[18] M. Vetterli, "Running FIR and IIR filtering using multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, no. 5, pp. 730–738, May 1988.
[19] A. B. Watson, "The cortex transform: Rapid computation of simulated neural images," *Comput. Graphics Image Processing*, vol. 39, no. 3, Sept. 1987.
[20] J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 5, pp. 1278–1288, Oct. 1986.

A Parallel Adaptive Algorithm for Moving Target Detection and Its VLSI Array Realization

Chi-Min Liu and Chein-Wei Jen

***Abstract*—The application of an innovations-based detection algorithm (IBDA) in moving target detector (MTD) radars has been shown to be efficient. In this correspondence, an adaptive algorithm for the IBDA is derived. This algorithm behaves well in numerical properties and computational complexity. Also, it possesses high computing parallelism and data locality which lead to the feasibility of VLSI array realization. A systolic array is designed with the iteration time being $(2n + 1)$ and the number of processing elements being $[(n^2 + 3n)/2]$, where n is the length of the adaptive filter.**

I. Introduction

The use of an innovations-based detection algorithm (IBDA) in a moving target detector (MTD) has been shown to be more efficient than traditional methods [1]. However, the highly nonstationary environment in an MTD necessitates choosing adaptive algorithms which possess both fast adaptation and good steady-state behavior. The performance in the simulation of [1] showed that the Kalman filter [2] was the preferred algorithm in the IBDA. Although the Kalman filter is a theoretically optimum estimator, its computational complexity and numerical properties should be seriously considered. It is known that the computational complexity of the Kalman filter is $O(n^3)$ [2], [3], where n is the filter length. If such large computations are executed serially (as in the case of traditional computers), the limited sampling rate will severely limit its real-time applications. Additionally, the numerical instability of the filter is one factor that should be considered in nonstationary environments. The numerical error arises from the accumulation of quantization errors in finite precision arithmetic. It affects the implementation cost by increasing the required precision and may cause the phenomenon of "divergence" [4]. Several square-root algorithms [3] were developed to address this concern. Recently, a square-root algorithm was applied to IBDA and shown to provide better performance than the conventional Kalman filter [5]. Despite superior numerical properties, both square-root algorithms and the conventional Kalman filter are numerically unstable. In addition, the computational complexity of square-root algorithms is higher than that of the conventional Kalman filter. In this correspondence, we derive an adaptive algorithm for IBDA. The algorithm has a lower computational complexity than the Kalman filter and square-root algorithms. The new algorithm is also numerically stable and has the potential for parallel computation in VLSI arrays. Therefore, VLSI implementations with a reasonable level of precision can be adopted to attain fast computation and good steady-state performance.

With the advent of parallel processing systems, great interest has been focused on obtaining high processing speeds through parallelism in algorithms hitherto considered sequential in nature. Among different structures of parallel systems, VLSI arrays [6] can meet the high processing speed demands and make VLSI implementation feasible. The multiple and pipeline processing in VLSI arrays are means to attain high processing speeds while structural regularity and local interconnections contribute to the suitability of VLSI implementation. Despite all the benefits of VLSI arrays, one important point is that not every existing algorithm is suitable for VLSI array realization. In the literature, some VLSI arrays [7]–[11] were designed based on the square-root algorithm proposed by Paige and Saunders [12]. The Paige and Saunders' algorithm adopts the weighted least squares approach in its derivative process and leads to the feasibility of a VLSI array realization [7]–[11]. In this correspondence, we derive an adaptive algorithm for the IBDA based on such an approach. The specific application of the IBDA is taken into account in the derivation process to simplify the complexity and minimize numerical instability. A systolic array is designed with the iteration time being $(2n + 1)$ and the number of processing elements (PE's) being $[(n^2 + 3n/2]$, where n is the filter length. The new array presented in this correspondence outperforms those arrays derived from other design approaches [7]–[11].

II. The Adaptive Algorithm for the IBDA

A. The IBDA in MTD

The detection of a target echo $s(j)$ in the combined presence of clutter and receiver noises is formulated in terms of hypothesis testing, i.e.,

$$\text{Hypothesis } H_1\text{: } y(j) = s(j) + c(j) + w(j), \quad j = l - N + 1, \cdots, l \tag{2.1a}$$

$$\text{Hypothesis } H_0\text{: } y(j) = c(j) + w(j), \quad j = l - N + 1, \cdots, l \tag{2.1b}$$

where $\{y(j)\}$ is the complex baseband signal from the lth range-azimuth cell of a surveillance radar, $\{s(j)\}$ is a target process, $\{c(j)\}$ is usually a non-Gaussian clutter process, and $\{w(j)\}$ is a white-Gaussian noise process. The data record $\{y(j), j = l - N + 1, \cdots, l\}$ is obtained by sampling the N consecutive radar returns from a specific range ring as the radar scans across the lth azimuth cell. N is the number of pulses illuminating the range-azimuth cell of interest.

By using the chain rule, the likelihood ratio of the hypotheses testing becomes

$$\ln \Lambda_{H_1/H_0}(l) = \frac{1}{2} \sum_{k=l-N+1}^{l} \left[\ln \left(\frac{\sigma^2(k \mid H_1)}{\sigma^2(k \mid H_1)} \right) + \frac{|\bar{y}(k \mid H_0)|^2}{\sigma^2(k \mid H_0)} - \frac{|\bar{y}(k \mid H_1)|^2}{\sigma^2(k \mid H_1)} \right] \tag{2.2}$$

where $\{\bar{y}(k \mid H_i) = y(k) - \hat{y}(k \mid H_i), k = 0, \cdots, N - 1\}$ is the set of prediction errors or innovation process on the hypothesis that H_i is correct [1]. $\sigma^2(k \mid H_i)$ is its variance, for $i = 0, 1$. Fig 1 [1]

Manuscript received November 8, 1989; revised August 9, 1991. This work was supported by the National Science Council, Taiwan, Republic of China, under Grant NSC80-0404-E009-39.

The authors are with the Institute of Electronics, National Chiao Tung University, Hsinchu, 30039, Taiwan, Republic of China.

IEEE Log Number 9202789.

1053-587X/92$03.00 © 1992 IEEE

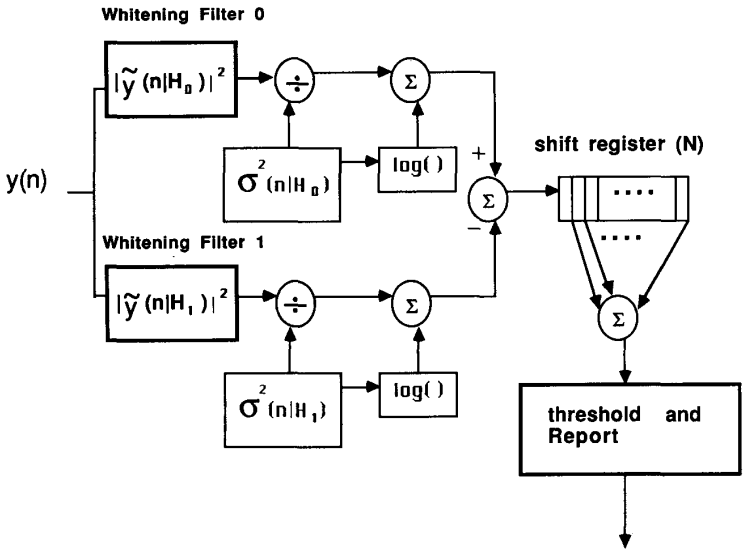

Fig. 1. Block diagram of IBDA.

illustrates an IBDA operation by calculating the "one pulse information" and using shift registers to sum up N terms.

The kernel of the IBDA shown in Fig. 1 is the whitening filters that support the hypothesis H_1 and H_0. If the whitening filters are stable and converge fast enough to adapt to the changes of a nonstationary environment, the MTD will behave well. In the realization of the IBDA, the two whitening filters designed for hypothesis H_1 and H_0 should be modeled correctly for the corresponding assumption. Experiments showed that the use of Kalman filtering (assuming the random-walk state model) for these two filters will track the localized nonstationarity caused by the presence of a target echo when the additive clutter is stationary (as in the case of ground clutter) as well as nonstationary (as in the case of weather clutter) [1]. It provides at least 3 dB average improvement for weather clutter- and ground clutter-dominated radar data as compared to the conventional MTD algorithms.

B. The Adaptive Algorithm

Since hypotheses H_0 and H_1 can be modeled approximately by autoregressive (AR) processes, the adaptive transversal filter with random-walk state model can be used [1]. The system is expressed as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k' \tag{2.3}$$

$$y_k' = \boldsymbol{C}_k' \boldsymbol{x}_k + w_k'. \tag{2.4}$$

The dimension of state vector $\boldsymbol{x}_k$ is $n \times 1$. The observation y_k' is the current received data and is a scalar. The dimension of coefficient vector $\boldsymbol{C}_k$ is $1 \times n$ and is constituted by the radar signals come from the most recent steps [1]. The Gaussian processes $\{\boldsymbol{v}_k'\}$ and $\{w_k'\}$ are assumed zero mean and uncorrelated, i.e.,

$$E(\boldsymbol{v}_k') = \boldsymbol{0}; \quad E(\boldsymbol{v}_k' \cdot \boldsymbol{v}_k' H) = \boldsymbol{V}_k; \quad E(w_k') = 0;$$
$$E(w_k' + w_k'^H) = W_k, \quad \text{and} \quad E(w_j \boldsymbol{v}_k') = 0$$

where the superscript H denotes the Hermitian transpose of a matrix or a vector. Let

$$\boldsymbol{L}_k \boldsymbol{L}_k^H = \boldsymbol{V}_k^{-1} \quad \text{and} \quad \mathfrak{L}_k \cdot \mathfrak{L}_k^* = W_k^{-1} \tag{2.5}$$

be the Cholesky decompositions of the inverse of the covariance matrices, where the superscript * denotes the complex conjugation of a scalar. Premultiplying the $\boldsymbol{L}_k^H$ and $\mathfrak{L}_k^*$ in (2.3) and (2.4) separately, they become

$$\boldsymbol{0} = \boldsymbol{L}_k^H \boldsymbol{x}_{k+1} - \boldsymbol{L}_k^H \boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.6}$$

and

$$y_k = \mathfrak{L}_k^* \cdot y_k' = \mathfrak{L}_k^* \cdot \boldsymbol{C}_k' \cdot \boldsymbol{x}_k + \mathfrak{L}_k^* \cdot w_k' = \boldsymbol{C}_k \boldsymbol{x}_k + w_k \tag{2.7}$$

where

$$\boldsymbol{v}_k \equiv -\boldsymbol{L}_k^H \cdot \boldsymbol{v}_k', \quad \text{and} \quad E(\boldsymbol{v}_k) = 0, \quad E(\boldsymbol{v}_k \boldsymbol{v}_k^H) = \boldsymbol{L}_k^H \boldsymbol{V}_k \boldsymbol{L}_k = \boldsymbol{I} \tag{2.8}$$

$$w_k \equiv \mathfrak{L}_k^* \cdot \mathrm{w}_k', \quad \text{and} \quad E(w_k) = 0, \quad E(w_k w_k^H) = 1 \tag{2.9}$$

and

$$\boldsymbol{C}_k \equiv \mathfrak{L}_k^* \cdot \boldsymbol{C}_k'. \tag{2.10}$$

So (2.6) and (2.7) with the iterations from 1 to k are combined to give the following formulation:

$$\begin{bmatrix} \boldsymbol{0} \\ y_1 \\ \boldsymbol{0} \\ y_2 \\ \vdots \\ \boldsymbol{0} \\ y_k \end{bmatrix} = \begin{bmatrix} -\boldsymbol{L}_1^H & \boldsymbol{L}_1^H & & & \\ \boldsymbol{C}_1 & & & & \\ & -\boldsymbol{L}_2^H & \boldsymbol{L}_2^H & & \\ & \boldsymbol{C}_2 & & & \\ & & & \ddots & \\ & & & -\boldsymbol{L}_k^H & \boldsymbol{L}_k^H \\ & & & \boldsymbol{C}_k & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_1 \\ w_1 \\ \boldsymbol{v}_2 \\ w_2 \\ \vdots \\ \boldsymbol{v}_k \\ w_k \end{bmatrix}. \tag{2.11}$$

The least squares solution of (2.11) can be computed from the orthogonal transform composed of a sequence of Givens rotations

[13] and expanded as

$$\begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_{1,2} & & & \\ & \boldsymbol{R}_2 & \boldsymbol{R}_{2,3} & & \\ & & \cdots & & \\ & & & \boldsymbol{R}_{k-1} & \boldsymbol{R}_{k-1,k} \\ & & & & \tilde{\boldsymbol{R}}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{1,k} \\ \boldsymbol{x}_{2,k} \\ \vdots \\ \boldsymbol{x}_{k-1,l} \\ \boldsymbol{x}_{k,k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{k-1} \\ \tilde{\boldsymbol{b}}_k \end{bmatrix} \tag{2.12}$$

where submatrices $\boldsymbol{R}_i$, $i = 1, 2, \cdots, k$, are all upper triangular matrices. The optimum estimated vector $\hat{\boldsymbol{x}}_k$ at time k, i.e., $\hat{\boldsymbol{x}}_{k,k}$, can be obtained from the last line in (2.12), or $\tilde{\boldsymbol{R}}_k \cdot \hat{\boldsymbol{x}}_{k,k} = \tilde{\boldsymbol{b}}_k$. At the next iteration $k + 1$, the updating process depends on only the matrice in the last line of (2.12). Based on (2.11), we can construct the equation for the iteration $k + 1$:

$$\boldsymbol{Q}_{k+1}^H \left[\begin{array}{cc|c} \tilde{\boldsymbol{R}}_k & \boldsymbol{0} & \tilde{\boldsymbol{b}}_k \\ -\boldsymbol{L}_{k+1}^H & \boldsymbol{L}_{k+1}^H & \boldsymbol{0} \\ \boldsymbol{C}_{k+1} & & y_{k+1} \end{array}\right] = \left[\begin{array}{cc|c} \tilde{\boldsymbol{R}}_k & \tilde{\boldsymbol{R}}_{k,k+1} & \boldsymbol{b}_k \\ 0 & \tilde{\boldsymbol{R}}_{k+1} & \tilde{\boldsymbol{b}}_{k+1} \\ 0 & 0 & r_{k+1} \end{array}\right]. \tag{2.13}$$

The innovation process $\tilde{y}$ required in (2.2) is derived as

$$\tilde{y}_{k+1} = (\mathfrak{L}_{k+1}^*)^{-1}(y_{k+1} - \hat{y}_{k+1}) = (\mathfrak{L}_{k+1}^*)^{-1}(y_{k+1} - \boldsymbol{C}_{k+1} \cdot \hat{\boldsymbol{x}}_{k+1}).$$

Since

$$\hat{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{R}}_{k+1}^{-1} \cdot \tilde{\boldsymbol{b}}_{k+1} \tag{2.14}$$

it then follows that

$$\tilde{y}_{k+1} = (\mathfrak{L}_{k+1}^*)^{-1}(y_{k+1} - \boldsymbol{C}_{k+1} \cdot \tilde{\boldsymbol{R}}_{k+1}^{-1} \cdot \tilde{\boldsymbol{b}}_{k+1}). \tag{2.15}$$

The matrix inverse operation should be removed to improve its numerical properties. The rows of matrix $\boldsymbol{Q}_{k+1}$ can be separated into two groups labeled by $\boldsymbol{S}_{k+1}$ and $\boldsymbol{Z}_{k+1}$, i.e.,

$$\boldsymbol{Q}_{k+1} = \begin{bmatrix} \boldsymbol{S}_{k+1} \\ \boldsymbol{Z}_{k+1} \end{bmatrix} \tag{2.16}$$

where $\boldsymbol{Q}_{k+1}$ is a $(2n + 1)$-by-$(2n + 1)$ matrix, $\boldsymbol{S}_{k+1}$ is a $2n$-by-$(2n + 1)$ matrix, and $\boldsymbol{Z}_{k+1}$ is a 1-by-$(2n + 1)$ matrix. $\boldsymbol{Z}_{k+1}$ can be expressed as

$$\boldsymbol{Z}_{k+1} = [\bigstar \bigstar \bigstar \cdots \bigstar \eta_{k+1}] \tag{2.17}$$

where the symbol "$\bigstar$" stands for values which will not affect the proceeding derivation. Since $\boldsymbol{Q}_{k+1}^{-1} = \boldsymbol{Q}_{k+1}^H$, we may use (2.17) and (2.13) to deduce (2.15) to

$$\begin{aligned} \tilde{y}_{k+1} &= (\mathfrak{L}_{k+1}^*)^{-1} \cdot (y_{k+1} - \boldsymbol{C}_{k+1} \cdot \hat{\boldsymbol{x}}_{k+1}) \\ &= (\mathfrak{L}_{k+1}^*)^{-1} \cdot \boldsymbol{Z}_{k+1} \cdot [0 \quad 0 \cdots r_{k+1}]^t \\ &= (\mathfrak{L}_{k+1}^*)^{-1} \cdot \eta_{k+1} \cdot r_{k+1}. \end{aligned} \tag{2.18}$$

If the elementary Givens rotation for two elements h and k is represented as

$$\boldsymbol{G}\boldsymbol{Y} = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & c_i & \cdots & s_i & \\ & \vdots & & \vdots & \\ \cdots & -s_i & \cdots & c_i & \end{bmatrix} \begin{bmatrix} \vdots \\ h \\ \vdots \\ k \end{bmatrix} = \begin{bmatrix} \vdots \\ h' \\ \vdots \\ 0 \end{bmatrix} \tag{2.19}$$

where $c_i = \cos\phi = |h|/\sqrt{|h|^2 + |k|^2}$ and $s_i = \sin\phi = (k^*/h^*)\cos\phi$, it can be proved that η_k is the multiplication of all the sequent c_i's of the elementary Givens rotations which are used to eliminate the elements in matrix $\boldsymbol{C}_{k+1}$ in (2.13) [18], i.e.,

$$\eta_{k+1} = c_1 \cdot c_2 \cdots c_n = \prod_{i=1}^{n} c_i. \tag{2.20}$$

Combining (2.20) and (2.18), we get

$$\tilde{y}_{k+1} = (\mathfrak{L}_{k+1}^*)^{-1} \cdot r_{k+1} \cdot \prod_{i=1}^{n} c_i. \tag{2.21}$$

To summarize, this algorithm consists of three parts: i) the preprocessing which includes the Cholesky decompositions of the matrix $\boldsymbol{V}_k^{-1}$ and scalar W_k^{-1} in (2.5); ii) the orthogonal transformation in (2.13) for each iteration; and iii) the postprocessing in (2.21).

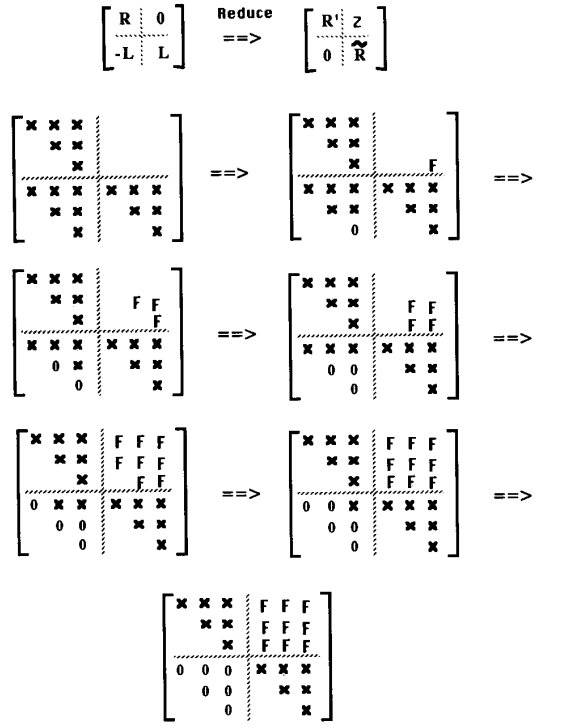

Fig. 2. The nullification progression of the $-\boldsymbol{L}$ matrix, where the blank spaces are zero elements. The F's indicated the filled-in elements during the process and the 0's indicate where nullification is taking place.

C. The Computational Complexity and the Numerical Analysis

Observing (2.13), the matrices $\boldsymbol{L}_{k+1}^H$ and $\tilde{\boldsymbol{R}}_k$ are upper triangular forms. To effectively take advantage of the sparseness in $\boldsymbol{L}_{k+1}^H$ and $\tilde{\boldsymbol{R}}_k$, a series of Givens rotations to eliminate elements in matrix

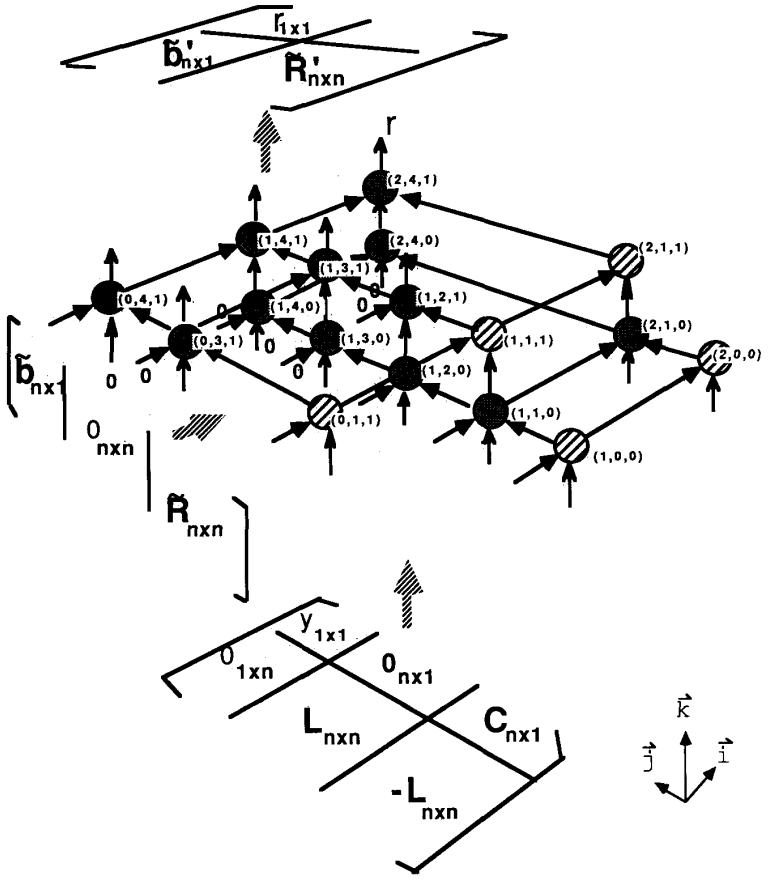

Fig. 3. The DG for the orthogonal transform in (2.13) with $n = 2$.

$-\boldsymbol{L}_{k+1}^H$ is as follows. For $i = n, n-1, \cdots, 1$, an element in the ith row of $-\boldsymbol{L}_{k+1}$ can be set to be zero by using the sth row of altered $\tilde{\boldsymbol{R}}_k$ to eliminate the sth element of the ith row of altered $-\boldsymbol{L}_{k+1}^H$; this must be done in the sequence $s = 1, 2, \cdots, n$. The progress can be demonstrated schematically in the case of $n = 3$ as shown in Fig. 2. It is interesting that if the reduction for the rows of matrix $-\boldsymbol{L}_{k+1}^H$ proceeds in such a reverse order, matrix $\boldsymbol{L}_{k+1}^H$ will reserve its triangular form in the elimination process and finally become $\tilde{\boldsymbol{R}}_{k+1}$.

For the preprocessing, it is sometimes assumed that the statistics of state noise $\{\boldsymbol{v}_k'\}$ and measurement noise $\{w_k'\}$ in the IBDA are obtained beforehand. As a result, we can determine the required values for matrix L_k and $\mathcal{L}_k^*$. Hence the desired computations in the preprocessing will be just the multiplication of $\boldsymbol{C}_k'$ by a scalar $\mathcal{L}_k^*$ in (2.10), i.e., n multiplications. Examining next the orthogonal transformation in (2.13) with the progress in Fig. 2, the number of rotations taken is $[n^3/3 + 3n^2 + 2n/3 + 1]$. One standard rotation requires 4 multiplications each, while it is possible to compute stable two- or three-multiplication rotations [13] with some added overhead. Assuming λ multiplications for one rotation in general cases, the complete rotations can be carried out in $[\lambda(n^3/3 + 3n^2 + 2n/3 + 1)]$ multiplications. The postprocessing is composed of $(n + 1)$ multiplications. The total multiplications for the algorithm are $\{n + [\lambda(n^3/3 + 3n^2 + 2n/3 + 1)] + (n + 1)\}$. Compared with the computational complexity $O(\rho n^3)$ (where ρ is much greater than one) in the conventional Kalman filter [2] and the square-root algorithms [3], the proposed algorithm has greatly simplified computational complexity.

Three considerations in the derivation process have manifested the superiority of the algorithm in numerical properties. First, the derivation process using (2.11) was shown to be a kind of square-root information Kalman filter [12], which does not have the difficulty of assuring the positive semidefinite of the covariance matrix in the computing process [3]. Second, the algorithm avoids the need of matrix inverse, which is a numerically unstable operation but necessary in the conventional Kalman filter and other square-root algorithms. Third, the limited dynamic range in Givens rotations [15] assures low quantization errors in the computing process.

III. VLSI Array Realization

A. The VLSI Array for the IBDA

Adopting Givens rotations for (2.13), the geometrical representation of the computing algorithm (called a dependence graph or DG [10], [14]) can be constructed as in Fig. 3 with $n = 2$. The functions of the nodes in Fig. 3 are illustrated in Fig. 4(b). In Fig. 3, the nodes mean the operations to be executed. The data in matrices $-\boldsymbol{L}_{k+1}$, $\boldsymbol{L}_{k+1}$, $\boldsymbol{C}_{k+1}$, and y_{k+1} flow into the DG in $\vec{k}$ direction, and the resultant data in matrices $\tilde{\boldsymbol{R}}_{k+1}$ and $\tilde{\boldsymbol{b}}_{k+1}$ go out from the upper part of the DG and reenter the DG along $\vec{i}$ direction for

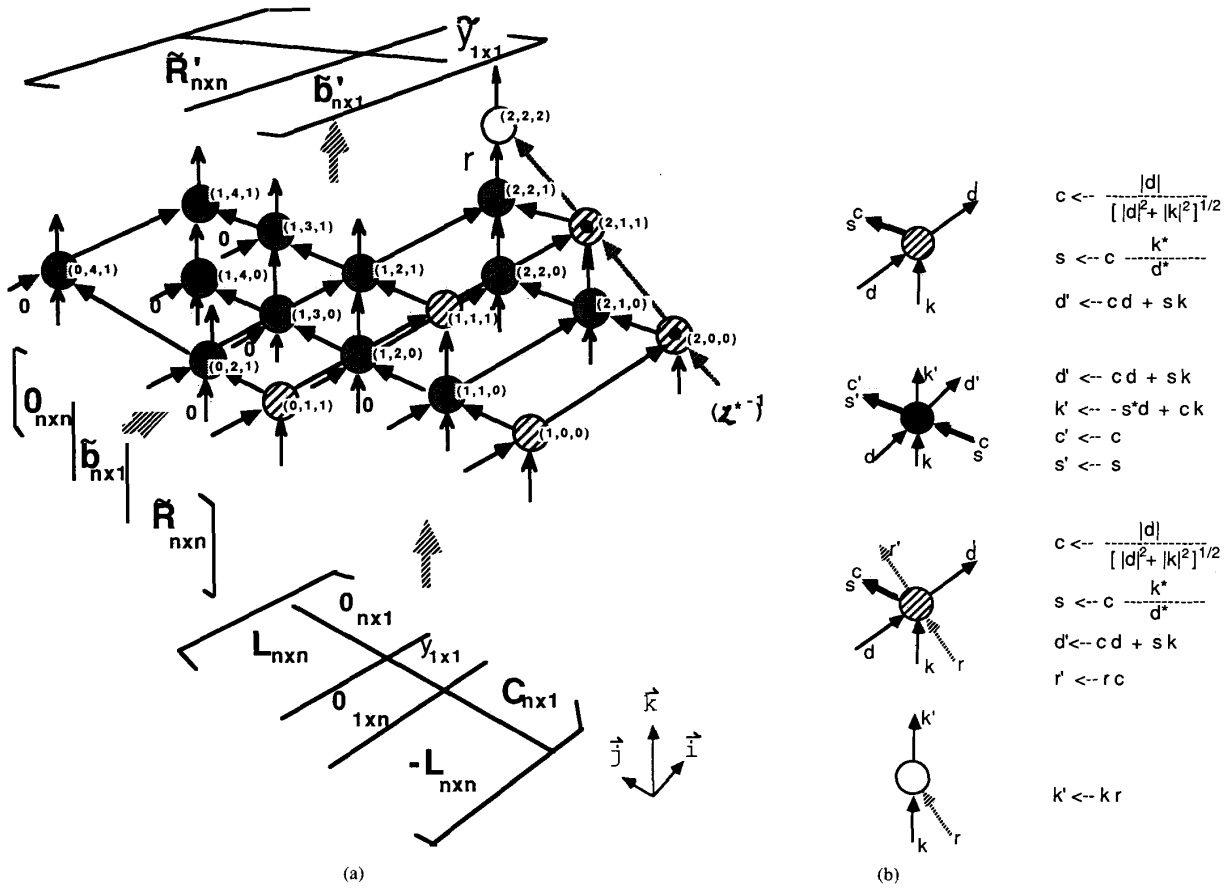

Fig. 4. (a) The DG for the presented algorithm with $n = 2$. (b) The functions of nodes for Figs. 3 and 4(a), where "*" means complex conjugation.

the next iteration. We label the nodes of different patterns for different kinds of operations. A directed arc denotes the data dependence between two nodes; that is, the computed result from one node should be sent along the arc for operating in another node. The relative positions of the nodes in the DG can be captured from the associated indexes in a Cartesian coordinate labeled in Fig. 3. From the geometrical representation of an algorithm, the computational complexity of (2.13) can be figured out immediately. In the graph, the number of nodes is $[n^3/3 + 3n^2 + 2n/3 + 1]$ which is consistent with the number of rotations given in Section II.

To embed the postprocessing of (2.21) into the DG in Fig. 3, we modify (2.13) as

$$\boldsymbol{Q}_{k+1}^H \begin{bmatrix} \tilde{\boldsymbol{R}}_k & \tilde{\boldsymbol{b}}_k & \boldsymbol{0} \\ -\boldsymbol{L}_{k+1}^H & \boldsymbol{0} & \boldsymbol{L}_{k+1}^H \\ \boldsymbol{C}_{k+1} & y_{k+1} & \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{R}}_k & b_k & \tilde{\boldsymbol{R}}_{k,k+1} \\ 0 & \tilde{b}_{k+1} & \tilde{\boldsymbol{R}}_{k+1} \\ 0 & r_{k+1} & 0 \end{bmatrix}. \quad (3.1)$$

Such a representation changes the computational sequence and the topology of the final array but will not change the correctness of the final data because the coefficients c_i and s_i in (2.19) are just obtained from the first n columns of (2.13). The final DG representing the computations of (3.1) and (2.21) is shown in Fig. 4. Fig. 4 shows that the data η_{k+1} and r_{k+1} are sent in the locally connected links for the multiplications in the white nodes. If (2.13) instead of (3.1) is used, nonlocally connected links should result and some delay elements in the final array will be needed additionally. Fig. 5 illustrates a 3-dimensional DG using four 2-dimensional layers with the filter length being three. Given a DG, the design issues are how to schedule the execution time for the nodes in the graph and how to design a suitable array to realize maximum parallelism. In Fig. 5, we assign all the nodes along the same line in direction $\vec{j}$, i.e., vector $[0 \quad 1 \quad 0]'$, executed by a processing element (PE). Such assignment is a linear transformation from 3-dimensional space to 2-dimensional space and was shown to be efficient in the systematic design methods for VLSI arrays [14]. The designed array is illustrated in Fig. 6. Direction $\vec{j}$ is adopted because it induces the minimal number of PE's among all directions. In the graph, $\tilde{\boldsymbol{R}}_{k+1}$ and $\tilde{\boldsymbol{b}}_{k+1}$ are generated in the upward links and reenter the array from the left input links for the next iteration. In Fig. 6(a), the "data reordering" is used to exchange the output sequence of the data in matrix $\boldsymbol{R}$ and vector $\boldsymbol{B}$ so that the required sequence in the next iteration is satisfied. In Fig. 6, tag control [16] is a one-bit data flow signal which is used to indicate PE's executing either the functions of the shaded nodes or those of the gray nodes. The execution time to annihilate the elements in $-\boldsymbol{L}_{k+1}^H$ and $\boldsymbol{C}_{k+1}$ and generate $\tilde{\boldsymbol{R}}_{k+1}$ and $\tilde{\boldsymbol{b}}_{k+1}$ is shown in

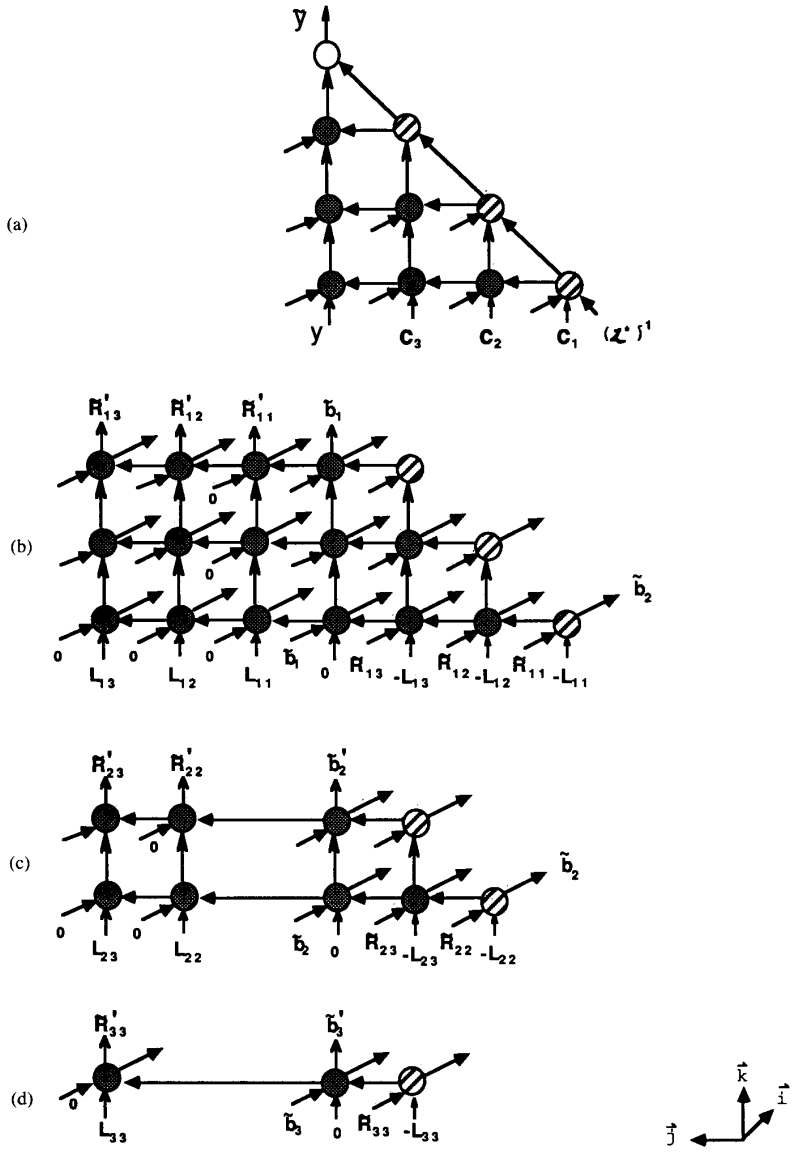

Fig. 5. The four layers of DG with $n = 3$, (a) the layer with $i = 3$, $k = 0, 1, 2, 3$, (b) the layer with $i = 2$, $k = 0, 1, 2$, (c) the layer with $i = 1$, $k = 1, 2$, and (d) the layer with $i = 0$, $k = 2$.

Fig. 7, where all the elements in matrices $-L_{k+1}^H$, C_{k+1}, b_{k+1}, L_{k+1}^H are manipulated at the times labeled in the table entries. Attentive readers may find that the computational progression in Fig. 2 is not coincident with the execution time in Fig. 7. The concept is, however, that a DG can be constructed from a computational progression, but the designed arrays will not necessarily follow the sequence of the progression. The array in Fig. 6 is designed from the DG constructed from the progression in Fig. 2. Since the DG captures the data dependence of the progression instead of its sequence, the computing sequence of the designed array retains the data dependence instead of its sequence. The designed systolic array shows the iteration time as $(2n + 1)$ and the number of PE's as $[1/2(n^2 + 3n)]$, where n is the number of the filter state. The iteration time is the minimum time interval between the initiation of the kth iteration and the $(k + 1)$th iteration. The clock period in the systolic array is the required time to compute the PE functions in Fig. 5(b). The CORDIC processor was suggested to implement these functions [17].

B. The Related VLSI Arrays in the Literatures

In the literature, various VLSI arrays [7]–[11] have been designed for the algorithm proposed by Paige and Saunders [12]. the computations of the Paige and Saunders' algorithm mainly consist of three steps: i) the whitening process composed of Cholesky decomposition of noise covariance and matrix multiplication; ii) the orthogonal transformation similar to (2.13); and iii) the backward substitution similar to (2.14). If the algorithm in this correspondence is compared with the Paige and Saunders' algorithm, the algorithm captures the applications of the IBDA to simplify the

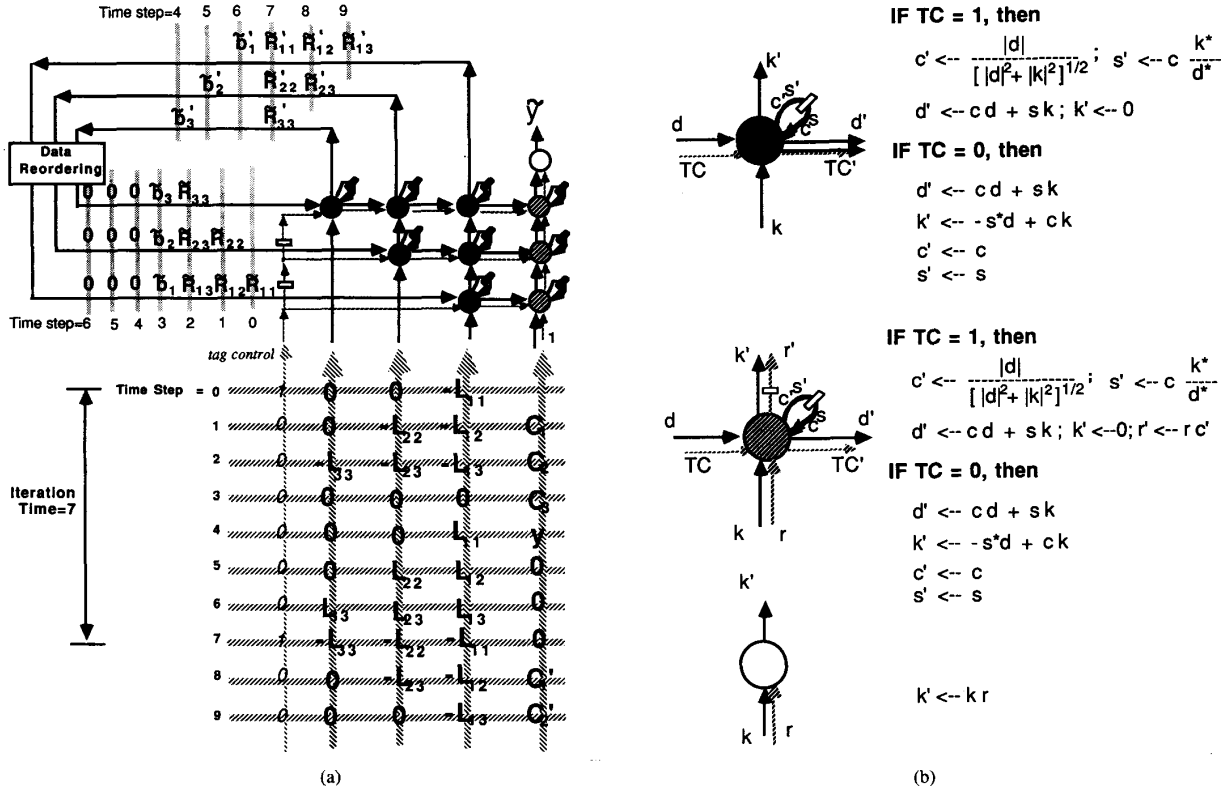

Fig. 6. (a) A systolic array with the number of processor elements being $[1/2(n^2 + 3n)]$ and the average computation time being $(2n + 1)$. (b) Functions of PE's, where TC stands for tag-control signal.

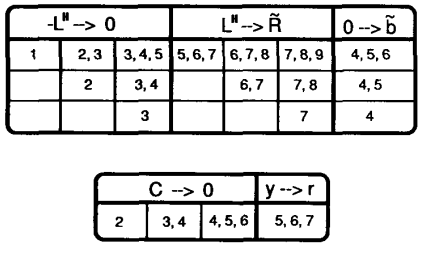

Fig. 7. Time sequence to manipulate the elements in matrices $-L^H_{k+1}$, b_{k+1}, L^H_{k+1}, C_{k+1}, and y_{k+1} of (3.1) in one iteration.

TABLE I
THE COMPARISONS OF VARIOUS DESIGN APPROACHES BASED ON RANDOM-WALK MODEL

Design Approaches \ Performance	Numbers of PE's	Iteration Steps	Average PE Utilization ($n \gg 1$)
Chen-Yao [7]	$(n/2)(3n + 5)$	$3n + 1$	7.4%
Gaston-Irwin [8]	$n(n + 1)$	$3n + 1$	11.1%
Kung-Hwang [9]	$(n/2)(n + 3)$	$4n + 1$	16.7%
Lincoin-Yao [10]	$(n/8)(5n + 13)$	$3n + 1$	17.78%
McWhirter-Shepherd [11]	$(n/2)(3n + 5)$	$3n$	7.41%
Liu-Jen	$(n/2)(n + 3)$	$2n + 1$	33.3%

computational complexity in step 1 and replace (2.15) by (2.21) to avoid unstable computation. Concerning the orthogonal transformation in step 2, we adopt the computational process in Section II to capture the sparseness of the matrice in (2.13) and reduce the computational complexity. Also, some structural difference between (2.13) and that of the orthogonal transformation in [12] can be found. The differences are deliberately constituted from the derivation process to induce the high computing speeds of the presented array. In addition, the projection direction $\vec{j}$, which is different from those in [7]-[11], is adopted to reduce the number of PE's. To give a clear insight into the effect, various arrays based on the design approaches [7]-[11] are deduced by applying the random-walk model in (2.3) and (2.4) to their system model. Their performance is summarized in Table I. Obviously, the array in Fig. 6 outperforms others in the iteration time, the number of PE's, and the average PE utilization. The average PE utilization is computed from

$$\frac{\text{no. of rotations}}{(\text{no. of PE's})(\text{iteration time})} \cdot 100\% = \frac{[n^3/3 + 3n^2 + 2n/3 + 1]}{(\text{no. of PE's})(\text{iteration time})} \cdot 100\%. \quad (3.2)$$

IV. Conclusions

In this correspondence, we derive an adaptive algorithm for MTD based on the following considerations: IBDA applications, computational complexity, numerical properties, and the feasible computing parallelism in VLSI arrays. A DG is presented for this algorithm. Based on the DG, a systolic array is designed with the iteration time being $(2n + 1)$ and the number of processing elements being $[1/2(n^2 + 3n)]$.

Acknowledgement

The authors are grateful to Prof. V. J. Mathews and the anonymous referees for their constructive suggestions.

References

[1] P. A. S. Metford and S. Haykin, "Experimental analysis of an innovations-based detection algorithm for surveillance radar," *Proc. Inst. Elec. Eng.*, pt. F, vol. 132, no. 1, pp. 18–26, Feb. 1985.

[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME J. Basic Eng.*, pp. 35–45, Mar. 1960.

[3] P. G. Kaminshi, A. E. Bryson, and S. F. Schmidt, "Discrete root filtering: A survey of current techniques," *IEEE Trans. Automat. Contr.*, vol. AC-16, no. 6, pp. 727–735, Dec. 1971.

[4] F. H. Schlee, C. J. Standish, and N. F. Toda, "Divergence in the Kalman filter," *J. AIAA*, vol. 5, pp. 1114–1120, June 1967.

[5] C. J. Kim, H. S. Lee, C. K. Un, and C. H. Kang, "Performance of adaption algorithms applied to innovation-based moving target detection," in *Proc. 1989 Int. Symp. Noise Clutter Rejection in Radars and Imaging Sensors* (Kyoto, Japan), pp. 508–513.

[6] H. T. Kung and C. E. Leiserson, "Systolic arrays (for VLSI)," *Proc. SIAM*, pp. 256–282, 1978.

[7] M. J. Chen and K. Yao, "On the realization of least squares estimation and Kalman filtering by systolic arrays," presented at the Int. Workshop Systolic Arrays, Univ. Oxford, U.K., Dep. External Studies, 1986.

[8] F. M. F. Gaston and G. W. Irwin, "A systolic square root information Kalman filter," in *Proc. Int. Conf. Systolic Array* (San Diego, CA), 1988, pp. 643–652.

[9] S. Y. Kung and J. N. Hwang, "An efficient triarray systolic design for real-time Kalman filtering," in *Proc. IEEE Int. Conf. ASSP*, (New York), 1988, pp. 2041–2044.

[10] R. A. Lincoin and K. Yao, "Efficient systolic filtering design by dependence graph mapping," in *VLSI Signal Processing, III*, R. W. Brodersen and H. S. Moscovitz, Eds. New York: IEEE, 1988, pp. 396–407; also presented at and based on the IEEE Workshop on VLSI Signal Processing, Monterey, CA, 1988.

[11] J. G. McWhirter and T. J. Shepherd, "Direct extraction of the state vector from systolic implementations of the square root Kalman filter," in *Proc. Int. Conf. Systolic Arrays* (Northern Ireland), 1989, pp. 42–51.

[12] C. C. Paige and M. A. Saunders, "Least squares estimation of discrete linear dynamic systems using orthogonal transformations," *SIAM J. Numer. Anal.*, vol. 14, no. 2, Apr. 1977.

[13] W. M. Gentleman, "Least squares computations by Givens transformation without square roots," *J. Inst. Math. Appl.*, 12, pp. 329–336, 1973.

[14] S. Y. Kung, S. C. Lo, and P. S. Lewis, "Optimum systolic design for transitive closure and shortest path problems," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 603–614, May 1987.

[15] W. M. Gentleman, "Error analysis of *QR* decompositions by Givens transformations," *Linear Alg. Appl.*, vol. 10, pp. 189–197, 1975.

[16] C.-W. Jen and H. Y. Hsu, "The design of a systolic array with tags input," in *Proc. Int. Symp. Circuits Syst.* (Finland), 1988, pp. 2263–2266.

[17] H. M. Ahmed, "Alternative arithmetic unit architectures for VLSI digital signal processors," in *VLSI and Modern Signal Processing*, S. Y. Kung and T. Kailath, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, pp. 277–303.

[18] J. G. McWhirter, "Recursive least squares minimization using a systolic array," *Proc. SPIE Int. Soc. Opt. Eng.*, pp. 105–110, 1983.

Simple Computational Methods of the AP Algorithm for Maximum Likelihood Localization of Multiple Radiating Sources

Seong Keun Oh and Chong Kwan Un

***Abstract*—In this correspondence, we present two simple computational algorithms of the alternating projection (AP) algorithm, which is an iterative algorithm for computing efficiently the deterministic maximum likelihood (ML) estimator of the locations of multiple sources in passive sensor arrays. One is a recursive projection (RP) algorithm that utilizes the projection matrix updating formula, and the other is a maximum eigenvector approximation (MEA) algorithm that approximates the Hermitian maximization problem in every iteration to a problem for maximizing the modulus of the projection onto the maximum eigenvector subspace. By transforming the computation of Hermitian forms into that of only inner products of vectors, these algorithms reduce significantly the computational complexity per iteration without any recognizable loss in the estimation performance and convergence behaviors. Computer simulation results that validate this approximation are also included.**

I. Introduction

The localization of radiating sources in passive sensor arrays is a problem of considerable importance in radar, sonar, seismology, and radio astronomy. Among various techniques [1]–[13], the maximum likelihood (ML) techniques provide optimum estimates to the source locations [6]–[13]. More recent works in dealing with the ML techniques have been devoted to decomposing the multidimensional search problem into a sequence of much smaller dimensional search problems, thus drastically reducing their computational complexity [7]–[10]. These algorithms have received considerable attention since they can handle the array of arbitrary geometry despite the reduced computational complexity.

Among these, the alternating projection (AP) algorithm requires the smallest number of iterations to convergence by making use of all the available information in every iteration [10]. Although the algorithm computes efficiently the deterministic ML estimator [7]–[13] of source locations, it still requires the computation of two Hermitian forms per search point in every iteration. Since, in general, the number of search points per iteration is far larger as compared to the number of sensors, the complexity per iteration in the AP algorithm is still excessive.

In this correspondence, we present two simple computational algorithms of the AP algorithm. We first develop an RP algorithm that transforms the computation of two Hermitian forms into that of only four inner products of vectors by utilizing the projection matrix updating formula [14]. Then, we develop an MEA algorithm that approximates the Hermitian maximization problem to the problem for maximizing the modulus of the projection onto the maximum eigenvector subspace.[1] Using this approximation, we reduce the computation of two Hermitian forms to that of only three

Manuscript received September 10, 1990; revised September 2, 1991.

S. K. Oh is with the Radio Systems Laboratory, Communication Systems RD Center, Information Systems Business, Samsung Electronics, Seoul, Korea.

C. K. Un is with the Communications Research Laboratory, Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Seoul, Korea.

IEEE Log Number 9202801.

[1]The subspace spanned by the eigenvector corresponding to the maximum eigenvalue (hereafter, this eigenvector is referred to as the maximum eigenvector).

1053-587X/92$03.00 © 1992 IEEE