THE RSMI ALGORITHM FOR AIRBORNE MTI RADAR

Eduardo Hermesmeyer[*]

Department of Electrical Engineering
Federal University of Santa Catarina
Florianópolis, SC - Brazil

José Carlos M. Bermudez[†]

Department of Electrical Engineering
Federal University of Santa Catarina
Florianópolis, SC - Brazil

ABSTRACT

We introduce a new space-time adaptive processing (STAP) algorithm for clutter mitigation in airborne MTI radar systems. The algorithm, named RSMI, is order-recursive and allows data processing in the fast-time, an idle period for most algorithms (data acquisition period). We show that the RSMI optimum output converges in the mean-square sense to the SMI optimum output as the number of pulses increases from 1 to the SMI order, always with smaller computational cost. We also show that the whole CPI may not be required to obtain satisfactory performance. Thus, the new algorithm opens the possibility of real-time adaptation to further reduce complexity.

1. INTRODUCTION

Real-world computing limitations is a constant concern in designing space-time adaptive processing (STAP) algorithms. Several papers on STAP algorithms emphasize the benefits of reduced-rank and reduced-dimension processing. The benefits are: less computational demanding algorithms and faster statistical convergence [1–3]. These algorithms may be either data independent or data dependent [4]. Data independent algorithms must match interference environment properties to obtain a satisfactory performance [5] (i.e., *ad hoc* methods). Optimal data dependent algorithms often rely on hard-to-implement routines such as the singular value decomposition (SVD) [6].

This work introduces a new order-recursive STAP algorithm that possesses atractive features for real-time signal processing. Data processing in the fast-time domain together with a reduced number of processed pulses *per* CPI are the main characteristics of the proposed algorithm. During the algorithm's derivation it will be clear that the filtered signal (to be compared with a threshold) will be computed without the need of explicitly computing the filter's weights. The incoming data is processed in the fast-time domain using an order-recursive structure [7]. The algorithm is thus named order-recursive SMI (RSMI). The paper is organized as follows. Section 2 briefly describes the signal models used. In section 3 the RSMI algorithm is derived, while in section 4 simulation results are presented. Finally, section 5 summarizes the most important aspects of the RSMI algorithm.

*This work has been funded by a CNPq scholarship.

†This work has been supported in part by CNPq under grants 308095/2003-0 and 472762/2003-6

2. BACKGROUND

Consider the $N \times M \times L$ dimensional data cube in Fig. 1. These dimensions denote the number of array sensors, the number of pulses processed on a CPI and the number of unambiguous range cells, respectively [1, 8]. Suppose we

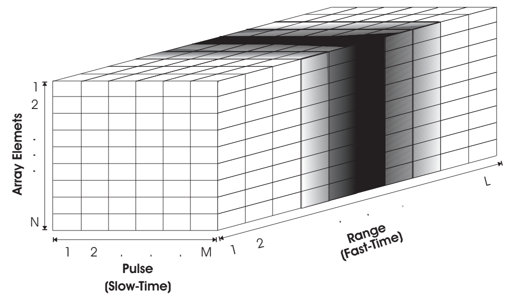

Fig. 1. Space-time radar data cube (in black: range cell under test; in gray: guard cells)

wish to estimate the presence/absence of a target at the ℓ-th range cell. If we know the space-time covariance matrix $\mathbf{R}$ of the statistically independent interfering signals, we simply solve the normal equations as follows:

$$\mathbf{w} = \gamma \mathbf{R}^{-1} \mathbf{s}_{s-t} \qquad (1)$$

where $\mathbf{w}$ is the $NM \times 1$ space-time weight vector, γ is a complex scalar, $\mathbf{R}$ is the $NM \times NM$ dimensional space-time covariance matrix and $\mathbf{s}_{s-t}$ is the $NM \times 1$ space-time steering vector, a unit norm vector which contains the exact information about the target's spatial and Doppler frequencies. Once $\mathbf{w}$ is determined, the filtered signal is given by:

$$y_\ell = \mathbf{w}^H \mathbf{x}_\ell, \qquad (2)$$

where H means Hermitian transposition, and $\mathbf{x}_\ell$ is the $NM \times 1$ space-time data (sample) vector corresponding to the ℓ-th range cell (data in the ℓ-th $N \times M$ face of the data cube arranged lexicographically into a column vector). Then, y_ℓ is compared to a threshold to decide whether or not the target is in the ℓ-th range cell. Most of the time, the space-time covariance matrix $\mathbf{R}$ is unknown. In this case, one solution to construct the normal equations (1) is to estimate $\mathbf{R}$ from the available samples (i.e., the data cube). This is known as the SMI algorithm [9]. The data from the cell under test, as well as from the neighboring cells (known as guard cells) should not be used in the estimation of $\mathbf{R}$ to prevent the target-nulling effect. It is clear that this estimation approaches the ideal covariance matrix as the number of sample vectors becomes large (assuming a wide-sense stationary environment). The sample support size K is the number of i.i.d sample vectors and the sample covariance matrix $\hat{\mathbf{R}}$ can be obtained as follows:

$$\hat{\mathbf{R}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k \mathbf{x}_k^H. \tag{3}$$

Note fom (1)–(3) that the space-time weight vector is obtained only after the whole data cube is assembled. Thus, estimation, signal filtering and target detection can only be performed after all the data acquisition is complete. In the next section we propose a recursive algorithm that filters the incoming data in the fast-time domain, allowing the detection to be made in the fast-time domain as well.

3. THE RSMI ALGORITHM

To derive the RSMI algorithm we first assume the interferences are zero-mean wide-sense stationary processes. Considering the snapshots from the first pulse ($m = 1$) are being received by the array, these fast-time (see Fig. 1) data are then collected to estimate the covariance matrix on the fly. The estimation still follows (3), except now the sample vectors $\mathbf{x}_k$ are $N \times 1$ vectors. The data snapshot received by the array in the fast-time domain (range) is arranged in an $N \times 1$ vector $\mathbf{x}_{m,k} = [x_{1,m,k} \; x_{2,m,k} \; \cdots \; x_{N,m,k}]^T$, where $x_{i,m,k}$ denotes the sample received at the i-th sensor, m-th pulse and k-th range cell; T denotes matrix transposition. L snapshots are collected for each transmitted pulse m, that is, $\mathbf{x}_{m,1} \quad \mathbf{x}_{m,2} \quad \cdots \quad \mathbf{x}_{m,L}$. Selecting K vectors which do not include the vectors corresponding to the range cell under test or the guard cells, we have the sample support for each range cell. Using these K vectors in (3) we obtain the first pulse covariance matrix $\hat{\mathcal{R}}_{\mathbf{11}}$. For the sake of simplicity we will refer to vector $\mathbf{x}_{m,k}$ as $\mathbf{x}_m$ throughout this text. We then estimate the first pulse SMI-MVDR (Minimum Variance Distortionless Response) weight vector from

$$\mathbf{w_1} = \gamma_1 \hat{\mathcal{R}}_{\mathbf{11}}^{-1} \mathbf{s_1} \tag{4}$$

where $\mathbf{s}_m$ is the space-time steering vector for the m-th pulse, that is, the first Nm components of vector $\mathbf{s}_{s-t}$:

$$\mathbf{s}_m = \mathbf{s}_{s-t}^{\lceil Nm \rceil} \tag{5}$$

and

$$\gamma_m = (\mathbf{s_m}^H \hat{\mathcal{R}}_{\mathbf{mm}}^{-1} \mathbf{s_m})^{-1} \tag{6}$$

is the m-th pulse complex scalar. Since (4) considers only samples from a single pulse, we do not have a space-time filter yet. Now, suppose the snapshots from the second pulse are being received. While receiving this data, we estimate its marginal sample covariance matrix $\hat{\mathcal{R}}_{\mathbf{22}}$ (as we did for the first pulse) and the sample cross-covariance matrix $\hat{\mathbf{\Gamma}}_2$. Using these estimations we assemble the $2N \times 2N$ space-time sample covariance matrix

$$\hat{\mathbf{R}}_{\mathbf{2}} = \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{1}} & \hat{\mathbf{\Gamma}}_2 \\ \hat{\mathbf{\Gamma}}_2^H & \hat{\mathcal{R}}_{\mathbf{22}} \end{bmatrix}. \tag{7}$$

In (7) $\hat{\mathbf{R}}_2$ is the augmented covariance matrix for the data corresponding to the first 2 pulses. Note that $\hat{\mathbf{R}}_{\mathbf{1}} = \hat{\mathcal{R}}_{\mathbf{11}}$, since for the first pulse we have an $N \times N$ covariance matrix. We now proceed to the m-th pulse. We, again, estimate the m-th pulse marginal sample covariance matrix $\hat{\mathcal{R}}_{\mathbf{mm}}$, as well as the $N(m-1) \times N$ sample cross-covariance matrix $\hat{\mathbf{\Gamma}}_m$ given by (8):

$$\hat{\mathbf{\Gamma}}_m = \frac{1}{K}\sum_{k=1}^{K} \begin{bmatrix} \mathbf{x}_1 \mathbf{x}_m^H \\ \mathbf{x}_2 \mathbf{x}_m^H \\ \vdots \\ \mathbf{x}_{m-1} \mathbf{x}_m^H \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{\Gamma}}_1 \\ \hat{\mathbf{\Gamma}}_2 \\ \vdots \\ \hat{\mathbf{\Gamma}}_{m-1} \end{bmatrix} \tag{8}$$

and, according to (7), we obtain the $N(m+1) \times N(m+1)$ augmented sample covariance matrix as:

$$\hat{\mathbf{R}}_{\mathbf{m+1}} = \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{m}} & \hat{\mathbf{\Gamma}}_m \\ \hat{\mathbf{\Gamma}}_m^H & \hat{\mathcal{R}}_{\mathbf{mm}} \end{bmatrix}. \tag{9}$$

For any given order m, the augmented covariance matrix $\hat{\mathbf{R}}_{m+1}$ can be factored into the LDL^H form, (where L and D are, respectively, lower triangular and block diagonal matrices), since $\hat{\mathbf{R}}_{m+1}$ is Hermitian and full-rank (some diagonal loading factor may be necessary). The factorization has the following form:

$$\begin{aligned} \hat{\mathbf{R}}_{m+1} &= \mathbf{L}_{m+1} \mathbf{D}_{m+1} \mathbf{L}_{m+1}^H \\ \begin{bmatrix} \hat{\mathbf{R}}_m & \hat{\mathbf{\Gamma}}_m \\ \hat{\mathbf{\Gamma}}_m^H & \hat{\boldsymbol{\mathcal{R}}}_{mm} \end{bmatrix} &= \begin{bmatrix} \mathbf{L}_m & \mathbf{O} \\ \boldsymbol{\mathcal{L}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{D}_m & \mathbf{O} \\ \mathbf{O}^H & \mathbf{\Sigma} \end{bmatrix} \begin{bmatrix} \mathbf{L}_m^H & \boldsymbol{\mathcal{L}}^H \\ \mathbf{O}^H & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{L}_m \mathbf{D}_m \mathbf{L}_m^H & \mathbf{L}_m \mathbf{D}_m \boldsymbol{\mathcal{L}}^H \\ \boldsymbol{\mathcal{L}} \mathbf{D}_m \mathbf{L}_m^H & \boldsymbol{\mathcal{L}} \mathbf{D}_m \boldsymbol{\mathcal{L}}^H + \mathbf{\Sigma} \end{bmatrix} \end{aligned} \tag{10}$$

where $\boldsymbol{\mathcal{L}} = \hat{\mathbf{\Gamma}}_m^H (\mathbf{D}_m \mathbf{L}_m^H)^{-1}$ and $\mathbf{\Sigma} = \hat{\boldsymbol{\mathcal{R}}}_{mm} - \boldsymbol{\mathcal{L}} \mathbf{D}_m \boldsymbol{\mathcal{L}}^H$. The m-th order weight vector can be obtained by solving the normal equations $\mathbf{R}_m \mathbf{w}_m = \mathbf{s}_m$ and, with the help of the LDL^H decomposition, we may write

$$\mathbf{k}_m = (\mathbf{L}_m \mathbf{D}_m)^{-1} \mathbf{s}_m \tag{11}$$

where $\mathbf{L}_m^H \mathbf{w}_m \triangleq \mathbf{k}_m$ is an intermediate weight vector that can be computed recursively as follows:

$$\begin{aligned} \mathbf{k}_{m+1} &= (\mathbf{L}_{m+1} \mathbf{D}_{m+1})^{-1} \mathbf{s}_{m+1} \\ &= \begin{bmatrix} \mathbf{k}_m \\ \mathbf{k} \end{bmatrix} \end{aligned} \tag{12}$$

where $\mathbf{k} \triangleq -\boldsymbol{\Sigma}^{-1} \boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} \mathbf{s}_m + \boldsymbol{\Sigma}^{-1} \mathbf{s}$. Note that $\mathbf{k}_m \in \mathcal{C}^{mN \times 1}$ and $\mathbf{k} \in \mathcal{C}^{N \times 1}$. Vector $\mathbf{s}_{m+1}$ is given by $\mathbf{s}_{s-t}^{\lceil N(m+1) \rceil}$, while $\mathbf{s}_m = \mathbf{s}_{m+1}^{\lceil Nm \rceil}$ and $\mathbf{s} = \mathbf{s}_{m+1}^{\lfloor N \rfloor}$.

Next, we derive a recursive expression for the complex scalar γ_{m+1} to obtain the MVDR filtered signal without the need for calculating the weight vector. From (6) we have:

$$1/\gamma_{m+1} = \mathbf{s}_{m+1}^H \hat{\mathbf{R}}_{m+1}^{-1} \mathbf{s}_{m+1} \tag{13}$$

$$= \left(\mathbf{L}_{m+1}^{-1} \mathbf{s}_{m+1}\right)^H \mathbf{k}_{m+1} \tag{14}$$

and the term $\left(\mathbf{L}_{m+1}^{-1} \mathbf{s}_{m+1}\right)$ is given by:

$$\mathbf{L}_{m+1}^{-1} \mathbf{s}_{m+1} = \begin{bmatrix} \mathbf{L}_m^{-1} & \mathbf{O} \\ -\boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{s}_m \\ \mathbf{s} \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} \mathbf{L}_m^{-1} \mathbf{s}_m \\ \mathbf{s} - \boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} \mathbf{s}_m \end{bmatrix} \tag{16}$$

From (14) and (16) we then have:

$$1/\gamma_{m+1} = \begin{bmatrix} \mathbf{L}_m^{-1} \mathbf{s}_m \\ \mathbf{s} - \boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} \mathbf{s}_m \end{bmatrix}^H \begin{bmatrix} \mathbf{k}_m \\ \mathbf{k} \end{bmatrix} \tag{17}$$

$$\begin{aligned} &= \left(\mathbf{L}_m^{-1} \mathbf{s}_m\right)^H \mathbf{k}_m \\ &+ \left(\mathbf{s} - \boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} \mathbf{s}_m\right)^H \mathbf{k} \end{aligned} \tag{18}$$

$$= 1/\gamma_m + 1/\gamma \tag{19}$$

whith $1/\gamma_m = \left(\mathbf{L}_m^{-1} \mathbf{s}_m\right)^H \mathbf{k}_m$ and $1/\gamma = \left(\mathbf{s} - \boldsymbol{\mathcal{L}} \mathbf{L}_m^{-1} \mathbf{s}_m\right)^H \mathbf{k}$. Finally, the filtered signal for a given range cell can also be obtained recursively as follows:

$$y_{m+1} = \mathbf{w}_{m+1}^H \mathbf{x}_{m+1} = \mathbf{k}_{m+1}^H \hat{\mathbf{x}}_{m+1} \tag{20}$$

where

$$\hat{\mathbf{x}}_{m+1} = \mathbf{L}_{m+1}^{-1} \mathbf{x}_{m+1} = \begin{bmatrix} \hat{\mathbf{x}}_m \\ \hat{\mathbf{x}} \end{bmatrix} \tag{21}$$

is the innovations vector [7] of process $\mathbf{x}_{m+1}$. Notice that $\mathbf{x}$ is the vector which contains the last N samples of $\mathbf{x}_{m+1}$, that is, $\mathbf{x} = \mathbf{x}_{m+1}^{\lfloor N \rfloor}$, while $\mathbf{x}_m$ is vector $\mathbf{x}_{m+1}$ without its last N samples ($\mathbf{x}_m = \mathbf{x}_{m+1}^{\lceil mN \rceil}$).

Using (12) and (21), we may then rewrite (20) as:

$$\begin{aligned} y_{m+1} &= \gamma_{m+1} \left[\mathbf{k}_m^H \quad \mathbf{k}^H\right] \begin{bmatrix} \hat{\mathbf{x}}_m \\ \hat{\mathbf{x}} \end{bmatrix} \\ &= \gamma_{m+1} \left(\mathbf{k}_m^H \hat{\mathbf{x}}_m + \mathbf{k}^H \hat{\mathbf{x}}\right) \\ &= \gamma_{m+1} \left(y_m + y\right) \end{aligned} \tag{22}$$

where $y_m = \mathbf{k}_m^H \hat{\mathbf{x}}_m$ and $y = \mathbf{k}^H \hat{\mathbf{x}}$. This completes the order-recursive algorithm. Notice that the filter output was computed without using the filter's weight vector $\mathbf{w}_m$, since we avoided the inefficient computation of the upper-triangular system of equations $\mathbf{w}_m = \left(\mathbf{L}_m^H\right)^{-1} \mathbf{k}_m$.

Diagonal loading [10] can be easily implemented in the RSMI algorithm. The addition of a weighted identity matriz to $\hat{\boldsymbol{\mathcal{R}}}_{mm}$ in (9) (for $m = 1$, inclusive) brings the same results that would be obtained in the SMI-DL case (see section 4). Notice also that either SMI-DL or RSMI-DL algorithms can use a reduced sample support, with same order as for the principal components method, as detailed in [4].

4. SIMULATION RESULTS

In this section we illustrate the validity of the equations for the order-recursive algorithm proposed. Assuming the optimal space-time MVDR weight vector (1) known, we compute the MSE between the optimum output and the outputs from the SMI-MVDR and RSMI-MVDR algorithms. For the RSMI case, we compute the MSE as a function of m, the number of processed pulses. For SMI, the MSE is constant since it's computed only once per CPI. Fig. 2 compares the MSE's for $N = M = 16$ and $\delta^2 = 10$dB. The clutter, jamming and thermal noise were modelled as zero-mean, wide-sense stationary Gaussian processes. We considered a power of 0dB for the thermal noise, a clutter-to-noise ratio of 40dB and a jammer-to-noise ratio of 50dB. Fig. 3 com-

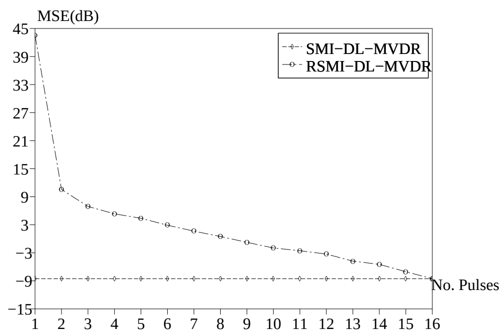

Fig. 2. MSE between the optimal filter and algorithms SMI-DL-MVDR and RSMI-DL-MVDR, with $N = M = 16$ and $\delta^2 = 10dB$

pares the computational effort of the SMI-MVDR, RSMI-

MVDR, SMI-DL-MVDR and RSMI-DL-MVDR algorithms in MFLOPS [11]. For the diagonally loaded version of algorithms SMI and RSMI, a reduced sample support $K_r = 2N_r$, given by Brennan's rule [8]

$$N_r = N + \beta(M - 1) \tag{23}$$

was used. Note that Fig. 3 does not take into account that RSMI starts operation well before (when the first snapshot is received) than SMI. Also, RSMI can be stopped for $m < M$ if monitoring reveals a sufficient SINR. This adds a new possibility of adaptation to reduce complexity. In the limit case ($m = M$), the computational burden is still smaller for the RSMI algorithm. Fig. 4 compares the SINR losses [4]

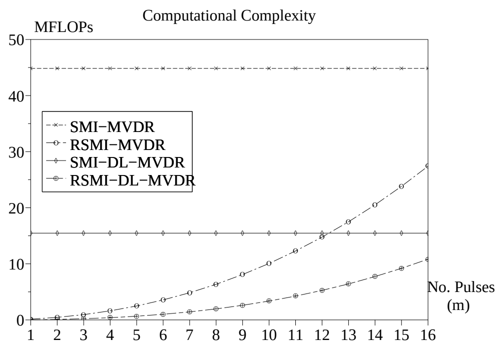

Fig. 3. Computational burden for the SMI-MVDR, RSMI-MVDR, SMI-DL-MVDR and RSMI-DL-MVDR algorithms with $N = M = 16$

for SMI, RSMI (with different orders) and the Principal Components (PC) algorithms. These results show a satisfactory performance of RSMI even with a reduced number of processed pulses, in agreement with the results in Fig. 2.

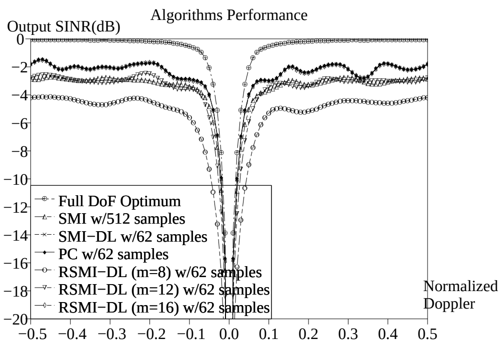

Fig. 4. SINR loss the optimal filter and algorithms SMI, RSMI ($m = 8, 12$ and 16) and PC, with $N = M = 16$

5. CONCLUSIONS

This paper proposed the RSMI algorithm for MTI radar. Its order-recursive nature allows data processing (and thus interference mitigation and target detection) in the fast-time domain, when other algorithms would still be idle collecting data. Reduced complexity is achieved and additional flexibility can be obtained by varying in real time the number of pulses used. The algorithm avoids the need for eigendecomposition and can be made robust to ill conditioned sample matrices through diogonal loading. Simulation results illustrated the anticipated algorithm efficiency.

6. REFERENCES

[1] William L. Melvin, "A STAP Overview," *IEEE Aerospace & Elect. Sys. Mag.*, vol. 19, pp. 19–35, Jan. 2004.

[2] J. Scott Goldstein and Irving S. Reed, "Theory of Partially Adaptive Radar," *IEEE Trans. on Aerospace and Elect. Sys.*, vol. 33, no. 4, pp. 1309–1325, Oct. 1997.

[3] J. R. Guerci, J. S. Goldstein, and I. S. Reed, "Optimal and Adaptive Reduced-Rank STAP," *IEEE Trans. on Aerospace and Elect. Sys.*, vol. 36, no. 2, pp. 647–663, Apr. 2000.

[4] J. R. Guerci, *Space-Time Adaptive Processing for Radar*, Artech House, Norwood, MA, 2003.

[5] Richard Klemm, *Principles of Space-Time Adaptive Processing*, IEE, London, 2002.

[6] A. Steinhardt and J. R. Guerci, "STAP for RADAR: what works, what doesn't, and what's in store," in *Proc. of the IEEE Radar Conf.*, Apr. 2004, pp. 469–473.

[7] Dimitris G. Manolakis, Vinay K. Ingle, and Stephen M. Kogon, *Statistical and Adaptive Signal Processing*, McGraw-Hill, 2000.

[8] James Ward, "Space-Time Adaptive Processing for Airborne Radar," in *Proc. of the ICASSP*, May 1995, pp. 2809–2812.

[9] L. E. Brennan and I. S. Reed, "Theory of Adaptive Radar," *IEEE Trans. on Aerospace and Elect. Sys.*, vol. 9, no. 2, pp. 237–252, Mar. 1973.

[10] Blair D. Carlson, "Covariance Matrix Estimation Errors and Diagonal Loading in Adaptive Arrays," *IEEE Trans. on Aerospace and Elect. Sys.*, vol. 24, no. 4, pp. 397–401, Jul. 1988.

[11] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Johns Hopkins, Baltimore, Maryland, 1989.