Systolic VLSI for Kalman Filters

H.-G. Yeh
California State University, Long Beach

J. J. Chang
Flight Computer Systems and Technology

A novel two-dimensional parallel computing method for real-time Kalman filtering is presented. The mathematical formulation of a Kalman filter algorithm is rearranged to be the type of Faddeev algorithm for generalizing signal processing. The data flow mapping from the Faddeev algorithm to a two-dimensional concurrent computing structure is developed. The architecture of the resulting processor cells is regular, simple, expandable, and therefore naturally suitable for VLSI chip implementation. The computing methodology and the two-dimensional systolic arrays are useful for Kalman filter applications as well as other matrix/vector based algebraic computations.

I. Introduction

For more than 20 years, the Kalman filter has been applied extensively in many signal processing applications, including target tracking, adaptive controls, radar-signal processing, and failure-proof systems (Refs. 1 to 4). The applicability of the Kalman filter to real-time signal processing problems is in general limited by the relatively complex mathematical operations necessary in computing the Kalman filtering algorithm (Refs. 5 and 6). However, with the rapid development of VLSI integrated circuits (Refs. 7 to 9), it is quickly becoming technologically feasible to implement Kalman filters in real time.

The processing of a Kalman filter requires matrix/vector operations such as multiplication, addition, subtraction, and inversion. Among these, matrix inversion is the most difficult to implement in terms of speed and accuracy. The Faddeev algorithm (Refs. 10 and 11) has been suggested as a universal algorithm for various matrix manipulations due to the fact that it is easily systematized for matrix calculations and maps easily into a concurrent systolic array architecture. It is natural to arrange Kalman filter algorithms into a form of Faddeev algorithm to maximize the capability of hardware implementation of systolic arrays. First of all, this proposed algorithm avoids the direct matrix inverse computation as the usual back substitution in the Kalman Filter implementation, but obtains the values of desired results at the end of the forward course of computation, resulting in a considerable saving in added processing and storage. Secondly, since the Gaussian elimination procedure is applied through the computation, numerical stability is obtained. Thirdly, the parallel, modular computer architecture which consists of systolic processors provides simultaneous high throughput and a capability for a wide variety of linear algebraic operations.

In the following sections, we describe, in order, Kalman filters, the Faddeev algorithm, the implementation, a two-

dimensional systolic array architecture, and a numerical example.

II. Kalman Filter

Kalman filters have been shown to be the optimal linear estimator in the least square sense for estimating dynamic system states in linear systems. The Kalman filter updates state estimation based on prior estimates and observed measurements. It consists of the model of the dynamic process which performs the function of prediction and a feedback correction scheme. The measurements can be processed as they occur, and there is no need to store any measurement data. However, all the associated matrices which describe the system dynamic, measurement system, and noise is assumed to be known. The conventional discrete time-varying Kalman filtering process involves the propagation of state estimates and error covariance matrices from time sample to next time sample. Discussions and applications on Kalman filters can be found widely in literature (Refs. 1 to 6).

The following equations define a general dynamic system and a measurement system.

$$\mathbf{x}(k+1) = \phi(k)\,\mathbf{x}(k) + \mathbf{w}(k) \tag{1}$$

$$\mathbf{z}(k) = \mathbf{H}(k)\,\mathbf{x}(k) + \mathbf{v}(k) \tag{2}$$

In Eq. (1), $\phi(k)$ is a $n \times n$ matrix called the state transition matrix which describes the plant, $\mathbf{x}(k)$ is the state vector with n-dimension, and $\mathbf{w}(k)$ is a n-vector called the disturbance. In Eq. (2), $\mathbf{z}(k)$ is a m-vector termed the measurement vector; $\mathbf{v}(k)$ is also a m-vector called the measurement noise vector, and $\mathbf{H}(k)$ is a $m \times n$ matrix called the measurement matrix. The disturbance $\mathbf{w}(k)$ and noise $\mathbf{v}(k)$ are assumed to be zero-mean Gaussian white noise sequences with covariance matrices $\mathbf{Q}(k)$ and $\mathbf{R}(k)$, respectively. Furthermore, sequences $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are assumed to be statistically independent. The Kalman filter is described by the following equations under assumptions that matrices $\phi(k)$, $\mathbf{H}(k)$, $\mathbf{R}(k)$, and $\mathbf{Q}(k)$ are known.

$$\hat{\mathbf{x}}(k/k-1) = \phi(k-1)\,\hat{\mathbf{x}}(k-1/k-1) \tag{3}$$

$$\mathbf{P}(k/k-1) = \phi(k-1)\,\mathbf{P}(k-1/k-1)\,\phi^T(k-1) + \mathbf{Q}(k-1) \tag{4}$$

$$\mathbf{P}^{-1}(k/k) = \mathbf{P}^{-1}(k/k-1) + \mathbf{H}^T(k)\,\mathbf{R}^{-1}(k)\,\mathbf{H}(k) \tag{5}$$

$$\mathbf{K}(k) = \mathbf{P}(k/k)\,\mathbf{H}^T(k)\,\mathbf{R}^{-1}(k) \tag{6}$$

$$\Delta\mathbf{z}(k) = \mathbf{z}(k) - \mathbf{H}(k)\,\hat{\mathbf{x}}(k/k-1) \tag{7}$$

$$\hat{\mathbf{x}}(k/k) = \hat{\mathbf{x}}(k/k-1) + \mathbf{K}(k)\,\Delta\mathbf{z}(k) \tag{8}$$

$$k = 1, 2, \cdots$$

Initial conditions are given as follows:

$$\mathbf{x}(0/0) = 0 \tag{9}$$

$$\mathbf{P}(0/0) = \mathbf{P}(0) \quad \text{or} \quad \mathbf{P}^{-1}(0/0) = \mathbf{P}^{-1}(0) \tag{10}$$

Note that matrices $\mathbf{P}(k/k)$ and $\mathbf{P}(k+1/k)$ are commonly called error covariance matrices. The vector $\hat{\mathbf{x}}(k/k)$ represents the optimal estimate of $\mathbf{x}(k)$ based on the measurement sequence $\{\mathbf{z}(1), \mathbf{z}(2), \mathbf{z}(3), \cdots, \mathbf{z}(k)\}$. Equations (3) and (4) are referred to as time updates and Eqs. (5), (6), and (8) are referred to as measurement updates. These filter equations are computed in order as listed from Eqs. (3) to (8).

III. Faddeev Algorithm

Consider a linear equation:

$$\mathbf{AX} = \mathbf{B} \tag{11}$$

Suppose that it is desired to find $\mathbf{CX} + \mathbf{D}$, a linear combination of $\mathbf{X}$ without first finding $\mathbf{X}$, which is an unknown. This can be represented in matrix form as:

A	**B**
-C	**D**

(12)

where **A**, **B**, **C**, and **D** are matrices or vectors. The dimension of the matrices will be discussed later. By adding suitable combinations of **A** and **B** to **-C** and **D**, the term "**-C + WA**" appears in the lower left hand quadrant and "**D + WB**" in the bottom right hand quadrant. It shows in matrix form as:

A	**B**
-C + WA	**D + WB**

(13)

If **W** specifies the appropriate linear combination such that the lower left hand side of Eq. (13) is zero, then **CX + D** will appear on the lower right hand side.

That is,

$$W = CA^{-1} \tag{14}$$

$$D + WB = D + CA^{-1}B$$

$$= CX + D \tag{15}$$

Representing the above equation in matrix form, we obtain:

A	B
0	CX + D

(16)

Note that **CX + D** is the desired result and appears in the bottom right hand quadrant after the process to annul the bottom left hand quadrant.

It is clear from the above illustration that given matrices **A**, **B**, **C** and **D**, $CA^{-1}B + D$ can be computed. Consequently, by proper selection of matrices **A**, **B**, **C**, and **D**, the capability of the Faddeev algorithm can be fully utilized. The various possible matrix manipulations are shown in Fig. 1.

The simplicity of the algorithm is due to the absence of a necessity to actually identify the multipliers of the rows of **A** and the elements of **B**; it is only necessary to "annul the last row." This can be done by triangularization, a numerically stable procedure, combined with an equally stable Gaussian elimination procedure (Refs. 12 and 13).

One of the more important features of this algorithm is that it avoids the usual back substitution or solution to the triangular linear system and obtains the values of the unknowns directly at the end of the forward course of the computation, resulting in a considerable savings in added processing and storage.

IV. Implementation

Kalman filter algorithms are adjusted to be the type of Faddeev algorithm in this section (Ref. 14). Computations are cyclically propagated through the following ordered set of passes. Note that the new data could be shifted into the array from the top, row by row as the calculation proceeds, so that there would be no delay in starting the next matrix computation.

1st pass:

I	$\hat{x}(k-1/k-1)$	
$-\phi(k-1)$	0	$\hat{x}(k/k-1)$

2nd pass:

$P^{-1}(k-1/k-1)$	$\phi^T(k-1)$	
$-\phi(k-1)$	$Q(k-1)$	$P(k/k-1)$

3rd pass:

R(k)	I	
$-H^T(k)$	0	$H^T(k)R^{-1}(k)$

4th pass:

P(k/k - 1)	I	
-I	0	$P^{-1}(k/k-1)$

5th pass:

I	H(k)	
$-H^T(k)R^{-1}(k)$	$P^{-1}(k/k-1)$	$P^{-1}(k/k)$

6th pass:

$P^{-1}(k/k)$	$H^T(k)R^{-1}(k)$	
-I	0	K(k)

7th pass:

I	$\hat{x}(k/k-1)$	
H(k)	z(k)	$\Delta z(k)$

8th pass:

I	$\Delta z(k)$	
-K(k)	$\hat{x}(k/k-1)$	$\hat{x}(k/k)$

Note that the results (lower right quadrant) of each pass in general, needs to be stored and is used in later passes as new entries.

V. Architecture

A general architecture is proposed to process all eight passes as mentioned in the previous section. To fit the type of Faddeev algorithm, the number of columns of the matrix (lower left quadrant) should be less than or equal to that of the matrix (upper left quadrant). Fortunately, all eight passes meet this requirement so that the matrix (lower left quadrant) will be annulled row by row. This can be done by ordinary Gaussian elimination. However, Gaussian elimination in general requires pivoting, and the pivoting strategy is not suited to a systolic array since it may require global communication for the pivot section. The neighbor pivoting technique introduces a zero to a row by subtracting a multiple of an adjacent row from it, interchanging the two rows when necessary to prevent the multiple from exceeding unity (Ref. 13).

This process is shown in the modified Faddeev algorithm (a two-step procedure):

A	B	step 1	T	MB
-C	D		-C	D

T	MB	step 2	T	MB
-C	D		-C + WT	D + WMB

where **T** is an upper triangular, and **M** is a nonsingular (triangularization) matrix. Since $\mathbf{W} = \mathbf{CT}^{-1}$, the final result is $\mathbf{G} = \mathbf{D} + \mathbf{WMB} = \mathbf{D} + \mathbf{CA}^{-1}\mathbf{B}$. In order to triangularize matrix **A** and to annul matrix **C**, it is necessary to have a two-step procedure. First, **A** is triangularized by the neighbor pivoting process (simultaneously applied to **B**); second, **C** is annulled by Gaussian elimination using the diagonal elements of **T** as pivot elements. The square processor arrangement for both triangularization and annulling steps is shown in Fig. 2 (Refs. 10 and 11). However, to compute all eight passes, the size of the processor is $2n$ cells (row) by $2n$ cells (column).

It is desirable to have a fixed size processor to handle all eight passes. However, the size of the entry matrix/vector varies from pass to pass. By padding zeros in appropriate places, the $2n$ cells (row) by $2n$ cells (column) become the proper size for implementing Kalman filters. The scheme of zero padding is illustrated in Fig. 3. By using the fixed size $2n \times 2n$ processor arrays, the estimate $\mathbf{x(k/k)}$ can be updated in each $16n$ time unit (assuming that it takes one time unit to operate data in a cell.)

VI. Numerical Example

An example of air-traffic-control radar is chosen for numerical simulation. In this example, one has range, range rate, bearing, and bearing rate as four components (in order) of the state vector $\mathbf{x(k)}$. The maneuver noise, which impacts on the change in range rate and bearing rate, is assumed to be zero-mean white, and the range rate and bearing rate are uncorrelated and with variances of 330 and 1.3×10^{-8}, respectively. The radar sensors are assumed to provide noisy measurements of the range and bearing. The noise of the radar sensors are assumed to be zero-mean white, and the range rate and bearing rate are uncorrelated and with variances of 10^6 and 2.89×10^{-4}, respectively. The values of the matrices and initial values are listed as follows:

$$\phi(\mathbf{k}) = \begin{bmatrix} 1 & 15 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H(k)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{Q(k)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 330 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.3 \times 10^{-8} \end{bmatrix}$$

$$\mathbf{R(k)} = \begin{bmatrix} 10^6 & 0 \\ 0 & 2.89 \times 10^{-4} \end{bmatrix}$$

$$\mathbf{P^{-1}\,(0/0)} = \begin{bmatrix} 0.2 \times 10^{-5} & -0.149 \times 10^{-4} & 0 & 0 \\ -0.149 \times 10^{-4} & 0.222 \times 10^{-3} & 0 & 0 \\ 0 & 0 & 0.662 \times 10^{4} & -0.483 \times 10^{5} \\ 0 & 0 & -0.483 \times 10^{5} & 0.738 \times 10^{6} \end{bmatrix}$$

$$\hat{\mathbf{x}}\mathbf{(0/0)} = \begin{bmatrix} 0.200 \times 10^{4} \\ 0 \\ 0.500 \\ 0 \end{bmatrix}$$

This example was simulated on a VAX 780. The Kalman gain $\mathbf{K}_{11}\,\mathbf{(k)}$, the range error covariance $\mathbf{P}_{11}\,\mathbf{(k/k-1)}$, and the bearing error covariance $\mathbf{P}_{33}\,\mathbf{(k/k-1)}$ were plotted against time index k and shown in Fig. 4. These plots show the numerical stability of the implementation of the Kalman filter on the concurrent systolic arrays.

References

1. Gebb, A., *Applied Optimal Estimation*, The M.I.T. Press, Cambridge, Mass., 1974.
2. Meditch, J. S., *Stochastic Optimal Linear Estimation and Control*, McGraw-Hill, New York, 1969.
3. Yeh, H. G., "A Design Method for Failure-Proof Systems," *Proceedings of the 1983 American Control Conference*, San Francisco, Calif., June 1983, pp. 1219-1223.
4. Yeh, H. G., *Techniques for the Detection, Estimation, Distinction, and Compensation of Failures in Linear Systems*, Ph.D. Dissertation, U.C. Irvine, 1982.
5. Sorenson, H. W., *Parameter Estimation*, Marcel Dekker, New York, 1980.
6. Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Rev. ed., Halsted Press, Div. of John Wiley & Sons, Waltham, Mass., 1975.
7. Yeh, C. S., Reed, I. S., Chang, J. J., and Truong, T. K., "VLSI Design of Number Theoretic Transforms for a Fast Convolution," *Proceedings of the 1983 IEEE International Conference on Computer Design: VLSI in Computers*, Port Chester, New York, Oct. 31 - Nov. 3, 1983, pp. 202-203.
8. Chang, J. J., Truong, T. K., Shao, H. M., Reed, I. S., and Hsu, I. S. "The VLSI Design of a Single Chip for the Multiplication of Integers Modulo a Fermat Number," *IEEE Trans. on ASSP*, Vol. ASSP-33, No. 6, Dec. 1985, pp. 1599-1602.
9. Kung, S. Y., Whitehouse, H. J., and Kailath, T., *VLSI and Modern Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985, pp. 375-338.
10. Nash, J. G., and Hansen, S., "Modified Faddeev Algorithm for Matrix Manipulation," *Proceedings of the 1984 SPIE Conference*, San Diego, Calif., August 1984, pp. 39-46.
11. Nash, J. G., and Petrozolin, C., "VLSI Implementation of a Linear Systolic Array," *Proceedings of ICASSP, 1985*, Tampa, Florida, March 26-29, 1985, pp. 1392- 1395.
12. Kung, H. T., Sproull, R., and Steele, G., *VLSI Systems and Computations*, Computer Science Press, Carnegie-Mellon University, Pa., Oct. 1981, pp. 367-378.
13. Gentleman, W. M., and Kung, H. T., "Matrix Triangularization by Systolic Arrays," *SPIE Real-Time Signal Processing IV*, San Diego, Calif., Aug. 1981, Vol. 298, pp. 19-26.
14. Yeh, H. G., "Kalman Filtering and Systolic Processors," *Proceedings of the IEEE International Conf. on Acoustic, Speech, and Signal Processing*. Tokyo, Japan, April 1986, pp. 2139-2142.

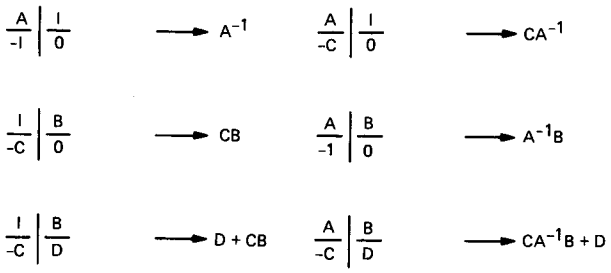

Fig. 1. Examples of a variety of matrix operations possible with the Faddeev Algorithm

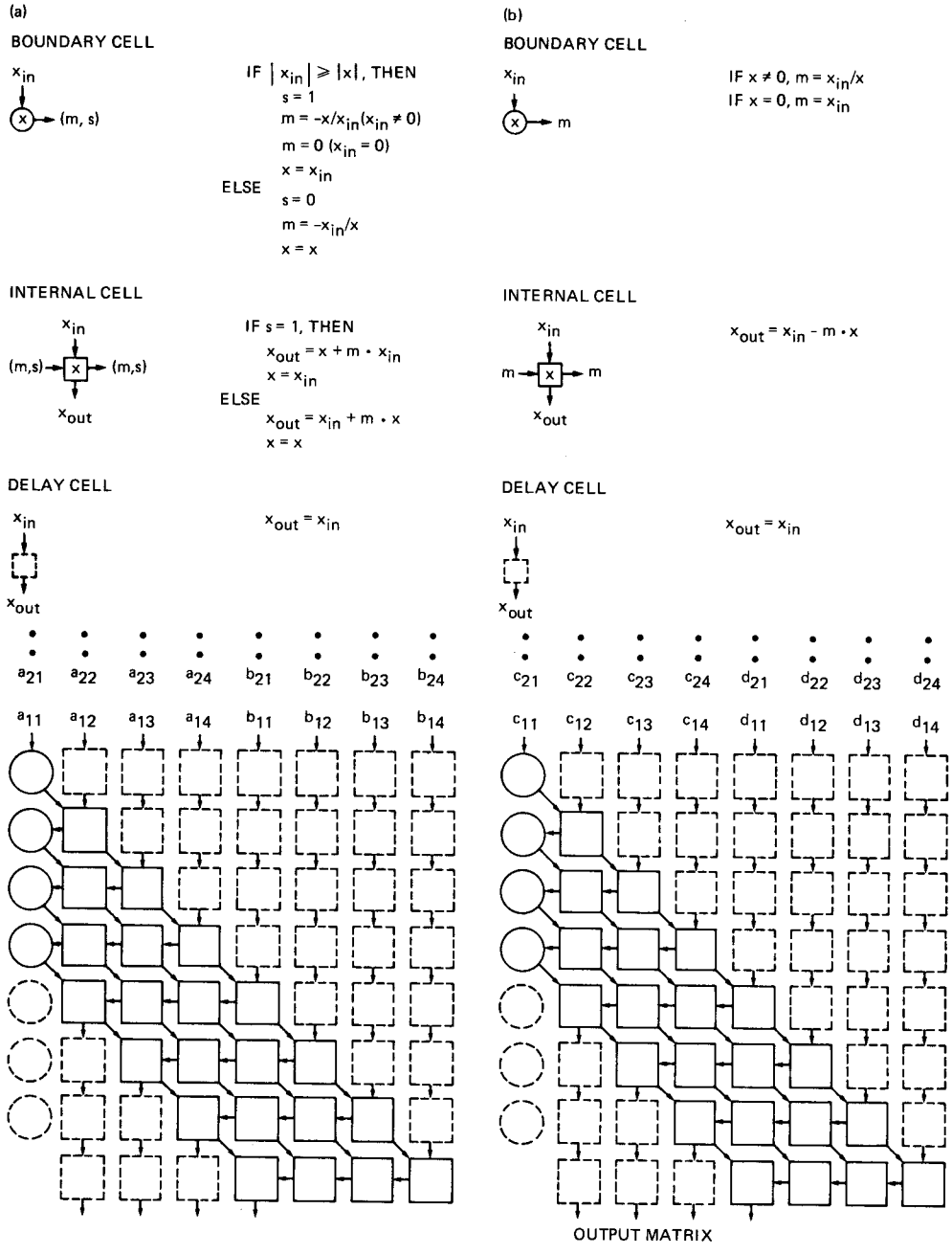

Fig. 2. Data Flow and PE arrangement: (a) triangularization process, $n = 4$; (b) annulling process, $n = 4$

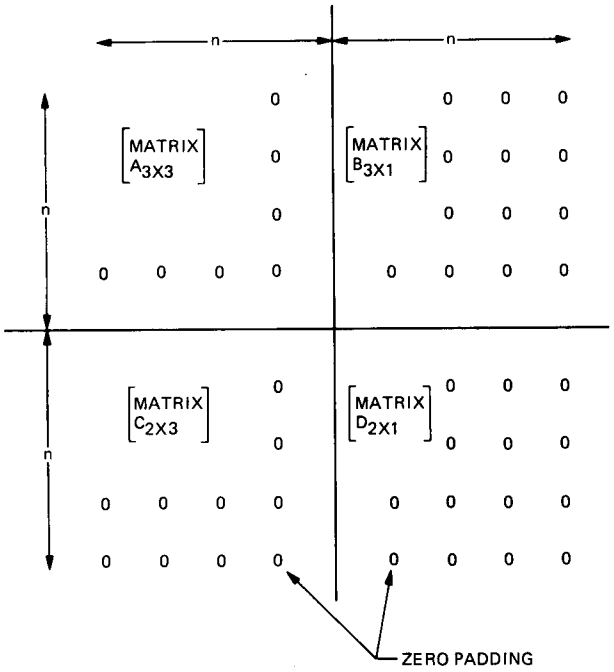

Fig. 3. Zero padding scheme for implementing a small-size matrix/vector on processor arrays

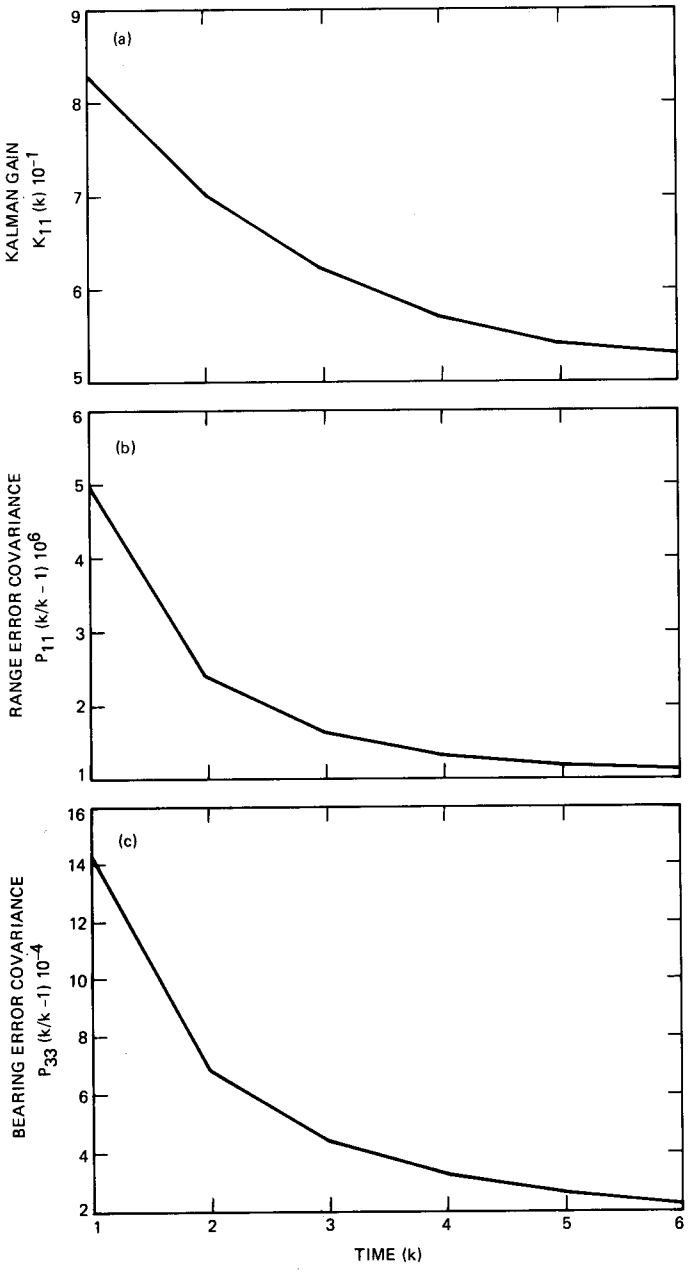

Fig. 4. Computer simulation on the numerical computation of a Kalman filter: (a) K_{11} (k) vs time *k*; (b) P_{11}(k/k − 1) vs time *k*, and (c) P_{33}(k/k − 1) vs time *k*