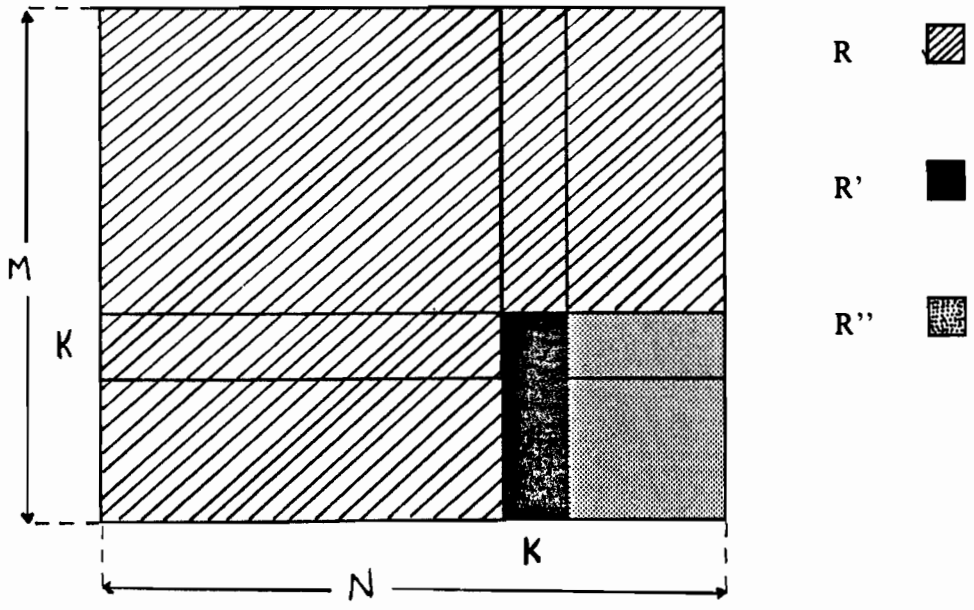
More explicitly, the elements on the diagonal matrix $D^{(k)}$ take the expression

$$d_{ij}^{(k)} = \begin{cases} d_{ij}^{(k-1)} & 1 \le j < k \\ s_{jN}^{(k-1)} & j = k \\ \dfrac{s_{k(j-1)}^{(k-1)}}{s_{kj}^{(k-1)}} d_{ij}^{(k-1)} & k < j \le N \end{cases} \qquad (3.2)$$

Analogous form for the $A^{(k)}$ matrix

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)} & \textit{for elements in } R \equiv \{1 \le i \le k-1, 1 \le j \le N\} \vee \{1 \le i \le M, 1 \le j \le k-1\} \\ \dfrac{s_{iN}^{(k-1)}}{s_{kN}^{(k-1)}} & \textit{for elements in } R' \equiv \{k \le i \le M, j = k\} \\ a_{ij}^{(k-1)} - \dfrac{s_{i(j-1)}^{(k-1)}}{s_{k(j-1)}^{(k-1)}} a_{kj}^{(k-1)} & \textit{for elements in } R'' \equiv \{k \le i \le M, k \le j \le N\} \end{cases} \qquad (3.3)$$

Graphically, R, R' and R'' represent the following positions for a M × N matrix



The parallel procedure uses a M × N array of processors. Let denote them by P(i,j). At the first step, k=0, P(i,j) holds $a_{ij}^{(0)}$ and $d_{ij}^{(0)}$ values, with $1 \le i \le M$ and $1 \le j \le N$. At the start of each k iteration $1 \le k \le M$, P(i,j) contains $a_{ij}^{(k-1)}$, $d_{ij}^{(k-1)}$ and proceeds

as follow

1.- P(i,j) holds $a_{kj}^{(k-1)}$, $k \leq i \leq M, k \leq j \leq N$

2.- P(i,j) operates $a_{kj}^{(k-1)}\, a_{ij}^{(k-1)}\, d_{jj}^{(k-1)}$, $k \leq i \leq M, k \leq j \leq N$

3.- using scan-with-add operation for $k \leq i \leq M$ we get expression (3.1)

$$P(i,k) \text{ holds } \sum_{j=k}^{k} a_{kj}^{k-1} a_{ij}^{k-1} d_{jj}^{k-1} = s_{ik}^{k-1}$$

.

.

.

$$P(i,N) \text{ holds } \sum_{j=k}^{N} a_{kj}^{(k-1)} a_{ij}^{(k-1)} d_{jj}^{(k-1)} = s_{iN}^{(k-1)}$$

4.- P(i,j) then contains $a_{ij}^{(k-1)}$, $s_{i(j-1)}^{(k-1)}$ $k \leq i \leq M, k \leq j \leq N$

5.- finally we compute $a_{ij}^{(k)}$, $d_{jj}^{(k)}$ $1 \leq i \leq M$, $1 \leq j \leq N$, in each P(i,j) following (3.2) and (3.3).

After $k=M$ iterations, P(i,j) contains the elements $a_{ij}^{*} = a_{ij}^{(M)}$ and $d_{jj}^{*} = d_{jj}^{(M)}$ $1 \leq i \leq M$, $1 \leq j \leq N$.

4. Square root kalman filter with contaminated observations

4.1. Formulation

In this section we obtain the parallel Kalman filter for the case of contaminated observations.

Let consider a dynamic linear system of the form

$$x_{t+1}=F_t\, x_t+\omega_t \qquad \omega_t \sim iid\ N(0,Q_t)$$
$$y_t=H_t\, x_t+v_t \qquad v_t \sim iid\ \epsilon\text{-}contaminated\ N(0,R_t)$$

where residuals $\{w_t\}$ and $\{v_t\}$ are mutually independent.

For the contaminated case, predictive values given by (2.5) and (2.6) can be obtained after some robustification procedure of the Kalman filter. For instance, an approach based on the M-estimation principle developed by the authors is given in [2]. It seems to be very efficient from the computational point of view, even in sequential implementation.

The state estimates formulas in this work [2] are given by

$$\hat{x}_t^t=\hat{x}_t^{t-1}+P_t^{t-1}H_t'[H_tP_t^{t-1}H_t'+R_t^{1/2}W_t^{-1}R_t^{1/2}]^{-1}(y_t-H\hat{x}_t^{t-1})$$
$$P_t^t=P_t^{t-1}-P_t^{t-1}H_t'[H_tP_t^{t-1}H_t'+R_t^{1/2}W_t^{-1}R_t^{1/2}]^{-1}H_tP_t^{t-1}$$

where $R_t^{1/2}$ denotes the square root matrix of R_t, and W_t is the $m\times m$ diagonal matrix with elements

$$\omega_{jt}=\frac{\psi_j(c_{jt}-b_{jt}\,\hat{x}_t^{t-1})}{c_{jt}-b_{jt}\,\hat{x}_t^{t-1}} \qquad 1\le j\le m$$

$\Psi_1, \ldots, \Psi_m$ are the robustifying psi-functions, and c_{jt} and b_{jt} are the elements of vectors

$$R_t^{-1/2}H_t=\begin{bmatrix} b_{1t} \\ . \\ . \\ . \\ b_{mt} \end{bmatrix} \quad and \quad R_t^{-1/2}y_t=\begin{bmatrix} c_{1t} \\ . \\ . \\ . \\ c_{mt} \end{bmatrix}$$

($R_t^{-1/2}$ is the inverse matrix of $R^{1/2}$)

So, the predictive formulas (2.7) and (2.8) can now be rewritten as

$$\hat{x}_{t+1}^{t}=F_t\hat{x}_t^{t-1}+K_t(y_t-H_t\hat{x}_t^{t-1})$$

$$P_{t+1}^{t}=F_tP_t^{t-1}F_t^{/}+Q_t-K_tR_{et}K_t^{/}$$

where

$$K_t=F_tP_t^{t-1}H_t^{/}R_{et}^{-1}$$

$$R_{et}=H_tP_t^{t-1}H_t^{/}+R_t^{1/2}W_t^{-1}R_t^{1/2}$$

and matrices P_t^{t-1} and R_{et} follow (2.9) and (2.10) as appropriate factorization expressions.

4.2. Algorithm

We proceed as in Section 2.3, except the construction of D matrix.

Time t	Input:	F_t, H_t, R_t, Q_t
	$t \geq 0$	y_t
		$\hat{x}_t^{t-1}$
		L_{pt}, D_{pt}
	Output:	$\hat{x}_{t+1}^{t}$
	$t > 0$	$L_{p(t+1)}$, $D_{p(t+1)}$ and then P_{t+1}^{t}

Prediction values are obtained as follows:

- Construct A ((m+n) × (m+2n)) and D ((m+2n) × (m+2n)) matrices

$$A=\begin{bmatrix} I & H_t L_{pt} & 0 \\ 0 & F_t L_{pt} & I \end{bmatrix}, \quad D=\begin{bmatrix} R_t^{1/2} W_t^{-1} R_t^{1/2} & 0 & 0 \\ 0 & D_{pt} & 0 \\ 0 & 0 & Q_t \end{bmatrix}$$

- Obtain A^* and D^* following Section 2.2, such that

  $ADA' = A^*D^*A^{*\prime}$

- Compute $\hat{x}_t^{t+1}=F_t\hat{x}_t^{t+1}+(K_tL_{et})L_{et}^{-1}(y_t-H_t\hat{x}_t^{t-1})$

  where K_tL_{et} and L_{et} are obtained from A^*.

- Give $\hat{x}_{t+1}^{t}$, $L_{p(t+1)}$ and $D_{p(t+1)}$ as input values for next step.

We present the scalar observations as a special case:

Let consider (2.1) and (2.2) with m=1. The predictive values for time t constructed at time t-1 following [2], are given by

$$\hat{x}_{t+1}^{t}=F_t\hat{x}_t^{t-1}+K_tr_{et}r_t^{-1/2}\psi_H(r_{et}^{-1}r_t^{1/2}(y_t-h_t\hat{x}_t^{t-1}))$$
$$P_{t+1}^{t}=F_tP_t^{t-1}F_t'+Q_t-K_tr_{et}K_t'$$

where

$$K_t=F_tP_t^{t-1}h_t'r_{et}^{-1}$$
$$r_{et}=h_tP_t^{t-1}h_t'+r_t$$

In this case matrices $A((1+n)\times(1+2n))$, $D((1+2n)\times(1+2n))$, $A^*((1+n)\times(1+2n))$ and $D^*((1+2n)\times(1+2n))$ take the form

$$A=\begin{bmatrix} 1 & h_t L_{pt} & 0 \\ 0 & F_t L_{pt} & I \end{bmatrix}, \qquad D=\begin{bmatrix} r_t & 0 & 0 \\ 0 & D_{pt} & 0 \\ 0 & 0 & Q_t \end{bmatrix}$$

$$A^*=\begin{bmatrix} 1 & 0 & 0 \\ K_t & L_{pt+1} & 0 \end{bmatrix}, \qquad D^*=\begin{bmatrix} r_{et} & 0 & 0 \\ D_{pt+1} & 0 & 0 \\ 0 & 0 & D_a \end{bmatrix}$$

So we obtain K_t, r_{et}, $L_{p(t+1)}$ and $D_{p(t+1)}$ from A^* and D^* and the predicted values for t+1 at time t:

$$\hat{x}_{t+1}^t = F_t \hat{x}_t^{t-1} + K_t r_{et} r_t^{-1/2} \psi_H(r_{et}^{-1} r_t^{1/2} (y_t - h_t \hat{x}_t^{t-1}))$$

$$P_{t+1}^t = L_{pt+1} D_{pt+1} L_{pt+1}$$

Computational results

We present early comparative results of the execution time for sequential and parallel methods of the algorithm described in section 4 (Kalman filter with contaminated observations).

Time measures are presented for input/output operations (Figure 1), algorithmic calculus (Figure 2) and finally, total time (Figure 3) adds both.

For each Figure and Table:

dimension represents n=m values (dimensions of state and observation vectors), and

time is execution time in 10^{-3} seconds.

Software has been coded in C language, using Turbo C compiler version 2.0, and 486 processor at 33 MHz as hardware resource.

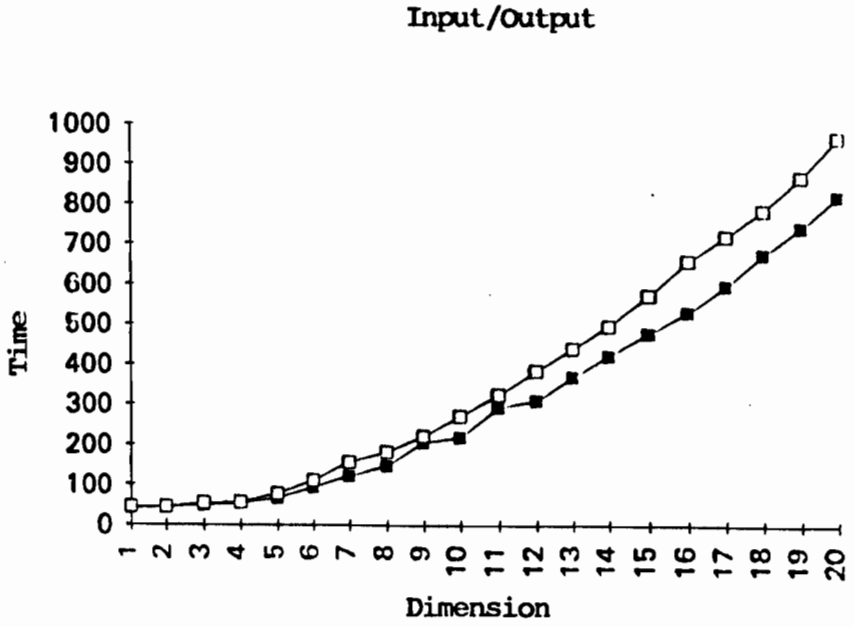


FIGURE 1

DIMENSION	SEQUENTIAL (■)	PARALLEL (□)
1	40	44
2	43	44
3	48	53
4	53	55
5	66	77
6	92	110
7	121	155
8	147	183
9	208	223
10	220	274
11	296	327
12	311	385
13	370	440
14	422	495
15	478	572
16	532	659
17	598	722
18	677	787
19	744	872
20	824	972

TABLE 1

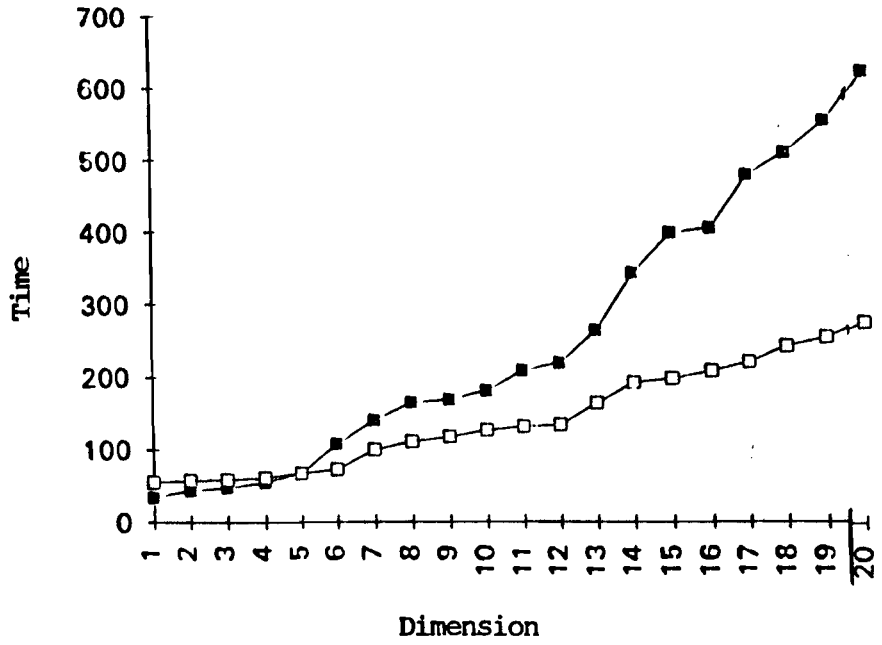


FIGURE 2

DIMENSION	SEQUENTIAL (■)	PARALLEL (□)
1	34	55
2	43	57
3	48	58
4	55	61
5	67	67
6	108	72
7	141	100
8	166	111
9	170	118
10	183	128
11	211	133
12	221	135
13	266	166
14	344	194
15	399	199
16	406	209
17	480	221
18	511	243
19	556	255
20	628	278

TABLE 2

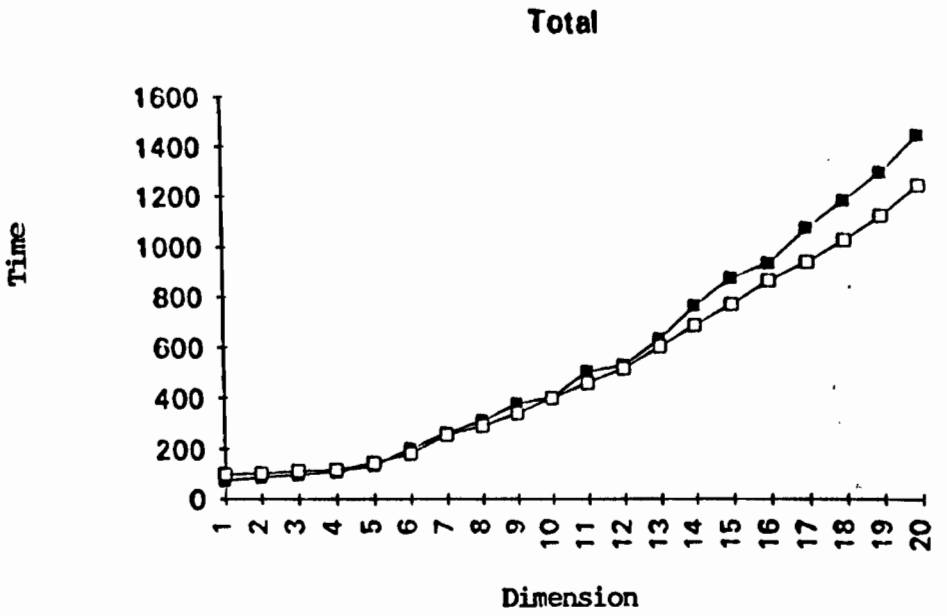


FIGURE 3

DIMENSION	SEQUENTIAL (■)	PARALLEL (□)	[(SEQ-PAR)/SEQ] x 100
1	74	99	--
2	86	101	--
3	96	111	--
4	108	116	--
5	133	143	--
6	200	182	9
7	262	255	2,6
8	313	294	6
9	378	341	9,7
10	403	402	--
11	507	460	9,2
12	532	520	2,2
13	636	606	4,7
14	766	689	10,0
15	877	771	12
16	937	868	7,3
17	1078	943	12,5
18	1188	1030	13,2
19	1300	1127	13,3
20	1452	1250	13,9

TABLE 3

Appendix

Verification of the relation (2.11)

$$ADA' = \begin{bmatrix} I & HL_{pt} & 0 \\ 0 & FL_{pt} & I \end{bmatrix} \begin{bmatrix} R_t & 0 & 0 \\ 0 & D_{pt} & 0 \\ 0 & 0 & Q_t \end{bmatrix} \begin{bmatrix} I & 0 \\ L'_{pt}H'_t & L_{pt}F'_t \\ 0 & I \end{bmatrix} =$$

$$= \begin{bmatrix} R_t + HL_{pt}D_{pt}L'_{pt}H'_t & HL_{pt}D_tL'_{pt}F'_t \\ FL_{pt}D_{pt}L'_{pt}H'_t & FL_{pt}D_{pt}L'_{pt}F'_t + Q_t \end{bmatrix} =$$

$$= \begin{bmatrix} R_t + H_tP_t^{t-1}H'_t & H_tP_t^{t-1}F'_t \\ F_tP_t^{t-1}H'_t & F_tP_t^{t-1}F'_t + Q_t \end{bmatrix} = \begin{bmatrix} R_{et} & R_{et}K'_t \\ K_tR_{et} & P_{t+1}^t + K_tR_{et}K'_t \end{bmatrix}$$

on the other hand

$$A^*D^*A^{*\prime} = \begin{bmatrix} L_{et} & 0 & 0 \\ K_tL_{et} & L_{pt+1} & 0 \end{bmatrix} \begin{bmatrix} D_{et} & 0 & 0 \\ 0 & D_{pt+1} & 0 \\ 0 & 0 & D_a \end{bmatrix} \begin{bmatrix} L'_{et} & K'_tL'_{et} \\ 0 & L'_{pt+1} \\ 0 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} L_{et}D_{et}L'_{et} & L_{et}D_{et}L'_{et}K'_t \\ K_tL_{et}D_{et}L'_{et} & K_tL_{et}D_{et}L'_{et}K'_t + L_{pt+1}D_{pt+1}L'_{pt+1} \end{bmatrix} =$$

$$\begin{bmatrix} R_{et} & R_{et}K'_t \\ K_tR_{et} & P_{t+1}^t + K_tR_{et}K'_t \end{bmatrix}. \quad \blacksquare$$

REFERENCES

[1] AKL, S.G.: The Design and Analysis of Parallel Algorithms, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[2] Cipra, T., Romera, R.: Robust Kalman filter and its application in time series analysis, Kybernetika 27 (1991), n° 6, 481-494.

[3] Cipra, T., Romera, R., Rubio, A.: Square root Kalman filter with contaminated observations, Working Paper 92-09, March 1992, Universidad Carlos III de Madrid.

[4] Itzkowitz, H.R., Baheti, R.S.: Demonstration of Square Root Kalman Filter on Warp Parallel Computer, Proceedings of the 1989 American Control Conference.

[5] Palis, M.A., Krecker, D.K.: Parallel Kalman Filtering on the Connection Machine, GE International Technical Report, Advanced Technology Laboratories, GE Aerospace, Moorestown, New Jersey, 1989.