NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

FAST HIGH-RESOLUTION TECHNIQUES APPLIED
TO THE DIRECTION OF ARRIVAL (DOA)
OF SIGNALS PROBLEM

by

Milton Pinto Ferreira

September 1992

Thesis Advisor: Monique P. Fargues
Co-Advisor: Ralph Hippenstiel

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) 3A	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No. / Project No. / Task No. / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
FAST HIGH-RESOLUTION TECHNIQUES APPLIED TO THE DIRECTION OF ARRIVAL (DOA) OF SIGNALS PROBLEM

12. PERSONAL AUTHOR(S) Milton Pinto Ferreira

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	September 1992	113

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	High-Resolution, Minimum Norm, Direction of Arrival, Angle of Arrival, Eigenstructure, Autocorrelation Matrix, Correlation Matrix, Tracking, Moving sources.

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

Subspace decomposition methods are a very useful technique to extract the signal information via eigen-based estimators. Although those techniques are very accurate, they are usually expensive to update, becoming difficult to implement for real-time applications. The Rank-Revealing QR (RRQR) factorization introduced by Chan, offers an attractive alternative to perform the subspace selection. In this work, we use the RRQR algorithm applied to the Direction of Arrival (DOA) problem to track moving sources, using passive linear arrays. In addition to the regular RRQR algorithm originally proposed by Chan, this thesis introduces an improvement. This refinement uses the signal subspace information and requires very little additional computation. It takes advantage of the Hermitian property of the signal correlation matrix and is implicity equivalent to applying one subspace iteration step to the estimated signal subspace. Simulations show that the performance obtained is equivalent to that obtained using classical eigen-based techniques. Alternative algorithms for finding an approximation of the smallest singular vector of the correlation matrix are discussed and compared to the original method. The final product is an adaptive algorithm that allows for tracking of DOA angles when moving sources are present. An analysis of the number of flops needed to execute the adaptive algorithm based on the RRQR factorization to track the DOA of moving sources has been included, showing its feasibility for being used in real-time implementations.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Monique P. Fargues	22b. TELEPHONE (*Include Area code*) (408) 646 2859	22c. OFFICE SYMBOL EC/Fa

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete

Approved for public release; distribution is unlimited.

Fast High-Resolution Techniques
Applied to the Direction of Arrival (DOA)
of Signals Problem

by

Milton Pinto Ferreira Filho
Lieutenant Commander, Brazil Navy
B.S., Brazil Naval Academy
ESEE, University of São Paulo, Brazil

Suomitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

Author: _____
Milton Pinto Ferreira Filho

Approved by: _____
Monique P. Fargues, Thesis Advisor

_____
Ralph Hippenstiel, Co-Advisor

_____
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Subspace decomposition methods are a very useful technique to extract the signal information via eigen-based estimators. Although those techniques are very accurate, they are usually expensive to update, becoming difficult to implement for real-time applications. The Rank-Revealing QR (RRQR) factorization introduced by Chan, offers an attractive alternative to perform the subspace selection. In this work, we use the RRQR algorithm applied to the Direction of Arrival (DOA) problem to track moving sources, using passive linear arrays. In addition to the regular RRQR algorithm originally proposed by Chan, this thesis introduces an improvement. This refinement uses the signal subspace information and requires very little additional computation. It takes advantage of the Hermitian property of the signal correlation matrix and is implicitly equivalent to applying one subspace iteration step to the estimated signal subspace. Simulations show that the performance obtained is equivalent to that obtained using classical eigen-based techniques. Alternative algorithms for finding an approximation of the smallest singular vector of the correlation matrix are discussed and compared to the original method. The final product is an adaptive algorithm that allows for tracking of DOA angles when moving sources are present. An analysis of the number of flops needed to execute the adaptive algorithm based on the RRQR factorization to track the DOA of moving sources has been included, showing its feasibility for being used in real-time implementations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. PROBLEM CONTEXT

Estimating the direction of arrival (DOA) of a signal is a problem of great importance in the operation theater of a task force at sea. The direction of arrival (DOA) vector may be used to distinguish a friendly approaching aircraft from a foe. If the aircraft is outside an expected bearing corridor, then it is generally thought to be an enemy aircraft. This expectation must be verified by other check points during the target identification and classification process. Accurate determination of the DOA of a signal is a very important issue in the operation of weapon systems and should be completed with extreme care.

The main goal of this work is to investigate the accuracy and speed of high resolution DOA techniques. The technique of choice will allow the DOA to be solved accurately in real time by an on board computer.

Passive linear equispaced phased array sensors are used to generate signals for the DOA determination. Initially, any signal received by the sensors is processed to estimate signal direction with consecutive samples tracking potentially moving targets.

B. THEORETICAL BACKGROUND

Consider the case of a linear equispaced array of n sensors receiving signals emitted from m sources. Furthermore, let us assume that the signal may be represented by a narrowband signal embedded in additive Gaussian white noise. The resulting signal received at the array is

$$x_t = s_t + n_t \ , \tag{1}$$

where s_t represents all narrowband components emitted by the source(s) and n_t is the additive noise.

The output at the q^{th} array element may be expressed [Ref. 1] as

$$y_{t,q} = \sum_{i=1}^{m} A_i \exp\left[j \cdot \left[w_c t - \frac{2\pi (q-1) \cdot d \cdot \sin \theta_i}{\lambda} + \Phi_i\right]\right] + n_{t,q}, \quad \text{for } 1 \le q \le n \tag{2}$$

where θ_i represents the i^{th} signal arrival angle, w_c represents the center frequency of the narrowband sources, d represents the array element spacing, A_i is the amplitude of each incoming signal, λ is the wavelength related to the traveling wave movement, ϕ_i represents the random phase of each incoming signal, assumed uniformly distributed over $[0,2\pi]$. Finally, $n_{t,q}$ is a zero mean random variable representing the noise at the q^{th} array element. The output vector at time t is a n-dimensional column vector

$$\underline{x_t} = [y_{t,1}, \ldots, y_{t,n}]^T. \quad (3)$$

The mode vector $\underline{m_\theta}$ is

$$\underline{m_\theta} = [1, \exp(-2\pi j d \sin(\theta)/\lambda) \ldots \exp(-2\pi j d(n-1)\sin(\theta)/\lambda)]^T. \quad (4)$$

The noise vector is

$$\underline{n_t} = [n_{t,1}, \ldots, n_{t,n}]^T. \quad (5)$$

Equations 3-5 may be used to write the received signal vector as

$$\underline{x_t} = \sum_{i=1}^{m} A_i \cdot \exp[j[w_c t + \Phi_i]] \begin{vmatrix} 1 \\ \exp(-2\pi j d \sin(\theta_i)/\lambda) \\ . \\ . \\ \exp(-2\pi j d(n-1)\sin(\theta_i)/\lambda) \end{vmatrix} + \underline{n_t}, \quad (6)$$

where the vector $\underline{x_t}$ is defined for continuous time. The estimated received signal correlation matrix is given by the following equation:

$$R_{xx} = \frac{1}{nest} \sum_{k=1}^{nest} \underline{x_k} \cdot \underline{x_k}^H = \frac{1}{nest} X \cdot X^H. \quad (7)$$

The vector, $\underline{x}_k$, is the expression defined in Equation (6), for each discrete time interval. The quantity nest is the number of estimates used to compute the autocorrelation matrix, R_{xx}.

The autocorrelation matrix, R_{xx}, may be used to extract the signal information by using high resolution techniques such as: Multiple Signal Classification (MUSIC) [Ref. 2] or Minimum Norm [Ref. 3]. These high-resolution techniques use the singular vector decomposition (SVD) or the eigenvalue decomposition (EVD) of the received signal autocorrelation matrix to estimate the DOA information. A new SVD (or EVD) decomposition is needed for each update when tracking moving sources. Since the SVD (or EVD) decomposition is computationally expensive, good results are not obtainable for real time systems. The goal of this work is to find a good approximation of the signal and the noise subspaces and to be able to use these approximations to find the desired signal information. To allow their use in real-time algorithms, it is anticipated that these approximations will be a compromise between accuracy and processing speed.

The method to be used is the Rank Revealing QR Factorization (RRQR). Assuming that $A\Pi=QR$ is the classical QR factorization of A, where Q is orthonormal, R is upper triangular, and Π is an orthonormal permutation matrix, this method provides a matrix of the form

$$R=\begin{vmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{vmatrix},$$

where the norm two of R_{22}, $\|R_{22}\|_2$, is small when A is ill-conditioned. This is accomplished by means of a special pivoting scheme following the cited QR factorization. From the dimension of the block R_{22} of this matrix, its rank may be deduced. This method also provides the means to find approximations for the signal and noise subspaces of the matrix A. It will be shown that this information is contained in the orthogonal matrix Q, generated from the RRQR. Details about the method follow in Section A of Chapter 2.

The RRQR will be used to decompose the noise-free autocorrelation matrix. This matrix is obtained by subtracting $\sigma^2 I$ from the signal autocorrelation matrix defined in (7), where σ^2 is the Gaussian white noise variance and I is the identity matrix. The need for the noise-free autocorrelation matrix is due to the special pivoting scheme developed by Chan [Ref. 4]. This scheme works for rank deficient or ill conditioned matrices. However, the autocorrelation matrix, R_{xx}, does not present this characteristic. Therefore, no accurate signal subspace information can be obtained from the RRQR decomposition of R_{xx}. To correct this deficiency of the algorithm, information about the Signal to Noise Ratio (SNR) is needed.

It is possible to update the classical QR decomposition of the noise-free autocorrelation matrix. The possibility of updating the RRQR will also be investigated, as this technique allows tracking of moving targets. The rank-one modification of a matrix will be used to update the matrices Q and R without accessing the updated noise-free autocorrelation matrix.

II. THE RANK-REVEALING QR (RRQR) FACTORIZATION ALGORITHM

This chapter introduces the theoretical background necessary to understand the RRQR algorithm presented by Chan [Ref. 4]. This decomposition will then be used to estimate the signal information.

A. THE RANK-REVEALING QR ALGORITHM

The QR factorization is a decomposition of a given m by n matrix A into three other matrices, Q, R and Π, so that $A\Pi=QR$. The matrix, $\Pi \in R^{n\times n}$, is an orthonormal permutation matrix, $Q \in C^{m\times n}$ is orthonormal, i.e., $Q^HQ=I_n$, and $R \in C^{n\times n}$ is upper triangular. The QR decomposition, using a fixed permutation Π, provides a unique pair of matrices Q and R.[Ref. 4]

When A is full rank, R is non-singular. However, if A is rank deficient with rank deficiency r, we may choose Π so that the rank deficiency of R is exhibited in the form of a small lower right block $R_{22}\equiv 0$, as shown in Equation (9). Note that since Q and Π are orthonormal matrices, their singular values are equal to one. Therefore, A and R have the same rank. The matrix R may be defined as

$$R=\begin{vmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{vmatrix}, \qquad (9)$$

where R_{22} is r by r [Ref. 4].

Let us assume that σ_i is the i^{th} singular value of A, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. It is possible to show that $\sigma_{n-r+1}(A) \leq \|R_{22}\|_2$ [Ref. 4]. Thus, if the norm two of the matrix R_{22}, $\|R_{22}\|_2$, is small, A has at least r small singular values. The converse is not true, that is, it is not guaranteed that, if A has r small singular values, its QR factorization will yield a small $\|R_{22}\|_2$.

Thus, it is necessary to find an algorithm that is able to reveal the matrix rank via a small R_{22} block. The Rank-Revealing QR (RRQR) factorization algorithm [Ref. 4] provides such a decomposition of A.

In this section, a brief discussion and the theoretical background of the method is provided. The main purpose here is to identify the characteristics which are important to the Direction of Arrival (DOA) problem. A thorough theoretical treatment of the RRQR factorization has been written by Chan [Ref. 4].

This proof starts from a rank-one deficient matrix. It will be extended later for a rank-r deficient matrix. Let us assume that a given matrix A is nearly rank-one deficient. The Theorem 2.1 of [Ref. 4] is reproduced here, adapted to the complex case.

THEOREM 2.1 Suppose that we have a vector $x \in C^n$ with $\|x\|_2=1$ such that $\|Ax\|_2=\epsilon$, and let Π be a permutation matrix such that if $\Pi^T x=y$, then $|y_n|=\|y\|_\infty$. Then if $A\Pi=QR$ is the QR factorization of $A\Pi$, then

$$|r_{nn}| \leq \sqrt{n}\ \epsilon.$$

Proof: First we note that since $|y_n| = \|y\|_\infty$ and $\|y\|_2 = \|x\|_2 = 1$, we have

$$|y_n| \geq \frac{1}{\sqrt{n}}. \tag{10}$$

Next, we have

$$Q^H A x = Q^H A \Pi \Pi^T x = Ry = \begin{vmatrix} \cdot \\ \cdot \\ r_{nn} y_n \end{vmatrix}. \tag{11}$$

Therefore,

$$\epsilon = \|Ax\|_2 = \|Q^H A x\|_2 = \|Ry\|_2 > |r_{nn} y_{nn}|, \tag{12}$$

and from (10) and (12) we have the desired relationship.■

Now, assume that $v_n \in C^n$ with $\|v_n\|_2 = 1$ is the smallest right singular vector of A, then we have

$$\|Av\|_2 = \sigma_n. \tag{13}$$

If we define

$$|(\Pi^T v)|_n = \|v\|_\infty, \tag{14}$$

where Π is the permutation as defined in Theorem 2.1, we see that $A\Pi$ has a QR factorization with pivot r_{nn} and that the magnitude of r_{nn} is less or equal to $|\sigma_n \sqrt{n}|$. In other words, the $(n,n)^{th}$ element of R is small. Therefore, it is possible to make r_{nn} small with an adequate choice of the permutation matrix.

Since we need only an adequate permutation matrix Π, an approximation of the SVD of A for the smallest singular vector, v, is adequate. This point shows the advantage of the RRQR algorithm over the SVD. An approximation of v may be computed much faster than the true singular vector. In Chapter 3, methods to find a good approximation for v is demonstrated.

The above results serve as the foundation to compute any QR factorization of A. An approximation of the smallest right singular vector of A is found. Then we determine Π, as in (14), and compute the QR factorization of $A\Pi$. Alternatively, the eigenvector corresponding to the eigenvalue closer to zero [Ref. 5] may be found, rather than the smallest right singular vector v. The proof that this algorithm is valid for the eigenvector case is trivial, as σ is replaced by $|\lambda|$ throughout the proof.

The above one-dimensional algorithm may be extended to the case when A is nearly rank-r deficient, with r>1. We want to find a permutation Π, such that

$$A\Pi = QR = Q\begin{vmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{vmatrix} \qquad \textbf{(15)}$$

is the QR factorization of A. The submatrix R_{22} is (r x r) dimensional and $\|R_{22}\|_2$ is small. We apply the one dimensional algorithm presented above to R_{11}, for r=1,2, ..., where R_{11} is

the leading principal (n-r x n-r) dimensional submatrix of R. After isolating a (r x r) dimensional R_{22} block, we may use this one-dimensional algorithm to compute a permutation P such that $R_{11}P=Q_1\hat{R}_{11}$ is the QR factorization of $R_{11}P$. Next, we isolate a (r+1 x r+1) block. This guarantees the $(n-r,n-r)^{th}$ element of $\hat{R}$ is small, leading to:

$$A\hat{\Pi} = \hat{Q}\begin{vmatrix} \hat{R}_{11} & Q_1{}^T R_{12} \\ 0 & R_{22} \end{vmatrix}, \tag{16}$$

where

$$\hat{\Pi} = \Pi\begin{vmatrix} P & 0 \\ 0 & I \end{vmatrix}, \tag{17}$$

and

$$\hat{Q} = Q\begin{vmatrix} Q_1 & 0 \\ 0 & I \end{vmatrix}. \tag{18}$$

We notice that Equation (16) is the QR factorization of $A\hat{\Pi}$. The complete algorithm is summarized below in steps 0 to 9. An implementation of this subroutine using the MATLAB™ software may be found in Appendix A.

0. Compute a first QR decomposition of the noise-free autocorrelation matrix.

"For i=n,n-1,...,n-r+1, do:

1. Let R_{11} be the leading i x i block of R.

2. Compute the singular vector $v \in C^i$ of R_{11}, corresponding to the minimum singular value $\sigma_{min}(R_{11})$ with $\|v\|_2=1$.

3. Compute a permutation $P \in C^{i \times i}$ such that $|(P^H v)_i| = \|P^H v\|_\infty$. (This means find the maximum absolute value element of the smallest right singular vector and swap it with the i^{th} element of the same vector).

4. Assign $\tilde{v} = \begin{vmatrix} v \\ 0 \end{vmatrix} \in C^n$ to the i^{th} column of W.

5. Let $W = \hat{P}^H W$, where $\hat{P} = \begin{vmatrix} P & 0 \\ 0 & I \end{vmatrix}$.

6. Compute $Q_1 \hat{R}_{11}$, the QR factorization of $R_{11} P$.

7. Let $\Pi = \Pi \hat{P}$.

8. Let $Q = Q \begin{vmatrix} Q_1 & 0 \\ 0 & I \end{vmatrix}$.

9. Let $R = \begin{vmatrix} \hat{R}_{11} & Q_1^T R_{12} \\ 0 & R_{22} \end{vmatrix}$.

For this algorithm to provide the desired R_{22} block of R small in norm, we must have a R_{11} block at step two with a small singular value to insure that the $(i,i)^{th}$ element of $\hat{R}_{11}$ is small. At step nine, the $(n-i)^{th}$ (last) row of $Q^H_1 R_{12}$ must be small in norm. If these two assertions are true, then the lower (n-i+1 x n-i+1) block R_{22} in step 9 is small in norm and the desired QR factorization exists. To prove the first assertion, we reproduce the Lemma 3.1 of [Ref. 4]. Chan's derivation for the case of a real valued matrix is adapted here for the complex matrix case.

"Lemma 3.1: Let B ϵ C^{mxk} be a matrix containing any subset of k columns of A. Then

$$\sigma_{min}(B) \equiv \sigma_k(B) \leq \sigma_k(A)."$$

The submatrix R_{11} is the subset of $Q^H A\Pi$, composed of its first i columns. Q^H and Π are orthonormal and therefore their singular values are one. Using this observation and the Lemma 3.1 of [Ref. 4], we have

$$\sigma_{min}(R_{11}) < \sigma_i(Q^H A\Pi) = \sigma_i(A).$$

The above inequality guarantees that R_{11} has a small singular value if $\sigma_i(A)$ is small.

To prove that R_{22} is small in norm, we reproduce here the Lemma 3.2 and Theorems 3.1 and 3.2 of [Ref. 4].

Lemma 3.2: The matrix $W \equiv [w_{n-r+1}, \ldots, w_n] \epsilon C^{n \times r}$ computed by the algorithm RRQR satisfies the *following properties:* For i=n,n-1,...,n-r+1,

1) $\|w_i\|_2=1$,

2) $(w_i)_j=0$ for $j>i$,

3) $|(w_i)_i| = \|w_i\|_\infty \geq 1/\sqrt{i}$,

4) $\|A\Pi w_i\|_2=\delta_i \leq \sigma_i(A)$, where $(\delta_i \equiv \sigma_{min}(R_{11}))_i$.

Theorem 3.1: Let the matrix $W \epsilon C^{nxr}$ as computed by the algorithm RRQR be partitioned as $W^H \equiv (W_1^H, W_2^H)$, where $W^H \epsilon C^{rxr}$ is upper triangular and non-singular. Then the QR factorization of $A\Pi$ as computed by the algorithm RRQR satisfies:

$$\|R_{22}\|_2 \leq \sigma_{n-r+1} \|W_2^{-1}\|_2 \sqrt{r}.$$

Proof: Denote the columns of W by $\{w_{n-r+1}, \ldots, w_n\}$. Define $Y = Q^H A\Pi W \in C^{n \times r}$. From property (4) of Lemma 3.2, we have

$$\|Y\|_2 \le \|Y\|_F = \sqrt{\sum_{i=n}^{n-r+1} \delta_i^2} \le \sqrt{(r)}\,\delta_{n-r+1} \le \sqrt{(r)}\,\sigma_{n-r+1}, \tag{23}$$

where $\delta_i = (\sigma_{\min}(R_{11}))_i$ and $\|.\|_F$ denotes the Frobenius norm. Next the matrix Y can be expressed as:

$$Y = Q^T A\Pi W = RW = \begin{vmatrix} R_{11}W_1 + R_{12}W_2 \\ R_{22}W_2 \end{vmatrix}, \tag{24}$$

which leads to

$$\|Y\|_2 \ge \|R_{22}W_2\|_2 \ge \frac{\|R_{22}\|_2}{\|W_2^{-1}\|_2}. \tag{25}$$

Combining Equations (23), (24) and (25), we get

$$\sigma_{n-r+1} \ge \delta_{n-r+1} \ge \frac{\|R_{22}\|_2}{\|W_2^{-1}\|_2 \sqrt{r}}, \tag{26}$$

from which the desired result follows.■

Theorem 3.2: The algorithm RRQR computes a permutation Π and a QR factorization of A, given by $A\Pi = QR$ where the elements of the lower (r x r) upper triangular block of R satisfy

$$\begin{aligned} |r_{ij}| &\le \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} 2^{j-1-k}\,\sigma_k\sqrt{k} \\ &\le 2^{j-i}\sigma_i\sqrt{n} \quad \text{for } n-r < i \le j \le n \end{aligned} \tag{27}$$

Proof: Using Lemma 3.2, we have, for $n-r<i\le j \le n$

$$\sigma_j \geq \|A\Pi w_j\|_2 = \|Rw_j\|_2 \geq |(Rw_j)_i| = |\sum_{k=i}^{j} r_{ik}(w_j)_k|.$$

Isolating the k=j term in the sum and rearranging terms, we get

$$|r_{ij}(w_j)_j| \leq \sigma_j + \sum_{k=i}^{j-1} |r_{ik}(w_j)_k|.$$

From Lemma 3.2, $|(w_j)_j| = \|(w_j)\|_\infty \geq 1/\sqrt{j}$, we have

$$|r_{ij}| \leq \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} |r_{ik}|.$$

Solving this recurrence in the index j, we get the bound given in the first inequality in Equation 27. Using the bound $\sigma_k\sqrt{k} \leq \sigma_i\sqrt{n}$ in each term of the sum in the desired result, we get the second bound.■

Equation (27) in Theorem 3.2 shows the bounds for the elements of the matrix R generated by the RRQR algorithm. The second inequality states the bounds for the elements of the block R_{22}. The factor 2^{j-i} indicates that one element in R_{22} increases with its distance from the main diagonal. Therefore, for large values of the rank deficiency r, the bounds may be quite large, as it grows exponentially. Numerical simulation cases will be shown later to demonstrate that these bounds are overconservative.

This algorithm was originally proposed to be iterated from n until n-r+1, where n is the number of sensors in the array and r is the noise-free autocorrelation matrix rank deficiency. Alternatively, Prasad and Chandna [Ref. 5] proposed to

iterate the recursion until n-m+1, where m is the number of sources. A faster algorithm is the result of this technique as generally there are fewer sources than sensors. The drawback of this approach, however, is a loss of precision in the estimated signal information, as the algorithm might not capture all the rank deficiency of the matrix to be decomposed. This problem will be investigated in Chapter 4.

B. USING THE RRQR ALGORITHM TO FIND THE NOISE AND SIGNAL SUBSPACES OF THE NOISE-FREE AUTOCORRELATION MATRIX

Assume that we have a linear equispaced array composed of n sensors and m sources, with n > m. The theoretical noise free autocorrelation matrix, (R_s), is r (=n-m) rank deficient. However, the practical noise-free autocorrelation matrix is nearly rank deficient.

It is possible to find the signal and noise subspaces of an incoming signal using the RRQR algorithm shown in Section A. After applying the complete RRQR algorithm, the matrix R reveals the rank deficiency of R_s. Consequently, the norm of R_{22} is small compared to the norm of the rest of the matrix R. Note that R is upper triangular. Here, R_{11} has dimension (n-r(=m) x n-r), R_{12} has dimension ((n-r) x r), and R_{22} has dimension (r x r). The signal subspace is contained in the first m columns, and the noise subspace is contained in the r last columns of the matrix Q. The matrix, W, of the RRQR

algorithm is not used here as an approximation of the noise subspace, as originally proposed by Chan [Ref. 4]. The use of this matrix to approximate the noise subspace yielded poor results, as shown by Fargues [Ref. 3].

C. USING THE GIVENS ROTATION TO REFACTOR $R_{11}P$

A new QR factorization of the $R_{11}P$ term is presented in step six of the RRQR algorithm. In step one, it is reasonable to run a complete QR algorithm for the first QR factorization. But as R_{11} is already upper triangular and P is only a permutation that changes the position of two rows, we can use inclusive Givens Rotations to zero out the few non-zero elements of $R_{11}P$ below its main diagonal. These rotations yield a more efficient algorithm to deal with the special structure of the problem.

The complex Givens Rotation matrix is defined as

$$G=\begin{vmatrix} c & s \\ -conj(s) & c \end{vmatrix}, \tag{31}$$

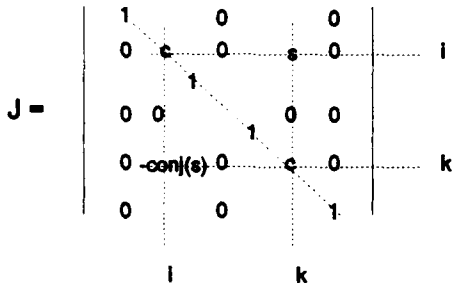
where c is real, s is complex and such that $|c|^2+|s|^2=1$.

The Givens Rotation matrix is defined such that the following equality is satisfied [Ref. 6]:

$$G*\begin{vmatrix} x \\ y \end{vmatrix}=\begin{vmatrix} r \\ 0 \end{vmatrix}. \tag{32}$$

Suppose we have a matrix, A, and we want to zero out a given element a(k,i), below the main diagonal, i.e., k>i. If

we multiply this matrix by the matrix J defined below, we will make the element a(k,i) equal to zero. If we repeat this procedure for all elements of A below the main diagonal and different from zero, the matrix A is transformed into an upper diagonal matrix. This is the essence of the Givens Rotation.



The subroutine GIVENS1 used to compute the complex Givens Rotation matrix J and the subroutine QRGIV used to perform the QR factorization using the Givens rotation may be found in Appendices B and C.

III. METHODOLOGY FOR APPROXIMATING THE SMALLEST RIGHT SINGULAR VECTOR

Chapter 2 presented the RRQR algorithm and indicated that a procedure is needed to find reliable approximations of the smallest right singular vector of a matrix. The inverse iteration method and the Incremental Condition Estimator are introduced to address this problem. The inverse iteration method is used to find the smallest singular vectors of a matrix. It may also be used to find the smallest eigenvector in absolute value. The Incremental Condition Estimator is used to find a reliable approximation of the smallest singular value and the corresponding singular vectors of a matrix.

A. THE INVERSE ITERATION METHOD TO FIND THE SMALLEST SINGULAR VECTOR

Step two of the RRQR algorithm is a computationally expensive part of the procedure [Ref. 7]. It is therefore desirable to find an algorithm that finds the minimum singular value and its corresponding right singular vector quickly and accurately.

The algorithm implemented is the inverse iteration method. Starting with an initial guess, v_0 (in this case a vector composed of ones), the following algorithm is iterated until convergence [Ref. 8].

1. Solve $A\tilde{u}_{i+1}=v_i$ for $\tilde{u}_{i+1}$,

2. Let $u_{i+1}=\tilde{u}_{i+1}/\|\tilde{u}_{i+1}\|_2$,

3. Solve $A^H\tilde{v}_{i+1}=u_{i+1}$ for $\tilde{v}_{i+1}$,

4. Let $v_{i+1}=\tilde{v}_{i+1}/\|\tilde{v}_{i+1}\|_2$.

Let the converged v_i be v_{sv} and compute u_{sv} and σ_{min} by:

$A\tilde{u}=v_{sv}$, $u_{sv}=\tilde{u}/\|\tilde{u}\|_2$, $\sigma_{min}=1/\|\tilde{u}\|_2$.

The above algorithm computes the right (v_{sv}) and left (u_{sv}) singular vectors as well the minimum singular value of the matrix A. This algorithm yields good results for a small number of iterations, as shown by the numerical simulations presented below. Three iterations were used here to estimate the singular vectors.

To evaluate the results achieved with the inverse iteration algorithm, we examined four different test cases. One to three iterations were run on each test case. In each case, the right singular vector was compared to the corresponding vector generated by the SVD decomposition computed with the MATLAB™ software. This procedure was followed for 1000 different random matrices.

Comparisons between approximated and computed singular vectors were obtained by evaluating the magnitude of the projection of the estimate over the true smallest right singular vector. If the two vectors are parallel, the projection is maximum and equal to one. If they are perpendicular, the projection is zero. The projections were

distributed among ten bins, ranging from 0 to 1, as shown in Tables 1-4. The mean and standard deviation for each one of the iteration steps were computed.

The first test case used a 10x10 dimensional random matrix generated using MATLAB™. The second one used a 10x10 random matrix, with the lower triangle was imposed as zero. The third test case, used an upper triangular matrix R, obtained from the QR decomposition without pivoting of a 10x10 random matrix. Finally, the fourth test case used an upper triangular matrix R obtained from the QR decomposition with pivoting of a 10x10 random matrix. These cases were evaluated in terms of performance to verify the adequacy of the method within different and perhaps more demanding contexts. The test results are summarized in Tables 1 through 5.

Table 1: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, RANDOM MATRIX, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations
0 and .1	.6	.1	.1
.1 and .2	1.4	.6	.2
.2 and .3	.7	.2	.1
.3 and .4	.5	.7	.4
.4 and .5	.8	.7	.1
.5 and .6	.9	.4	.3
.6 and .7	1.1	0	.4
.7 and .8	2.4	1.0	.6
.8 and .9	3.9	.7	.8
.9 and 1	87.7	95.6	97.0
Average	.9441	.9756	.9867
Standard Deviation	.1645	.1121	.0802

Table 2: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTOR, RANDOM MATRIX WITH LOWER TRIANGLE PORTION EQUAL TO ZERO, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations
0 and .1	0	0	0
.1 and .2	0	0	0
.2 and .3	.1	0	0
.3 and .4	.1	0	0
.4 and .5	0	.1	0
.5 and .6	.1	0	0
.6 and .7	.1	0	0
.7 and .8	.3	0	0
.8 and .9	.5	.4	.1
.9 and 1	98.8	99.5	99.9
Average	.9960	.9988	.9996
Standard Deviation	.0399	.0184	.0071

Table 3: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/O PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations
0 and .1	.7	.2	.2
.1 and .2	.5	.6	.1
.2 and .3	.7	.2	.4
.3 and .4	.6	.5	.4
.4 and .5	1.2	0	.1
.5 and .6	.6	.1	.1
.6 and .7	.9	.7	.1
.7 and .8	2.0	1.1	.2
.8 and .9	2.8	1.4	.9
.9 and 1	90.0	95.2	97.5
Average	.9528	.9781	.9868
Standard Deviation	.1480	.1044	.0843

Table 4: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/ PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations
0 and .1	0.1	.1	.1
.1 and .2	.6	.3	0
.2 and .3	.5	.1	.3
.3 and .4	.7	.2	.1
.4 and .5	.7	.6	.2
.5 and .6	.7	.6	.4
.6 and .7	.4	.6	.5
.7 and .8	.7	.5	.4
.8 and .9	3.0	.1	.6
.9 and 1	92.6	96.9	97.4
Average	.9660	.9823	.9887
Standard Deviation	.1225	.0922	.0743

Table 5: COMPARISON BETWEEN THE FOUR CASES EXAMINED IN TABLES 1 THROUGH 4.

	Totally Random Matrix	Imposed upper triangular	Resulting from QR decomp. w/o pivoting	Resulting from QR decomp. w/ pivoting
Percentage of projections between .9 and 1.0	97	99.9	97.5	97.4
Average of projections	.9867	.9996	.9868	.9887
Standard Deviation of Projections	.0802	.0071	.0843	.0743

Table 5 shows that the best result was obtained from the imposed triangular matrix experiment. However, all of the experiments show good results. These results justify the use of the inverse iteration method as a low cost alternative to approximate the smallest right singular vector in step two of the RRQR algorithm. The important point is to find a permutation so that the element of the least dominant singular vector that presents the largest magnitude be positioned at the i^{th} row [Ref. 4]. Therefore, it is not necessary to find the exact singular vector for the RRQR algorithm. The source code implemented to approximate the smallest singular vector by the use of inverse iteration method is shown in Appendix J.

B. THE INVERSE ITERATION METHOD TO FIND THE SMALLEST EIGENVECTOR

Prasad [Ref. 5] states that the RRQR-based algorithm works when using an EVD (Eigenvector Decomposition) rather than a SVD (Singular Vector Decomposition) of the noise-free autocorrelation matrix. The validity of this approach is investigated here. Theoretically, it is correct to only replace σ by $|\lambda|$ in the RRQR factorization proof presented in Chapter 2.

The inverse iteration method may be used to find the least dominant eigenvector (the one with a magnitude closer to zero). Again, we find an initial guess v_0, which may be a

vector composed solely of ones. The following algorithm should be iterated until convergence of v.

1. Solve $Ay_k=v_{k-1}$, for y_k.
2. Normalize $v_k=y_k/\|y_k\|_2$.

Comparing the two algorithms, we note that this inverse iteration algorithm uses the same initial two steps as the one used for the minimum singular value. To compare the performance of these two algorithms, the same test cases were run as in previous section. However, six iterations were run to provide a viable comparison. The results are shown in Tables 6 through 9.

Table 6: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST EIGENVECTORS, RANDOM MATRIX, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations	Four Iterations	Five Iterations	Six Iterations
0 and .1	4.0	2.1	2.0	0.6	0.9	0.6
.1 and .2	4.4	2.3	1.3	1.4	1.3	1.2
.2 and .3	3.6	2.6	1.7	1.4	1.8	1.0
.3 and .4	5.3	3.4	2.5	2.3	1.4	0.6
.4 and .5	6.8	4.0	4.0	2.6	2.8	3.0
.5 and .6	7.3	6.7	5.7	4.1	3.4	4.1
.6 and .7	9.3	7.0	5.0	6.8	4.7	6.0
.7 and .8	11.2	11.1	7.5	6.7	7.7	7.5
.8 and .9	17.2	13.0	10.6	11.0	9.6	9.3
.9 and 1	30.9	47.8	59.7	63.1	66.4	66.7
Average	.6964	.7861	.8353	.8586	.8694	.8775
Standard Deviation	.2758	.2491	.2347	.2124	.2116	.1986

Table 7 : INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST EIGENVECTORS, RANDOM MATRIX WITH LOWER TRIANGLE PORTION EQUAL TO ZERO, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations	Four Iterations	Five Iterations	Six Iterations
0 and .1	0.9	0.5	0.4	0.3	0.5	0.3
.1 and .2	0.7	0.7	0.5	0.3	0.4	0.2
.2 and .3	1.5	0.6	0.4	0.2	0.1	0.2
.3 and .4	0.7	0.6	0.4	0.4	0.4	0.2
.4 and .5	1.3	1.0	0.7	1.1	0.5	0.7
.5 and .6	1.9	1.5	0.6	0.9	0.5	0.3
.6 and .7	1.4	1.4	1.1	1.1	0.4	0.9
.7 and .8	3.5	2.9	1.1	0.8	0.9	0.4
.8 and .9	5.7	4.2	2.5	2.6	1.7	2.1
.9 and 1	82.4	86.6	92.3	92.3	94.6	94.7
Average	.9217	.9445	.9634	.9662	.9740	.9774
Standard Deviation	.1787	.1473	.1280	.1201	.1134	.1019

Table 8: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST EIGENVECTOR, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/O PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations	Four Iterations	Five Iterations	Six Iterations
0 and .1	3.1	2.0	0.8	1.0	0.3	1.2
.1 and .2	2.3	1.0	1.5	1.4	1.4	0.9
.2 and .3	3.1	1.3	1.0	1.2	0.9	0.7
.3 and .4	3.3	1.8	1.5	1.2	1.3	0.6
.4 and .5	4.9	3.7	1.8	1.9	1.3	2.0
.5 and .6	6.1	3.5	1.7	1.8	1.2	1.0
.6 and .7	6.9	4.1	3.1	1.1	1.5	0.8
.7 and .8	8.9	6.6	4.4	2.5	2.3	1.8
.8 and .9	17.3	10.6	6.0	6.0	3.9	3.2
.9 and 1	44.1	65.4	78.2	81.9	85.9	87.8
Average	.7654	.8562	.9018	.9158	.9328	.9379
Standard Deviation	.2579	.2227	.1943	.1945	.1728	.1767

Table 9: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST EIGENVECTORS, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/ PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	One Iteration	Two Iterations	Three Iterations	Four Iterations	Five Iterations	Six Iterations
0 and .1	1.3	0.4	0.3	0.0	0.2	0.1
.1 and .2	0.8	0.7	0.4	0.4	0.0	0.1
.2 and .3	1.1	0.5	0.2	0.3	0.0	0.0
.3 and .4	1.5	0.4	0.5	0.2	0.4	0.0
.4 and .5	3.1	0.8	0.4	0.2	0.3	0.2
.5 and .6	3.7	1.2	0.8	0.4	0.3	0.3
.6 and .7	5.6	2.2	0.8	0.7	0.5	0.4
.7 and .8	9.8	3.2	1.7	2.0	1.1	1.0
.8 and .9	16.9	7.1	5.0	3.1	2.6	2.4
.9 and 1	56.2	83.5	89.9	92.7	94.6	95.5
Average	.8451	.9344	.9611	.9734	.9807	.9856
Standard Deviation	.1969	.1432	.1167	.0949	.0803	.0668

The results found in Tables 1 through 4 and 6 through 9 are summarized in Table 10 below. Again, the case for imposed triangular matrices showed the best performance.

Table 10: COMPARISON BETWEEN THE FOUR CASES CONSIDERED FOR THE SVD VERSUS THE EIGENVECTOR INVERSE ITERATION METHOD.

	Totally Random Matrix		Imposed upper tri-angular		Resulting from QR decomp. w/o pivot-ing		Resulting from QR decomp. w/ pivoting	
	SVD	EIGV	SVD	EIGV	SVD	EIGV	SVD	EIGV
Percentage of projections between .9 and 1.0	97.0	66.7	99.9	94.7	97.5	87.8	97.4	95.5
Average of projections	.9867	.8775	.9996	.9774	.9868	.9379	.9887	.9856
Standard De-viation of Projections	.0802	.1986	.0071	.1019	.0843	.1767	.0743	.0668

The SVD columns show the results of three iterations of the inverse iteration method to find the singular vector corresponding to the minimum singular value. The columns EIGV are the results of six iterations of the inverse iteration method to find the least dominant eigenvector.

A cursory look at these results show that the eigenvector approximation is worse than those for the singular vectors approximation. A hypothesis test based on the two samples is therefore tested for each case. An assumption is made that each sample came from different populations with each group of

1000 projections considered to be one sample. The assumptions are

1. The first sample, for the eigenvector case, is a random sample from a population with mean μ_1 and standard deviation σ_1.

2. The second sample, for the SV case, is a random sample from a population with mean μ_2 and standard deviation σ_2.

3. Both samples are independent of one another.

4. The samples are large enough to apply the Central Limit Theorem.

The hypothesis test can be described by [Ref. 9]

$$H_0: \mu_1=\mu_2$$
$$H_1: \mu_1<\mu_2 \quad .$$

One hypothesis test will be conducted for each one of the four test cases. At a level of significance of 1%, H_0 will be rejected if $z \leq -2.33$, where

$$z=\frac{avg(x)-avg(y)}{\sqrt{\frac{s_1^2}{n}+\frac{s_2^2}{n}}} . \qquad (33)$$

If H_0 is false, we may decide that μ_1 is smaller than μ_2. In Equation (33), n is the sample size of 1000, avg(x) and avg(y) are the averages to be compared and s_1 and s_2 are the standard deviations computed from the samples. For example, the first test case avg(x)=.8775, avg(y)=.9867, s_1=.1986 and s_2=.0802 from Table 10. Applying these values to Equation (33), we find z=-16.12.

Table 11: RESULTS FOR THE HYPOTHESIS TEST CONDUCTED ON THE COMPARISON BETWEEN THE SVD VERSUS THE EIGENVECTOR INVERSE ITERATION METHOD.

Test Case	Value of z	Conclusion
1	-16.12	Reject H_0 => $\mu_1 < \mu_2$
2	-6.87	Reject H_0 => $\mu_1 < \mu_2$
3	-7.90	Reject H_0 => $\mu_1 < \mu_2$
4	-.98	Cannot Reject H_0

As can be seen in Table 11 above, the first average μ_1 is significantly smaller than μ_2 (at 1% of level of significance, $z<-2.33$), for all cases but case four. Hence, we reject H_0 for the first three cases. These results justify the choice of using the singular vector approximated by the inverse iteration method for use with the algorithm. Note that there is no theoretical reason for not using the smallest eigenvector rather than the smallest right singular vector in the RRQR algorithm. They span the same subspace. The reasons for choosing the singular vector relies solely on the fact that the inverse iteration method yields better results. The source code implemented to approximate the smallest eigenvector by the inverse iteration method is shown in Appendix M.

C. THE INCREMENTAL CONDITION ESTIMATOR

A third method tested here to estimate the singular vector corresponding to the minimum singular value is the Incremental Condition Estimator (ICE) [Ref. 10]. Suppose that

$A=[a_1,\ldots,a_n]$ is a (m x n) dimensional matrix and let $\sigma_1 \geq \ldots \geq \sigma_{min} \geq 0$ be its singular values. The minimum singular value of A measures how close this matrix is to rank deficiency. The condition number becomes

$$k_2(A) \equiv \frac{\sigma_1}{\sigma_{min}}. \tag{34}$$

Now, suppose we take the QR factorization of A. This algorithm is intended to work from a lower triangular matrix L. Since R is upper triangular let us define $L=R^T$. There will be no loss of generality here because the singular values of any matrix and its transpose are the same [Ref. 10].

If we have a n dimensional lower triangular matrix L generating one row at a time with an approximate singular vector x such that $\sigma_{min}(L) \approx 1/\|x\|_2$ and a new row (v^T,γ) of L , we can obtain

$$L' \equiv \begin{vmatrix} L & 0 \\ v^T & \gamma \end{vmatrix} \tag{35}$$

such that $\sigma_{min}(L') \approx 1/\|y\|_2$ without having to access L again [Ref. 10].

Given x such that Lx=d with $\|d\|_2=1$ and $\sigma_{min}(L) \approx 1/\|x\|_2$, let us find $s=\sin(\phi)$ and $c=\cos(\phi)$ such that $\|y\|_2$ is maximized, where y solves

$$L'y = \begin{vmatrix} L & 0 \\ v^T & \gamma \end{vmatrix} y = \begin{vmatrix} sd \\ c \end{vmatrix} = d' . \tag{36}$$

The solution for the above equation is

$$y = \begin{vmatrix} sx \\ \frac{c-s\alpha}{\gamma} \end{vmatrix}, \tag{37}$$

where $\alpha=v^Tx$ and $\|d'\|_2=\|d\|_2=1$.

Now define

$$B = \begin{vmatrix} 1+\beta & -\alpha \\ -\alpha & 1 \end{vmatrix}, \tag{38}$$

where $\beta=\gamma^2x^Tx+\alpha^2-1$. In this case, we have

$$\|y\|_2^2 = \frac{1}{\gamma^2}\begin{vmatrix} s & c \end{vmatrix} B \begin{vmatrix} s \\ c \end{vmatrix}. \tag{39}$$

Assuming $\gamma\neq0$, the optimal pair $(s,c)^T$ is the eigenvector corresponding to the largest eigenvalue λ_{max} of B [Ref. 10]. Also assuming $\alpha\neq0$, we may define

$$\eta=\frac{\beta}{2\alpha} \text{ and } \mu=\eta+sign(\alpha)\sqrt{\eta^2+1} \tag{40}$$

to obtain $\lambda_{max}=\alpha\mu+1$. Finally the optimal pair $(s,c)^T$ is given as

$$\begin{vmatrix} s \\ c \end{vmatrix} = \frac{1}{\sqrt{\mu^2+1}}\begin{vmatrix} \mu \\ -1 \end{vmatrix}. \tag{41}$$

After computing the optimal pair $(s,c)^T$, a new approximate singular vector, y, may be computed as defined in Equation (37) and the smallest singular value of L' by

$$\sigma_{min}(L') \approx \frac{1}{\|y\|_2}. \qquad (42)$$

Using the above estimator we ran the same cases as for the inverse iteration presented in Sections A and B above. The results are summarized in Tables 12 through 15.

Table 12: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, RANDOM MATRIX, 1000 TRIALS.

Percentage of Projection lying between	
0 and .1	0.6
.1 and .2	0.2
.2 and .3	0.5
.3 and .4	1.1
.4 and .5	0.7
.5 and .6	0.6
.6 and .7	0.9
.7 and .8	2.1
.8 and .9	5.6
.9 and 1	87.7
Average	.9015
Standard Deviation	.2041

Table 13: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, RANDOM MATRIX WITH LOWER TRIANGLE PORTION EQUAL TO ZERO, 1000 TRIALS.

Percentage of Projection lying between	
0 and .1	0.0
.1 and .2	0.0
.2 and .3	0.0
.3 and .4	0.0
.4 and .5	0.2
.5 and .6	0.3
.6 and .7	0.3
.7 and .8	0.3
.8 and .9	0.5
.9 and 1	98.4
Average	.9929
Standard Deviation	.0441

Table 14: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/O PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	
0 and .1	0.3
.1 and .2	0.3
.2 and .3	0.4
.3 and .4	0.1
.4 and .5	0.5
.5 and .6	0.5
.6 and .7	0.9
.7 and .8	0.7
.8 and .9	2.7
.9 and 1	93.6
Average	.9705
Standard Deviation	.1079

Table 15: INNER PRODUCT MAGNITUDE BETWEEN ESTIMATED AND TRUE SMALLEST SING. VECTORS, UPPER TRIANGULAR MATRIX RESULTING FROM QR FACT. W/ PIVOTING, 1000 TRIALS.

Percentage of Projection lying between	
0 and .1	0.3
.1 and .2	0.2
.2 and .3	0.0
.3 and .4	0.2
.4 and .5	0.1
.5 and .6	0.1
.6 and .7	0.2
.7 and .8	0.7
.8 and .9	1.3
.9 and 1	96.9
Average	.9859
Standard Deviation	.0797

Table 16 summarizes the results found for each one of the four test cases, comparing the inverse iteration method to find the smallest right singular vector and the Incremental Condition Estimator. As before, those numbers represent the percentage of projections lying between .9 and 1 of the corresponding vectors found via the corresponding approximation method over the true smallest right singular vector found via SVD.

Table 16: COMPARISON BETWEEN THE FOUR CASES CONSIDERED FOR THE SVD INVERSE ITERATION VERSUS THE INCREMENTAL CONDITION ESTIMATOR METHOD.

	Totally Random Matrix		Imposed upper tri-angular		Resulting from QR decomp. w/o pivot-ing		Resulting from QR decomp. w/ pivoting	
	INV. ITER.	ICE	INV. ITER.	ICE	INV. ITER.	ICE	INV. ITER.	ICE
Percentage of projections between .9 and 1.0	97.0	87.7	99.9	98.4	97.5	93.6	97.4	96.9
Average of projections	.9867	.9015	.9996	.9929	.9868	.9705	.9887	.9859
Standard De-viation of Projections	.0802	.2041	.0071	.0441	.0843	.1079	.0743	.0797

Table 16 shows the magnitude of the projections of the estimated *smallest right singular* vectors obtained using inverse iteration (INV. ITER.) method and the ICE, onto the smallest singular vector computed via EVD.

A hypothesis test was again performed on these results. Table 17 presents the results obtained for the hypothesis test.

Table 17: RESULTS FOR THE HYPOTHESIS TEST CONDUCTED ON THE COMPARISON BETWEEN THE SV INVERSE ITERATION AND THE INCREMENTAL CONDITION ESTIMATOR.

Test Case	Value of z	Conclusion
1	-12.29	Reject H_0 => μ_1 (INV.ITER.) < μ_2 (ICE)
2	-4.74	Reject H_0 => $\mu_1 < \mu_2$
3	-3.76	Reject H_0 => $\mu_1 < \mu_2$
4	-.81	Cannot Reject H_0

Tables 12 through 17 show that the results using the incremental condition estimator are not as good as those for the SV inverse iteration. Therefore, the inverse iteration used to find the least dominant singular vectors is preferred. The source code implemented to approximate the smallest singular value and its corresponding singular vectors via ICE is shown in Appendix I.

IV. COMPARISON BETWEEN THE PERFORMANCE OF THE RRQR ALGORITHM ITERATED FROM DIMENSION n UNTIL n-m+1 AND FROM DIMENSION n UNTIL n-r+1

Prasad [Ref. 5] states that the RRQR-based algorithm may be used to estimate the DOA information when the algorithm is iterated from n until n-m+1 rather than until n-r+1 as in Chan's work [Ref. 4]. The relative efficiency of the RRQR algorithm using both approaches is compared by running 1000 trials. The scenario is m=2 fixed sources at 30° and 32° with n=10 sensors. Therefore, the theoretical noise-free autocorrelation matrix is of size 10 x 10 and has rank two (m=2). The near rank deficiency is (r=n-m) 8. Three SNR cases are tested (-10, 0 and 10 dB) for each one of the situations, resulting in six simulations.

In the first situation, n-r+1 equals three. In the second, n-m+1 equals nine. It is intuitively obvious that the second one is much faster than the first, as it will be iterated only twice, from ten to nine. On the other hand, the smaller number of iterations, the less probable that the upper triangular matrix R will capture the near rank deficiency of the noise-free autocorrelation matrix (R_s).

For each run, the largest principal angle between the signal subspace computed via the eigenvector decomposition and the approximated signal subspace computed via the RRQR algorithm is used for comparison. The same is done for the

noise subspace. A vector of ones is expected if the subspace generated by the RRQR algorithm is parallel to its corresponding subspace generated by the true eigenvector. In the case that the two subspaces are perpendicular, a vector of zeros is expected.

Recall that the cosines of the principal angles between two subspaces F and G are defined as the singular values of the product $Q_F^T Q_G$ [Ref. 7], where the matrices Q_F and Q_G are the orthonormal matrices obtained from F and G. To find the SVD decomposition of this product, the inverse cosine of the singular values is taken using the largest angle. The largest angle represents a measure of the distance between the two subspaces. This measure is used for noise and signal subspaces.

Tables 18-20 present means and standard deviations obtained for the largest principal angle for signal and noise subspaces using the RRQR algorithm where the first QR decomposition, in step 0, is performed with pivoting. This computation is done for SNR -10, 0 and 10 dB.

Table 18: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=-10 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/ PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	46.80	13.65	47.05	14.11
Noise Subspace	46.80	13.65	47.05	14.11

Table 19: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=0 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/ PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	17.53	5.71	26.95	12.55
Noise Subspace	17.53	5.71	26.95	12.55

Table 20: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=10 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/ PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	2.24	0.69	4.85	4.09
Noise Subspace	2.24	0.69	4.85	4.09

Tables 18-20 show that the noise and signal subspaces yield identical means and standard deviations. This was to be expected as the noise and signal subspaces contain the same information.

Table 21 shows the results obtained for the hypothesis test performed to compare the two methods of iteration.

Table 21: RESULTS FOR THE HYPOTHESIS TEST CONDUCTED ON THE COMPARISON BETWEEN THE RRQR ALG. ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1. QR DEC. IN STEP 0 W/ PIVOTING

SNR value in dB	Value of z	Conclusion
-10	-.40	Cannot reject H_0
0	-21.6	Reject H_0 => $\mu_1 < \mu_2$
10	-19.9	Reject H_0 => $\mu_1 < \mu_2$

The results show that for SNR 0 and 10 dB, the null hypothesis is rejected. Therefore, the larger number of iterations yield better results. However, for the case of -10 dB the results are meaningless due to the small signal to noise ratio. They do not lead to detection of the signal embedded in the noisy environment.

Tables 22-24 present means and standard deviations obtained for the largest principal angle for signal and noise subspaces using the RRQR algorithm where the first QR decomposition, in step 0, is performed without pivoting. An hypothesis test is not necessary in this case due to the clear difference in results found for the different method of iterations.

Table 22: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=-10 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/O PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	46.76	13.58	59.59	13.79
Noise Subspace	46.76	13.58	59.59	13.79

Table 23: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=0 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/O PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	17.45	5.70	50.81	23.62
Noise Subspace	17.45	5.70	50.81	23.62

Table 24: COMPARISON BETWEEN THE RRQR ALGORITHM ITERATED FROM n THROUGH n-r+1 AND FROM n THROUGH n-m+1 FOR SNR=10 dB. QR DECOMPOSITION IN STEP 0 PERFORMED W/O PIVOTING, 1000 TRIALS.

	Iteration from n=10 until n-r+1=3 (# of flops=1,301,732) - Angle in degrees.		Iteration from n=10 until n-m+1=9 (# of flops=837,515) - Angle in degrees.	
	Mean (μ_1)	Std Dev	Mean (μ_2)	Std Dev
Signal Subspace	2.21	0.63	14.47	8.72
Noise Subspace	2.21	0.63	14.47	8.72

Tables 22 to 24 show that better performance is obtained when using a larger number of iterations. Note that no difference in performance is found using a QR decomposition with or without pivoting in step 0 of the RRQR algorithm in the case using more iterations. This result is not true for the second case. This is clear when Tables 18-20 are compared to their correspondent Tables 22-24. A hypothesis test is not needed to verify this, due to the proximity of the results. Therefore, the QR decomposition without pivoting is preferred. This method is less computationally intensive when the original algorithm is used, performed with the total number of iterations. Note that the option using only two iterations does not present the same performance regarding the pivoting. Therefore, if one chooses this option, care should be exercised when evaluating the pivoting needs.

V. COMPARING THE PERFORMANCE OF THE MINIMUM NORM AND MUSIC SPECTRAL ESTIMATORS

This Chapter investigated the computation of estimates of the direction of arrival of signals obtained using the RRQR. Two high-resolution techniques, the MUSIC (Multiple Signal Classification) [Ref. 2] and the Minimum Norm [Ref. 11] are evaluated to verify their adequacy when used with the RRQR algorithm for DOA estimation.

Rao [Ref. 12] shows that the Mean Square Error (MSE) of the Minimum Norm estimator is smaller than the MSE of the MUSIC estimator. The MUSIC spectral estimator is based on the orthogonality principle. Therefore, the principal eigenvectors $\{v_1, v_2, \ldots, v_m\}$ span the same subspace as the signal vectors $\{e_1, e_2, \ldots, e_m\}$ [Ref. 2]. Thus, the signal vectors are orthogonal to all vectors in the noise subspace. The power density corresponding to the sources DOA information is given by:

$$P_{MUSIC}(\theta) = \frac{1}{\sum_{j=m+1}^{n} |e^H(\theta) \cdot v_j|^2} \qquad (43)$$

where v_j is the singular vectors of the autocorrelation matrix, m is the number of signals, n is the number of sensors or the size of R_s, and $e(\theta)$ is the mode vector as defined in (4).

The angle θ is varied in fine steps from 0 to 2π. When it corresponds to one of the source DOA angles, $e(\theta)$ will be equal to e_i, i=1,2,..,m. Since the sum in the denominator of (43) is over m+1 through n, we have the singular vectors corresponding to the noise subspace. By the orthogonality principle, the product in the denominator of P_{MUSIC} will tend to zero and P_{MUSIC} will tend to infinity [Ref. 2]. The result is a peak at the source DOA angles.

The Minimum Norm estimator is based on the estimation of θ_i, the source DOA angles, from the eigenstructure of the autocorrelation matrix. Suppose we have a vector d so that $d=[d1,d2,\ldots,d_n]$, where n is the number of sensors in the array. If this vector has the property that $x_i^H d=0$, i=1,2,...,m, where m is the number of sources present and x the data snapshot vector at instant i, then we can find a polynomial D(z) as

$$D(z)=\sum_{i=0}^{n} d_i z^{-i}, \qquad (44)$$

so that the zeros of the polynomial lie at the elements of the mode vector corresponding to the source DOA angles.

The polynomial roots corresponding to the DOA angle locations lie on the unit circle for the autocorrelation matrix when additive noise is not present. However, the m roots of the polynomial D(z), corresponding to the m sources,

lie near but not on the actual unit circle when the estimated noise free autocorrelation matrix R_s is used.

The Minimum Norm Spectral Estimator presents the following advantages:

1. The estimates of θ_i, the source DOA angles, are more accurate even at relative low SNR values when compared with other procedures.

2. The n-m extraneous zeros of D(z) tend to be uniformly distributed within the unit circle and have less tendency for false sources. [Ref. 11]

By less tendency to false sources we mean that the zeros of D(z) corresponding to noise are much smaller in magnitude than the ones corresponding to sources. The method imposes d_1, the first element of vector d, to be equal to 1 and requires that the quantity Q in Equation (45) be minimum.

$$Q=\sum_{k=1}^{n} |d_k|^2 \qquad (45)$$

The effect of this minimization forces the extraneous n-m zeros to be uniformly distributed inside the unit circle [Ref. 11].

Next, let E_s be the matrix constructed with the signal subspace generated by EVD or SVD. We partition E_s as follows:

$$E_s=\begin{vmatrix} g^T \\ \hat{E}_s \end{vmatrix} \tag{46}$$

where $g=[e_{11}, e_{21}, \ldots, e_{m1}]^T$ has the first elements of the signal subspace eigenvectors and $\hat{E}_s$ is the matrix E_s with the first row deleted. It can be shown that in order to satisfy the desired minimization and forcing $d_1=1$ [Ref. 11], we have:

$$d=\begin{vmatrix} 1 \\ -\hat{E}_s g^* / (1-g^H g) \end{vmatrix} \tag{47}$$

Once the vector d, representing the coefficients of the polynomial D(z), is determined via (47), the roots are computed. There are m roots corresponding to the sources that present a large magnitude compared to the ones corresponding to noise. These roots reveal the desired DOA angles in their phase angles.

Results for both, MUSIC and Minimum Norm spectral estimators are shown in Figures 1-4, for two fixed sources at 30° and 32°. It can be seen from these examples that the Minimum Norm spectral estimator starts to resolve the two sources for a SNR of 5 dB, while MUSIC starts to resolve only for a SNR of 7 dB. This agrees with the results shown in [Ref. 12].

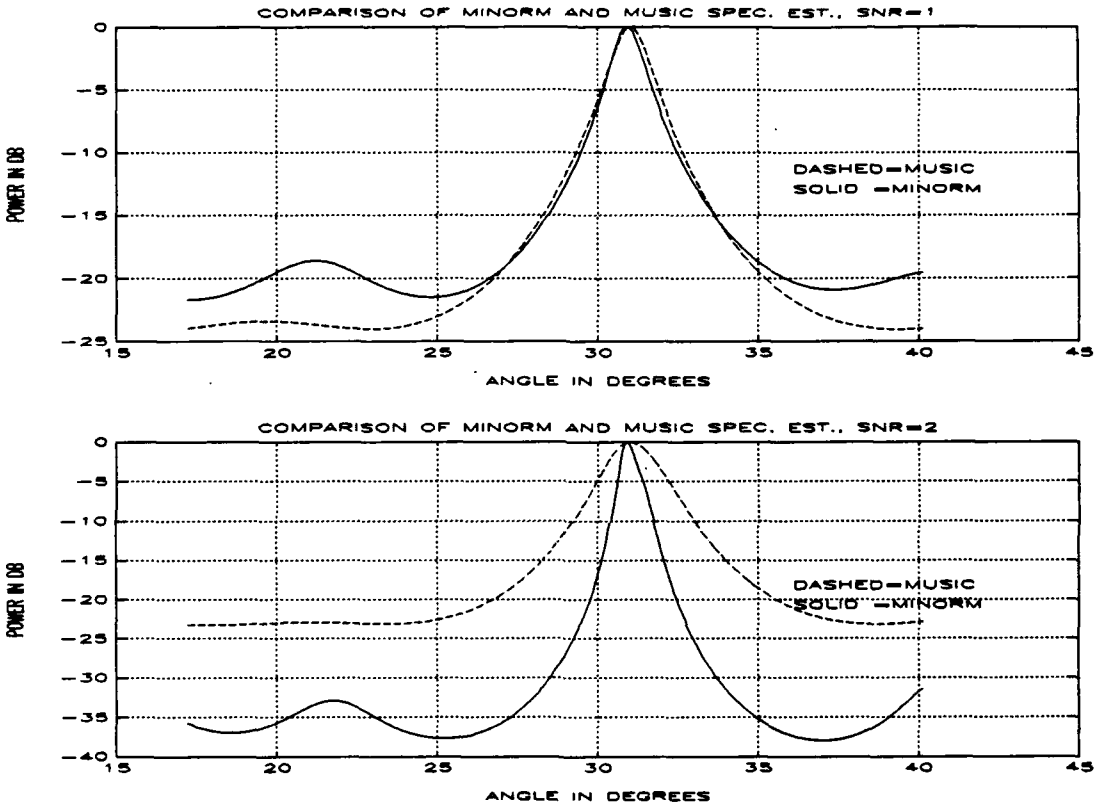


Figure 1: Comparison between the DOA estimated by MUSIC and MINNORM for two standing sources located at 30° and 32° for SNR=1 and 2 dB.

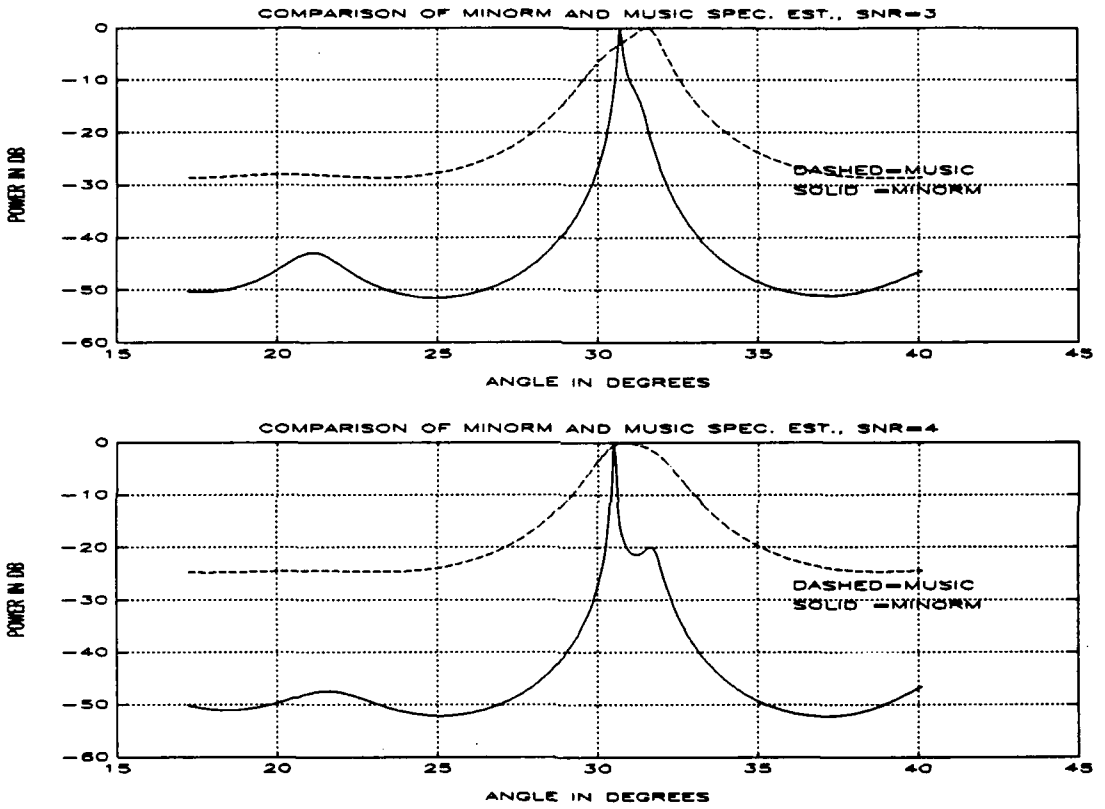


Figure 2: Comparison between the DOA estimated by MUSIC and MINNORM for two standing sources located at 30° and 32° for SNR=3 and 4 dB.

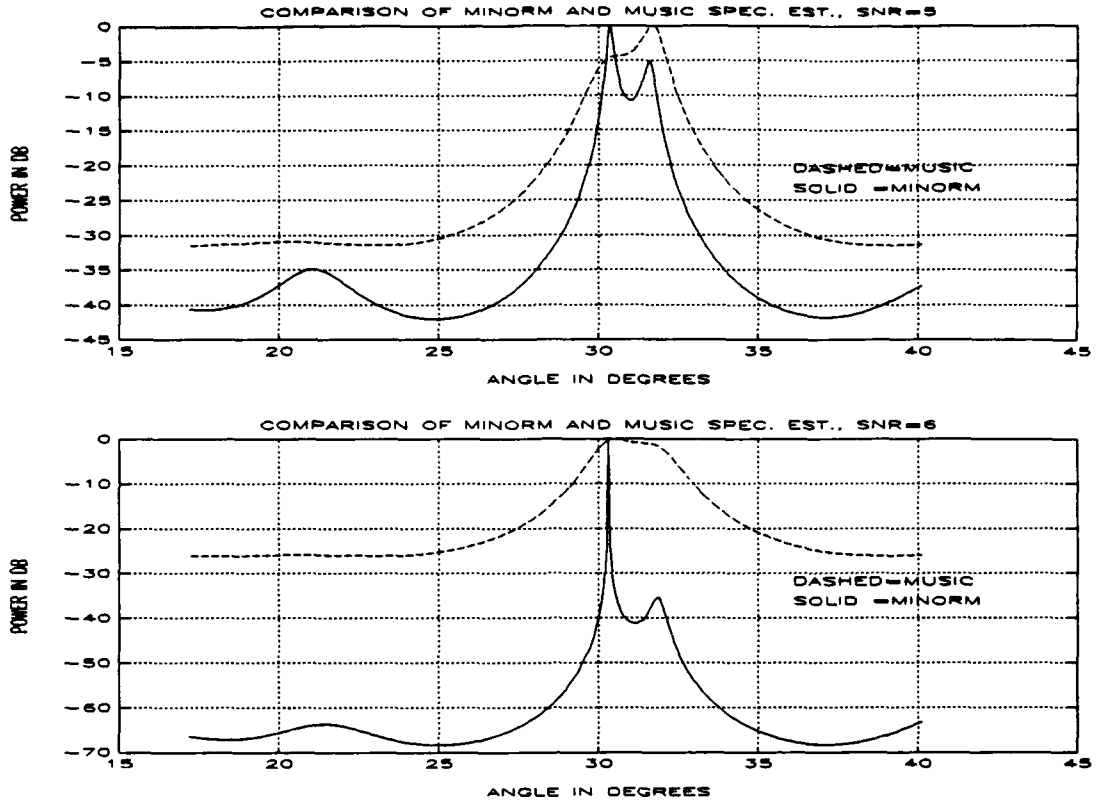


Figure 3: Comparison between the DOA estimated by MUSIC and MINNORM for two standing sources located at 30° and 32° for SNR=5 and 6 dB.

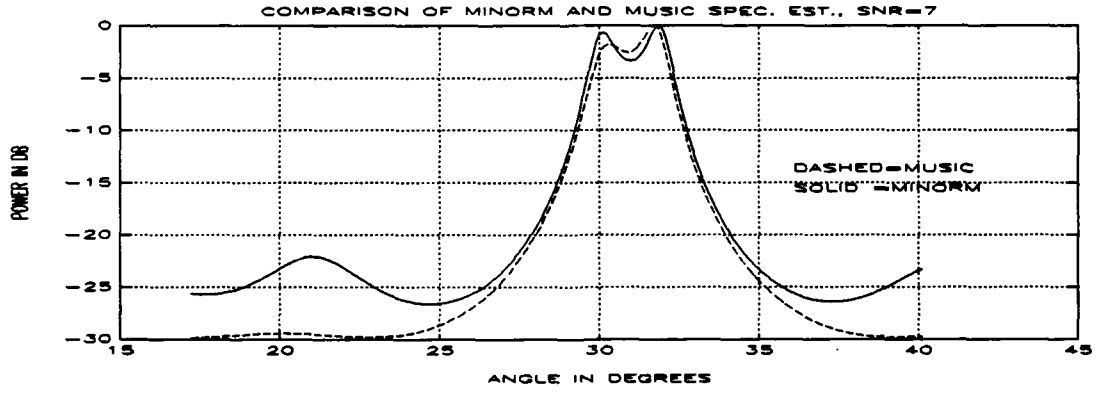


Figure 4: Comparison between the DOA estimated by MUSIC and MINNORM for two standing sources located at 30° and 32° for SNR=7 dB.

VI. USING THE ADAPTIVE RRQR ALGORITHM TO TRACK A MOVING SIGNAL

In a real case, we are interested in being able to detect and track a moving source. In this case, the moving source information was sampled every t_s interval to provide an update of the source information. The array of sensors will receive the signals from the sources leading to the data vector $x=[x_1, x_2, \ldots, x_n]$. From vector x, we compute the autocorrelation matrix, R_{xx}, from (7) and form the noise-free autocorrelation matrix, R_s, by subtracting the noise information, $\sigma^2 I$. Next we apply the RRQR algorithm for each update to identify the signal or noise subspaces. Finally, the Minimum Norm estimator may be applied which leads to the identification of the m source DOA angles from the n-m extraneous noise zeros. This algorithm is presented in this chapter.

A. THE ALGORITHM

In order to track the DOA of a moving signal, a noise-free autocorrelation matrix must first be generated. Next, we compute a first RRQR factorization and find the matrices Q, R and Π. The noise-free autocorrelation matrix, $R_s=R_{xx}-\sigma^2 I$, may then be updated for each one of the next signal positions in time, according to Equation (48) below

$$R_{xx}^{new}=R_{xx}^{old}+x\cdot x^{H}|_{new}-x\cdot x^{H}|_{old} \qquad (48)$$

where x is a (n x 1) dimensional vector containing the input signals at each one of the n array sensors. The update of the noise-free autocorrelation matrix is achieved using a moving window where the number of snapshots used to compute the noise-free autocorrelation matrix is constant. The new vector x is incorporated in the autocorrelation matrix information for each snapshot update, while the oldest snapshot information is discarded.

A possible approach in updating the information is to find the updated noise-free autocorrelation matrix using (48). Next we apply a QR decomposition followed by Chan's [Ref. 4] pivoting scheme, i.e., a complete RRQR algorithm.

However, it is possible to update directly the existing QR factorization [Ref. 3] for each new time sample. Suppose we want to add a rank-one matrix $C=x\cdot x^{H}$ to the matrix R_{xx}^{old}, whose QR factorization is known as $QR\Pi^{T}$. The new matrix $R_{xx}^{new}=R_{xx}^{old}+x\cdot x^{H}$ will have a QR factorization $Q_1R_1\Pi^{T}$, where $x\cdot x^{H}$ is the desired rank-one modification.

The rank-one QR factorization update may be computed as follows:

$$R_{xx}^{new} = R_{xx}^{old}+x\cdot x^{H} = Q(R+Q^{H}x\cdot x^{H}\Pi)\Pi^{T} = Q_1R_1\Pi^{T}$$

Let $w=Q^{H}\cdot x$. Complex Givens rotations can be used to zero out all elements of w except for the first component, leading to

$$J_1^H \ldots J_{n-1}^H w = [a\ 0\ 0\ \ldots\ 0]^T \qquad (49)$$

If the same Givens rotations are applied to R, it can be shown that $H=J_1^H \ldots J_{n-1}^H R$ is upper Hessenberg. Consequently, we have $(J_1^H \ldots J_{n-1}^H)(R_{xx}^{old}+w \cdot x^H)=H+[a \quad 0 \quad \ldots \quad 0]^T \cdot x^H=H_1$, also upper Hessenberg. [Ref. 7]

Next, we use complex Givens rotations to compute $G_1^H \ldots G_{n-1}^H H_1=R_1$, where R_1 is upper triangular. Combining all of the above, we have the desired new QR factorization $R_{xx}^{new}=R_{xx}^{old}+x \cdot x^H=Q_1 \cdot R_1 \cdot \Pi^T$, where

$$Q_1=Q \cdot J_{n-1} \ldots J_1 \cdot G_1 \ldots G_{n-1}. \qquad (50)$$

The reader should refer to Golub [Ref. 7] for additional detail.

We apply two successive rank-one modifications to update the noise-free autocorrelation matrix for the current time sample, finding a new set of matrices Q, R and Π. The first rank-one modification incorporates the new data vector to the autocorrelation matrix. The second accounts for removing the old information. Then, we apply Chan's [Ref. 4] pivoting scheme to insure a correct estimation of the noise and/or signal subspaces.

Next, we identify the signal and noise subspaces. The first m columns of Q constitute the signal subspace, where m is the number of sources present. The last n-m=r columns constitute the noise subspace, where n is the number of

sensors. Note that it is usually more efficient to use the signal subspace, rather than the noise subspace as it is smaller. Finally, the Minimum Norm algorithm is applied to estimate the source locations.

A total of m out of the n roots of the polynomial whose coefficients form the vector "d" in the MINORM algorithm corresponds to the source locations. It is expected that the m roots corresponding to the sources lie near the unit circle and the remaining ones have smaller magnitudes. Some sort of filtering must be applied to separate the m source zeros and the remaining n-m extraneous zeros. Filtering may be achieved either by sorting the expected range of source angles and/or by sorting the magnitude. Thus, the algorithm corresponding to the adaptive case becomes [Ref. 3]:

1) Initialization

$R_s = X \cdot X^H - w\sigma^2 I$ (note that the noise-free autocorrelation matrix is unnormalized) where X is defined in Equation (7), Chapter 1 and w is the number of snapshots used to compute the correlation matrix.

RRQR factorization of R_s ($R_s\Pi = QR$)

2) Start updating. For k=1 to number of updates do:

a) $R_s(k+1) = R_s(k) + x(k+1) \cdot x^H(k+1) - x(k+1-w) \cdot x^H(k+1-w)$.

b) Update the above QR factorization applying two successive rank-one modifications to $R_s(k)$, leading to: $R_s(k+1) = Q_1 R_1 \Pi^T$ (Alternatively we may find a new complete RRQR decomposition of the updated noise-free autocorrelation matrix. In this case, the next two steps should be skipped).

c) Apply Chan's Pivoting scheme (steps 1 through 9 of the RRQR algorithm) to Π, Q_1 and R_1, finding Π_2, Q_2 and R_2.

d) Let $\Pi=\Pi_2$, $Q=Q_2$ and $R=R_2$.

e) Identify the signal or noise subspace (Option to use a refinement as explained in Section B below).

f) Apply the Minimum Norm to find the estimated source angles.

g) Filter and store the source angles.

The source code implemented to generate the autocorrelation matrix is shown in Appendices K and L. The source code implemented to generate the adaptive algorithm is shown in Appendix D and the code corresponding to the rank-one modification shown in Appendix E. Appendix F presents the Minimum Norm algorithm. Finally, Appendix H presents the Minimum Norm identification procedure needed to isolate the signal source locations.

The adaptive algorithm presented above is an alternative to the identification of the signal/noise subspaces via SVD or EVD decomposition. Note that steps 4 to 6 in Chan's algorithm are only needed when the element of maximum magnitude of the smallest right singular vector is not in the last position. Thus, additional reduction in computation time is obtained when no pivoting is needed.

Next, we investigate how often pivoting is needed in Chan's algorithm. To test that effect, we ran two test cases with two sources (m=2). One of the sources is fixed, the other is time varying. Ten sensors are used to compute the

correlation matrix with 100 snapshots used to form the noise-free autocorrelation matrix and 200 updates are computed. The value of the SNR varies to try to identify correlation between the SNR and the need for pivoting during the RRQR decomposition (step 4 of Chan's algorithm). In our test cases, if the Chan's pivoting scheme were applied blindly, steps 4 through 9 would be executed for a total of 1600 times. (Because n-r+1=10-8+1=3 and the algorithm is iterated from n to n-r+1, a total of eight times for each one of the updates is needed. Since we have 200 updates in our test case, we get 8*200=1600).

First, we update Q and R directly using two successive rank-one modifications. Next, we update the noise-free autocorrelation matrix using Equation (48) and perform a new complete RRQR decomposition (steps 0 through 9). Figures 5 and 6 show the percentage of times that a pivoting is needed out of the 1600 tests as a function of the SNR (dB) of the source.

Figure 5 shows that the percentage of pivoting steps needed lies between 18% and 32% when two successive rank-one modifications are used to update the noise-free correlation matrix. This means that no pivoting is needed for every iteration.

For the second test case, where a complete RRQR algorithm is applied in the updated noise-free autocorrelation matrix, the percentage of pivoting steps needed remains between 70%

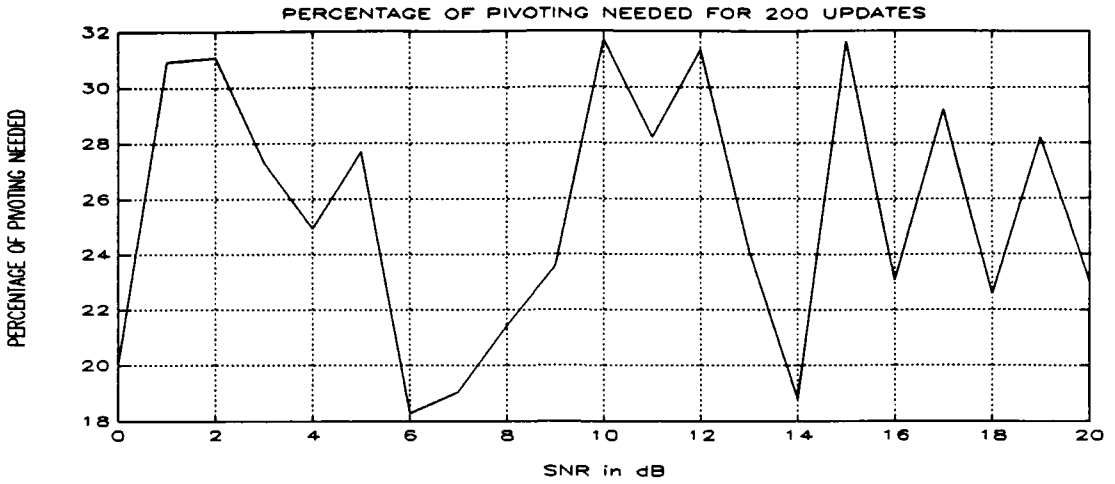


Figure 5: Percentage of times that a pivoting is needed out of 1600 iterations on the adaptive algorithm, using two rank-one modifications.

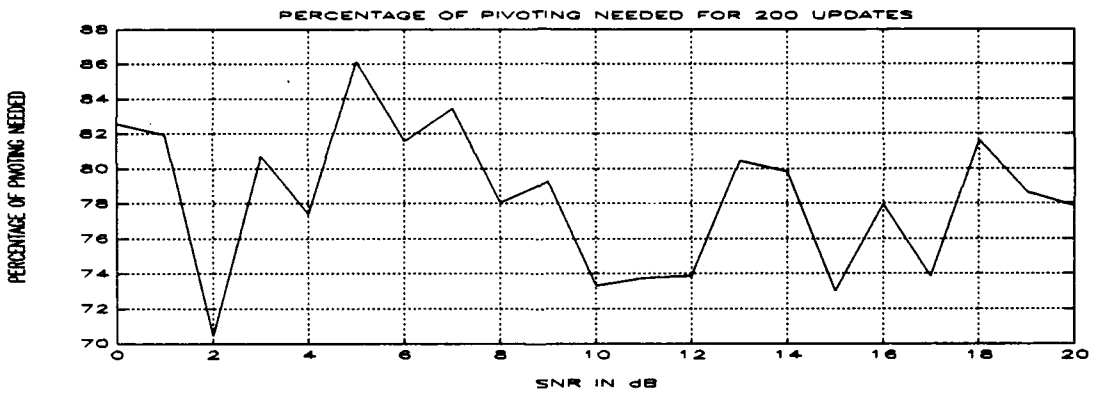


Figure 6: Percentage of times that pivoting is needed out of 1600 iterations on the adaptive alg., updating the noise-free correlation matrix and applying a complete RRQR algorithm.

and 88%. The two successive rank-one modifications are more computationally expensive than the complete RRQR algorithm. On the other hand, they require less pivoting. In Section D below, we analyze the implications of this on the processing time for the two approaches. Furthermore, Figures 5 and 6

indicate that there is no correlation between the magnitude of the SNR and the need for pivoting.

B. REFINEMENT ON THE RESULTS OF THE ADAPTIVE CASE

This section presents an improvement to the signal subspace estimation procedure which is applied to the adaptive algorithm. The basic idea behind the improvement is based on the fact that the noise-free autocorrelation matrix, $R_s=QR\Pi^T$ is Hermitian, and therefore $R_s=R_s^H$. So, $R_s^H=\Pi R^H Q^H$. Recall that the signal and noise subspaces, Q_s and Q_n are contained in the matrix Q. Thus,

$$R_s^H Q_s = \Pi R^H \begin{vmatrix} Q_s^H \\ Q_n^H \end{vmatrix} \begin{vmatrix} Q_s 0 \end{vmatrix} = \Pi R^H \begin{vmatrix} I_s & 0 \\ 0 & 0 \end{vmatrix} = \Pi \begin{vmatrix} R_{11}^H & 0 \\ R_{12}^H & R_{22}^H \end{vmatrix} \begin{vmatrix} I_s & 0 \\ 0 & 0 \end{vmatrix} = \Pi \begin{vmatrix} R_{11}^H & 0 \\ R_{12}^H & 0 \end{vmatrix}, \quad (51)$$

and therefore

$$R_s Q_s = \begin{vmatrix} \Pi_1 & \Pi_2 \end{vmatrix} \begin{vmatrix} R_{11}^H & 0 \\ R_{12}^H & 0 \end{vmatrix} = \Pi_1 R_{11}^H + \Pi_2 R_{12}^H. \quad (52)$$

The matrix resulting by the product R_sQ_s may be viewed as an one-step subspace iteration applied to the signal subspace Q_s. This iteration scheme improves the results obtained as shown in the next section. The drawback is that the resulting iterated signal subspace R_sQ_s is no longer orthonormal. An additional orthonormalization step needs to be applied to the iterated signal subspace in order to use the Minimum Norm algorithm. Note that no reorthonormalization is needed when the MUSIC estimator is used.

C. SIMULATION RESULTS

We now present the results generated by the simulation carried out for SNRs of 6, 10 and 20 dB. The noise is assumed to be zero-mean Gaussian and uncorrelated from sensor to sensor. We consider the case of two sources impinging on the ten-element array. The first source is assumed to be fixed at a normalized angle θ_1=40°, the second source location θ_2 is linear time-varying. Movement starts at 30° and stops at 22.5°, after 100 snapshots are used to form the correlation matrix and 200 updates are used to simulate the movement.

Figures 7, 8 and 9 present the estimated DOA information obtained using the Eigenvector decomposition (EVD), the initial RRQR approximation, and the "refined" RRQR algorithm. The EVD decomposition is used for convenience instead of the SVD decomposition, as both span the same subspace. The results obtained for the refined RRQR technique are nearly identical to those obtained using the EVD technique. Table 25 below shows means and standard deviations for the magnitude of the difference between the RRQR approximation for the signal DOA angle in degrees with/without refinement and the DOA generated by the EVD decomposition.

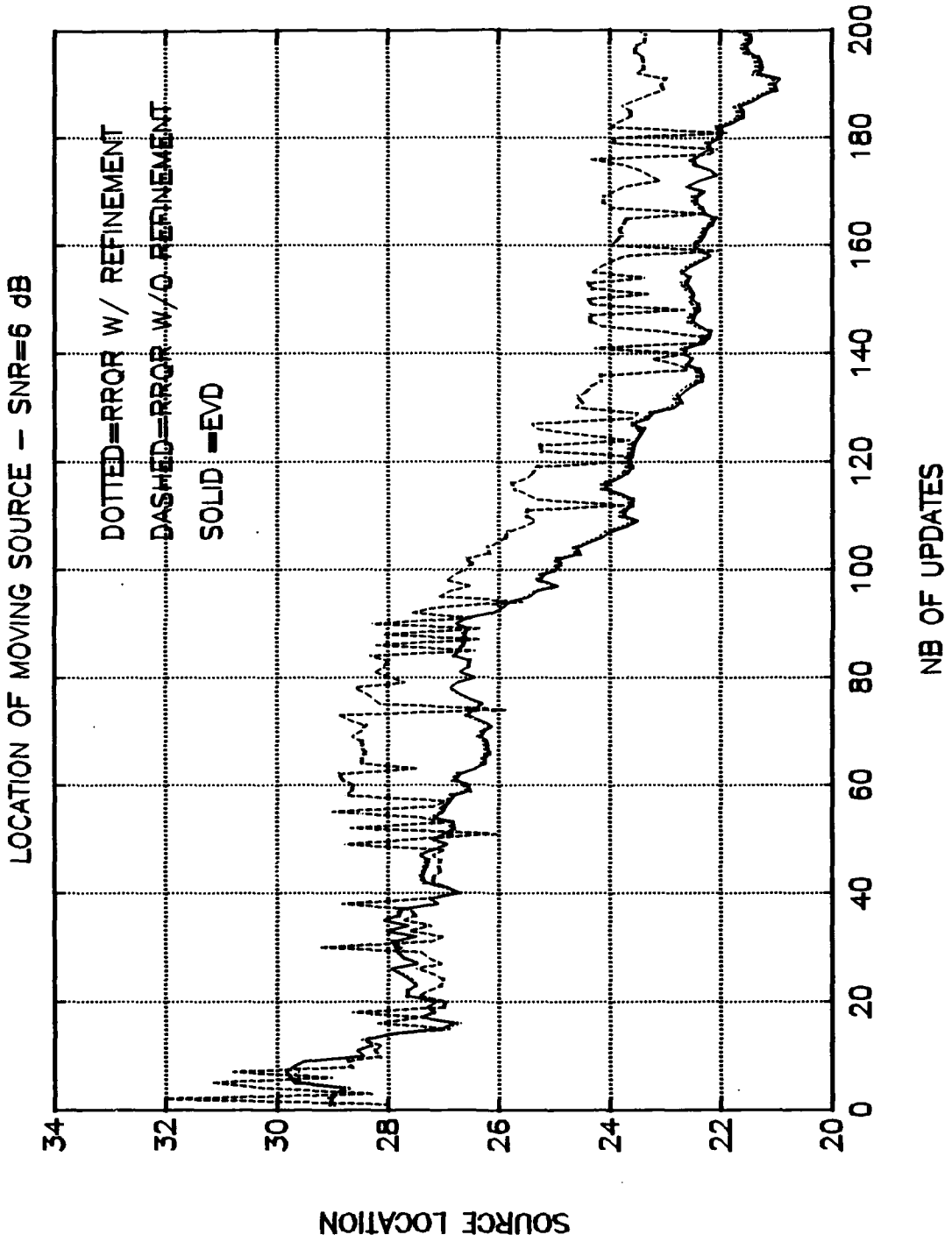


Figure 7: Position of the time-varying source for each update, SNR equal to 6 dB.

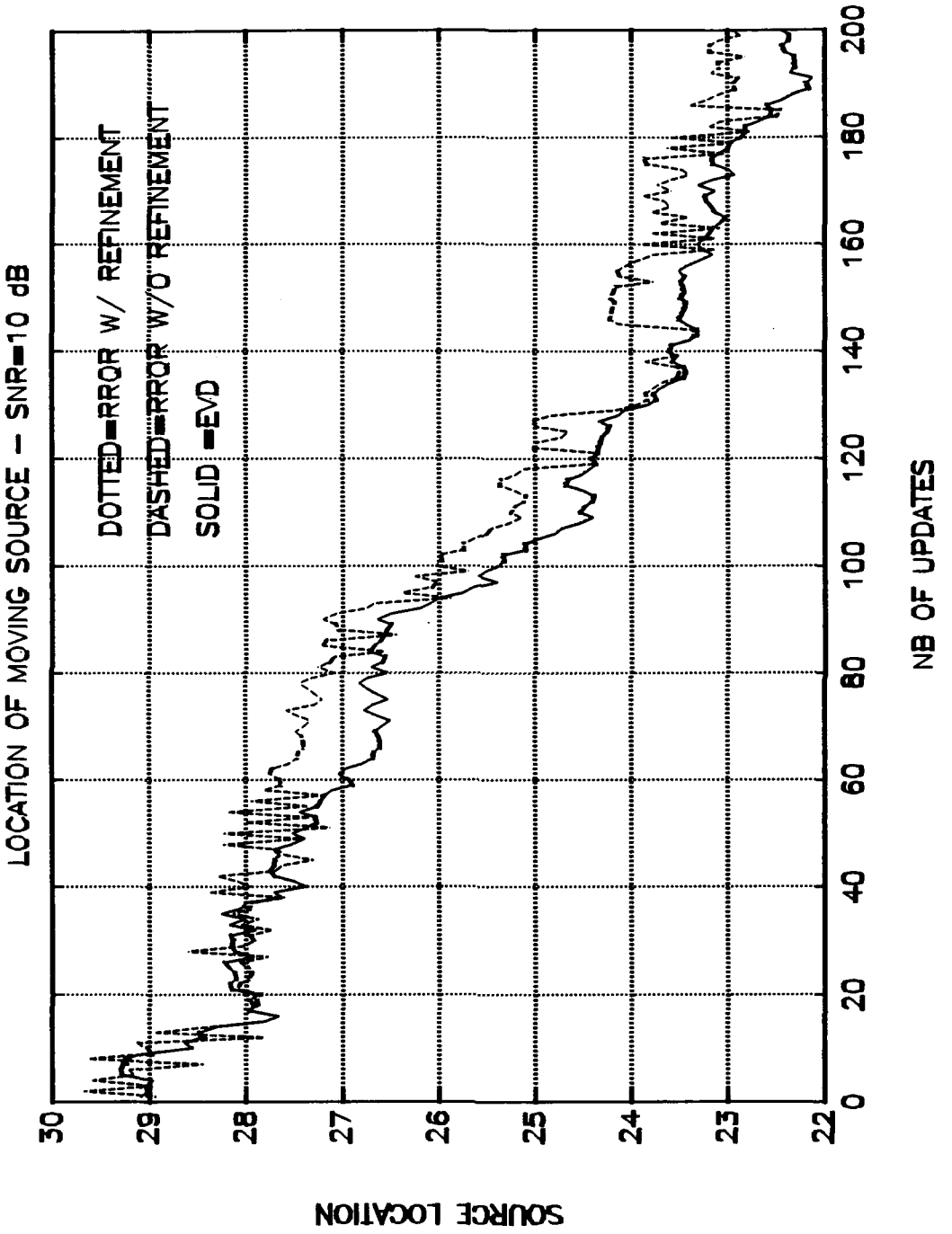


Figure 8: Position of the time-varying source for each update, SNR equal to 10 dB.

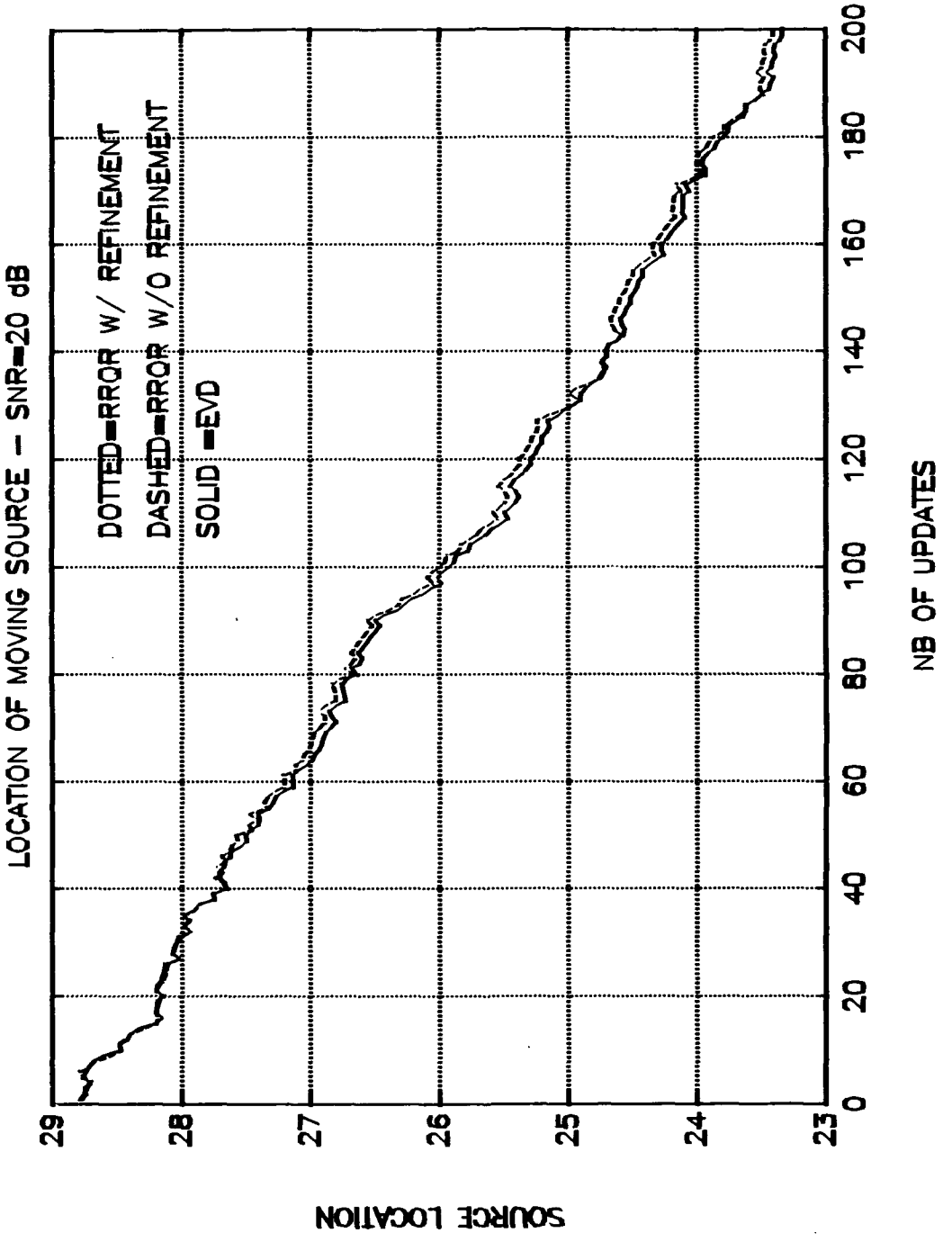


Figure 9: Position of the time-varying source for each update, SNR equal to 20 dB.

Table 25: MEAN AND STD. DEV. FOR THE MAGNITUDE OF THE DIFFERENCE BETWEEN THE RRQR APPROX. FOR THE SIGNAL DOA ANGLE IN DEG. FOR THE MOVING SOURCE WITH/WITHOUT REFINEMENT AND THE EVD.

	SNR=6 dB	SNR=10 dB	SNR=20 dB
Average w/o refinement (degrees)	1.2346	.4530	.0525
Standard Deviation w/o refinement (degrees)	.7569	.2853	.0302
Average w/ refinement (degrees)	.0542	.0079	8.6930 E-5
Standard Deviation w/ refinement (degrees)	.0323	.0061 .	6.2626 E-5

The results are excellent even for the approximation without refinement. The refinement improvement becomes better as the SNR increases.

Results found for the largest principal angle between the projection of the signal subspace found via RRQR and the true signal subspace found via EVD are shown next. Figures 10, 11 and 12 depict the results found for SNR values equal to 6, 10 and 20 dB.

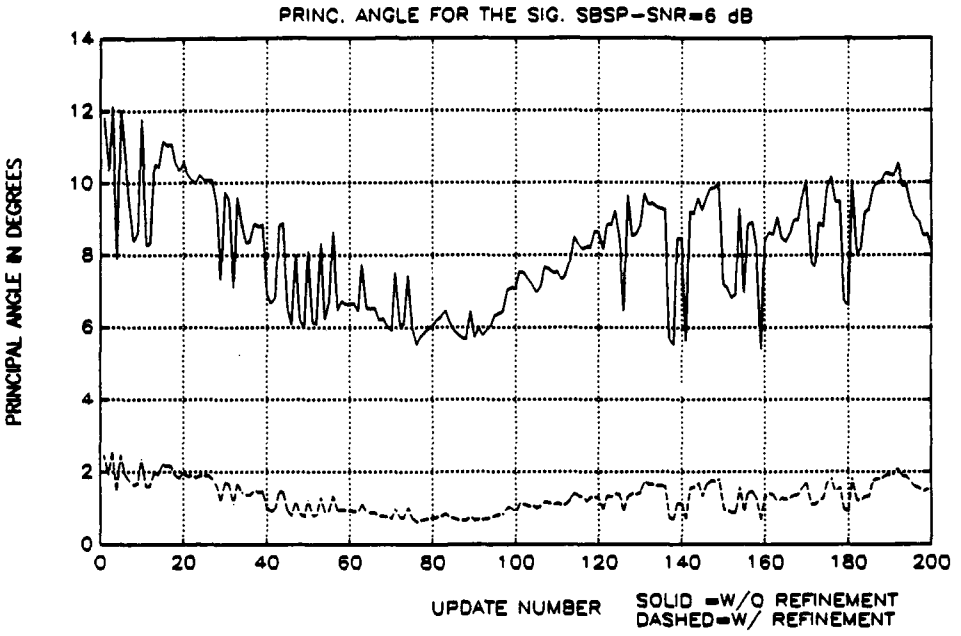


Figure 10: Largest principal angle between the signal subspace found via RRQR and the true signal subspace found via EVD for SNR=6 dB.

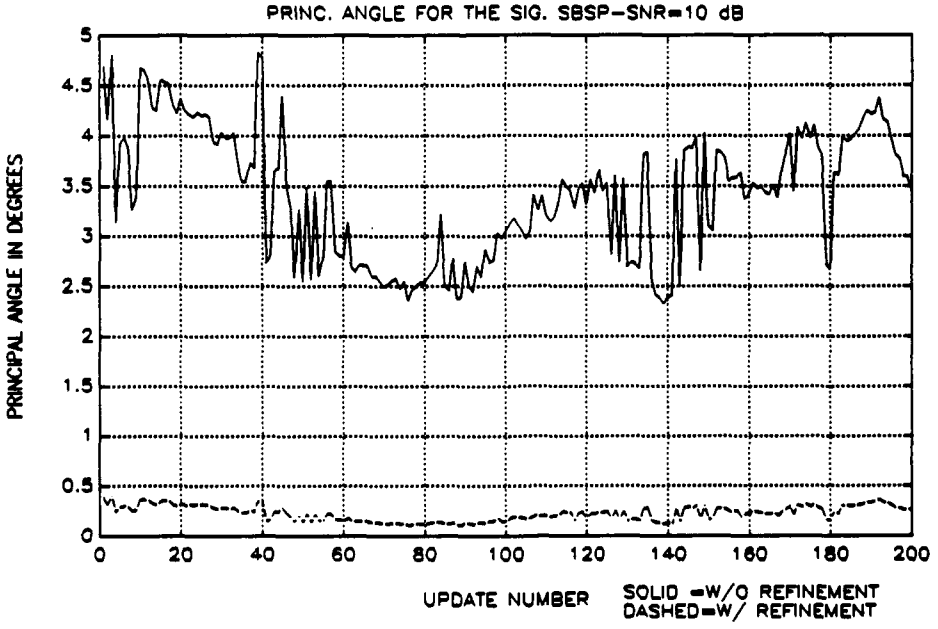


Figure 11: Largest principal angle between the signal subspace found via RRQR and the true signal subspace found via EVD for SNR=10 dB.

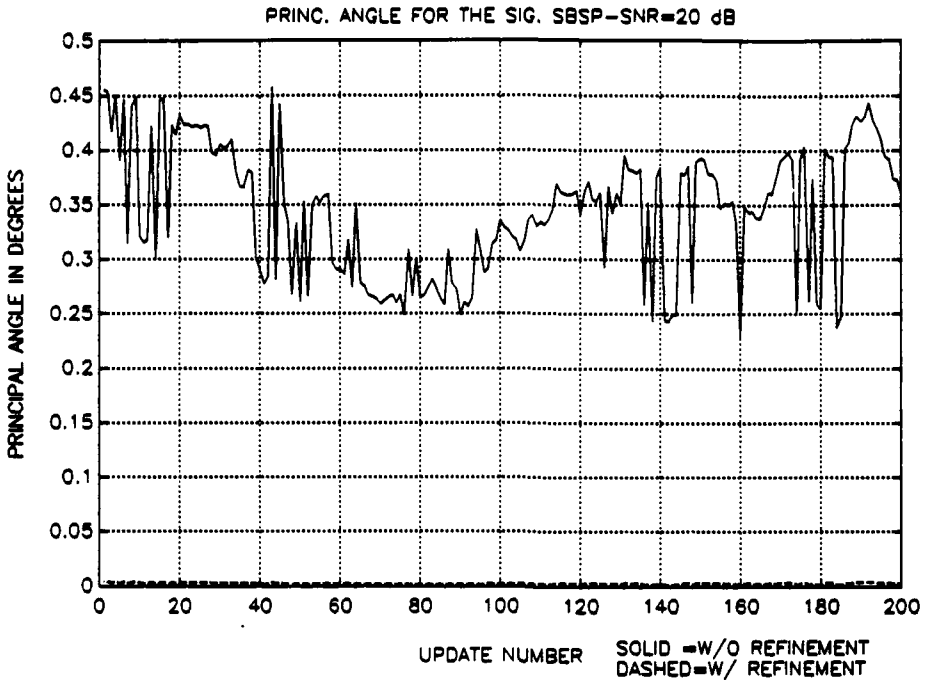


Figure 12: Largest principal angle between the signal subspace found via RRQR and the true signal subspace found via EVD for SNR=20 dB.

As can be seen, the results for the cases with refinement are much better than those without refinement. Table 26 shows the means and standard deviations obtained for the largest principal angle between the RRQR and EVD for both cases, with and without refinement.

Table 26: MEAN AND STANDARD DEVIATION FOR THE LARGEST PRINCIPAL ANGLE IN DEGREES BETWEEN THE RRQR APPROXIMATION AND THE EVD FOR THE SIGNAL SUBSPACE WITH/WITHOUT REFINEMENT.

	SNR=6 dB	SNR=10 dB	SNR=20 dB
Average w/o refinement (degrees)	.81707	3.4087	.3442
Standard Deviation w/o refinement (degrees)	1.5944	.6502	.0595
Average w/ refinement (degrees)	1.2878	.2252	.0023
Standard Deviation w/ refinement (degrees)	.4368	.0737	6.9273-4

Last, Figures 13, 14 and 15 present the behavior of the RRQR approximation for the estimation of the DOA of the second source that remained constant at 40°. Table 27 presents mean and standard deviation values for the magnitude of the difference between the DOA estimated by the RRQR with/without refinement and the EVD. As can be seen, the RRQR results agree closely with EVD results.

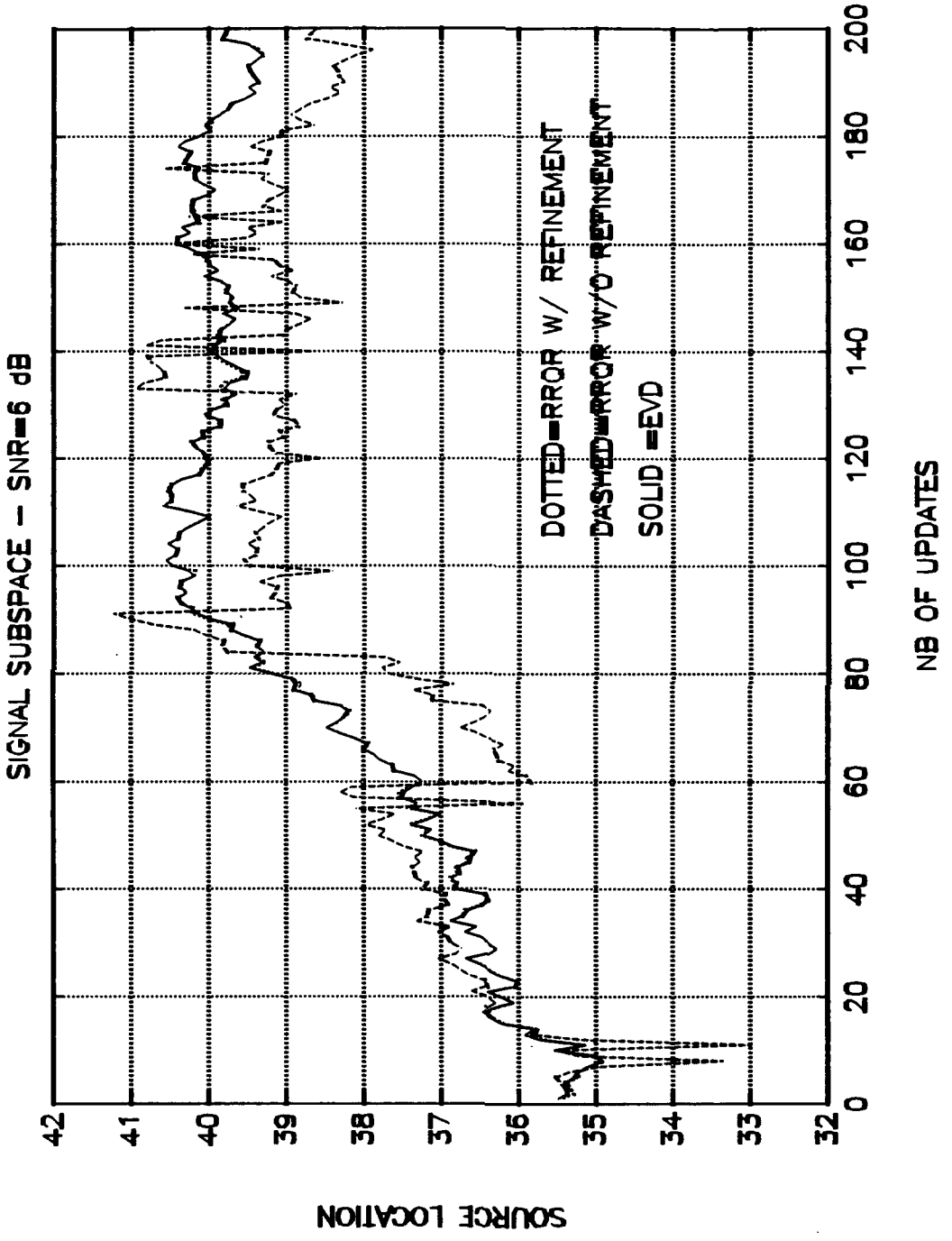


Figure 13: DOA in degrees for the fixed source for each update, SNR equal to 6 dB.

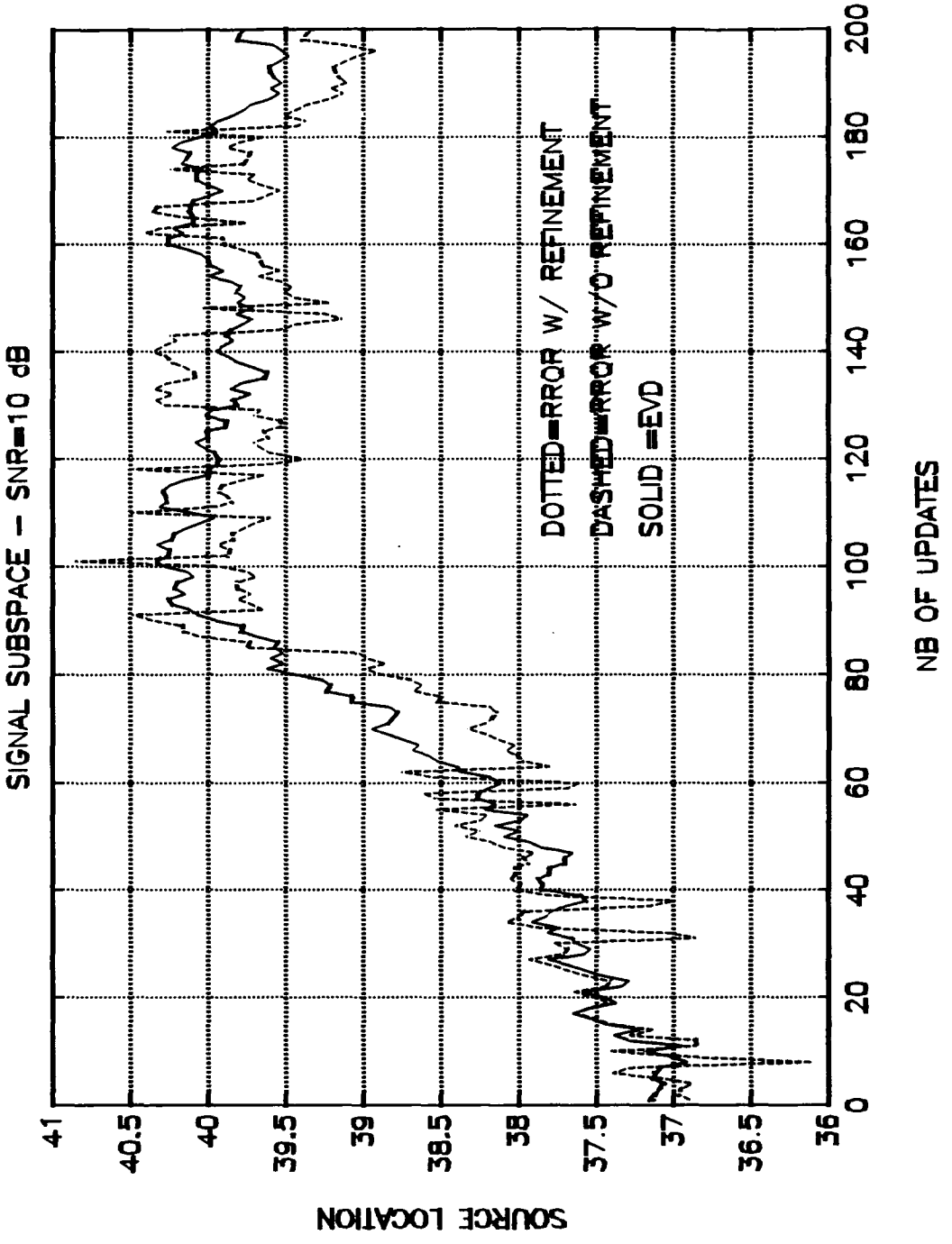


Figure 14: DOA in degrees for the fixed source for each update, SNR equal to 10 dB.

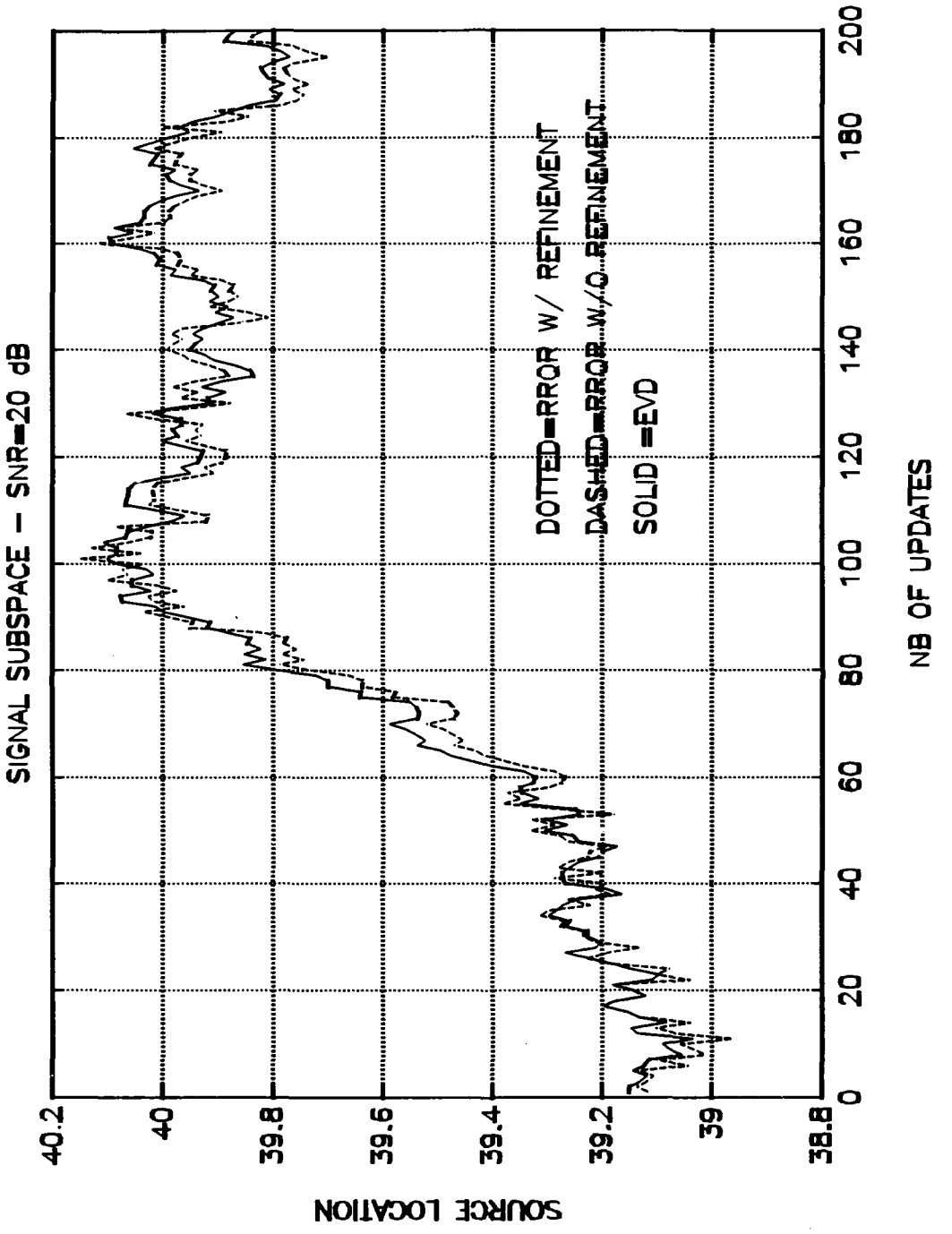


Figure 15: DOA in degrees of the fixed source for each update, SNR equal to 20 dB.

Table 27: MEAN AND STD. DEV. FOR THE MAGNITUDE OF THE DIFFERENCE BETWEEN THE RRQR APPROX. FOR THE SIG. DOA ANGLE IN DEG. WITH/WITHOUT REFINEMENT AND THE EVD FOR THE FIXED SOURCE.

	SNR=6 dB	SNR=10 dB	SNR=20 dB
Average w/o refinement (degrees)	.8970	.3623	.0418
Standard Deviation w/o refinement (degrees)	.4603	.1619	.0184
Average w/ refinement (degrees)	.0226	.0049	7.0319E-5
Standard Deviation w/ refinement (degrees)	.0201	.0047	6.4206E-5

D. COMPUTATIONAL EFFICIENCY OF THE RRQR APPROXIMATION

This section presents a basic estimation of the number of floating-point-operations (flops) needed to compute the RRQR approximation. The n-dimensional noise-free autocorrelation matrix is square and is not considered complex valued for flop computation. This fact will be taken into account later. The number of flops necessary to perform one SVD decomposition is $6n^3$ [Ref. 4]. Every time a new sample arrives, we need an $O(n^3)$ operation to recompute the SVD [Ref. 13].

The RRQR algorithm is composed of three distinct parts. The computation of the initial QR factorization without pivoting, computed only once in the beginning of the algorithm; the computation of the least dominant right singular vector v of R_{11} by inverse iteration, at each iteration; and the new QR factorization of $R_{11}P$, also at each iteration.

The first part takes $2n^3/3$ flops using the Householder algorithm if we do not need to accumulate Q. If Q is needed, as in our case, it takes $4n^3/3$ flops [Ref. 7]. The Modified Gram-Schmidt algorithm is preferred, since it takes only n^3 flops when the accumulation of Q is performed [Ref. 7].

Ignoring lower order terms, the second part of the RRQR factorization takes In^2r flops to be iterated [Ref. 4], where r is the noise-free autocorrelation matrix rank-deficiency and I is the number of iterations used in the inverse iteration method. Our case uses I=3, therefore, the second part takes $3n^2r$ flops to be performed.

In the third part of the RRQR algorithm, $2n^2r$ flops are needed when Givens rotations are used [Ref. 4]. However, note that not all elements below the main diagonal of the matrix $R_{11}P$ needs to be annihilated because they are already zero. Therefore, a conditional "IF" statement may be used to verify if the element is already zero to save additional flops. Thus, the RRQR algorithm totals n^3+5n^2r flops at most.

Following the first RRQR decomposition, we have two options. The first option updates the autocorrelation matrix, as in Equation (48) and performs a new QR decomposition. This update is followed by the pivoting scheme, as in steps 0 through 9 of Chan's algorithm. The second option updates the already obtained QR decomposition directly, using two successive rank-one modifications. A new Q_1 and R_1 is found

and the Chan's pivoting scheme is applied. An analysis of the number of flops necessary to perform the two options is made.

Recall that in the adaptive case, a double update must be performed for each sample. One update adds the new sample and the other subtracts the old one. If the rank-one modification option is used, the updates take $13n^2$ flops each for each snapshot, totaling $26n^2$ flops [Ref. 7]. This does not include the number of flops necessary to proceed the pivoting scheme. If the complete RRQR factorization option is used, the QR decomposition using the Modified Gram-Schmidt algorithm takes n^3 flops. Note that in such a case, the autocorrelation matrix must be updated as $R_{new}=R_{old}+(x\cdot x^H)_{new}-(x\cdot x^H)_{old}$, which takes $4n^2$ flops (n^2 for each multiplication and n^2 for each addition). The total is n^3+4n^2 for each snapshot.

Comparing n^3+4n^2 with $26n^2$, we see that to update the noise-free autocorrelation matrix (finding R_{new} as in Equation (48)) and to take its QR decomposition is more economical than to perform two rank-one modifications for $0 < n < 22$. Therefore, a complete RRQR algorithm including a Modified Gram-Schmidt QR decomposition and the pivoting scheme is preferable when compared to the two rank-one modification updates, for a number of sensors smaller than 22. This method does not take into consideration the difference of pivoting schemes needed after the update. Table 28 presents a comparison for the number of flops needed for both processes.

Table 28: COMPARISON OF THE NUMBER OF FLOPS NEEDED TO THE SNAPSHOT UPDATE OF THE ADAPTIVE ALGORITHM VIA A COMPLETE RRQR FACTORIZATION AND TWO RANK-ONE MODIFICATIONS.

Snapshot update via complete RRQR algorithm	Number of flops needed	Snapshot update via two rank-one modif.	Number of flops needed
Computation of R_{new}	$4n^2$	First rank-one modif.	$13n^2$
QR Factoriz. of R_{new}	n^3	Second rank-one modif.	$13n^2$
Smallest right SV	$3n^2r$	Smallest right SV	$3n^2r$
QR Factoriz. of $R_{11}P$	$2n^2r$	QR Factoriz. of $R_{11}P$	$2n^2r$
Total	$n^3+(5r+4)n^2$	Total	$(5r+26)n^2$

As seen earlier, using two successive rank-one modifications leads to pivoting for at most 32% of the 1600 iterations used to perform the 200 updates. This is compared to the maximum of 88% pivoting when updating the noise-free autocorrelation matrix and applying a complete RRQR algorithm. These numbers are used to compare the two approaches.

For the third part of the RRQR algorithm, we need $2n^2r$ flops. Using the first approach, two rank-one modifications over Q and R requires a total of $26n^2+(3+0.32x2)n^2r$ flops for each update. In the second approach, the updating of the noise-free autocorrelation matrix and the application of a complete RRQR algorithm requires $n^3+4n^2+(3+0.88x2)n^2r$ flops for each update. Figure 16 depicts the results showing the regions when the complete RRQR or the update approaches might be preferred.

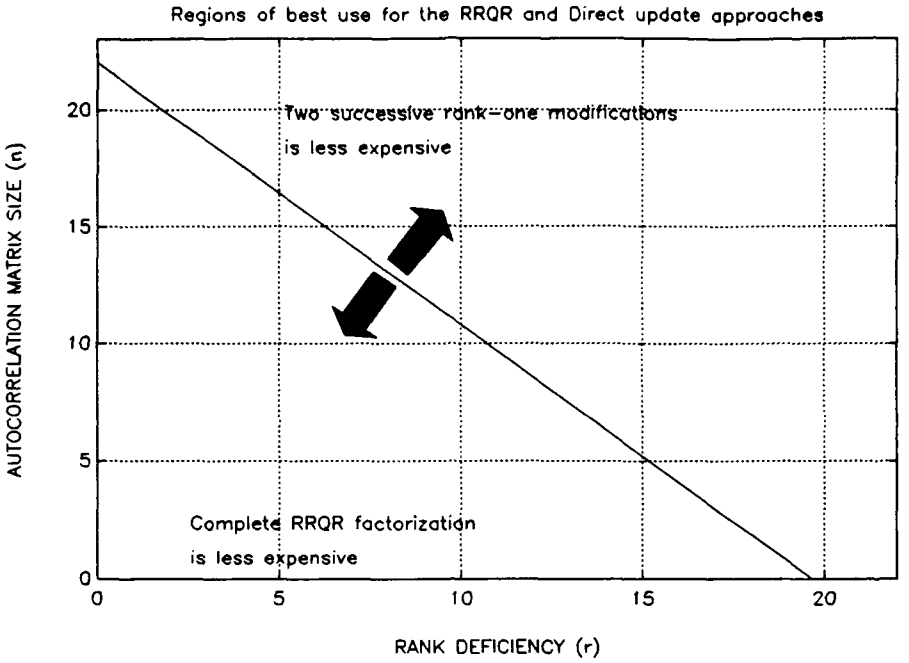


Figure 16: Graph showing the regions where the RRQR or the two rank-one modifications approach is preferred.

The MINNORM algorithm takes roughly a number of flops equals to 2nr plus the flops necessary to compute the polynomial roots. The root finding routine is iterative. Thus, it is impossible to obtain a specific expression to evaluate the number of flops necessary to run it. However, for the scenario simulated in the previous section, we were able to compute the mean and standard deviation of the number of flops spent by the MATLAB™ software to find the ten roots for each one of the 200 updates.

To evaluate the sensitivity of the root finding routine to the SNR, we tested for SNR=-100, 6 and 100 dB. The results are shown in Table 29.

Table 29: MEAN AND STD. DEV. FOR THE NUMBER OF FLOPS NECESSARY TO FIND TEN ROOTS OF THE POLYNOMIAL FORMED BY THE MINIMUM NORM ALGORITHM. SNR=-100, 6, 100 DB.

SNR(dB)	Mean of # flops	Std. Dev. # flops
-100	63159	3042
6	61007	2168
100	64420	148

According to Table 29, the number of flops necessary to the root finding routine presents a smaller standard deviation for SNR equals to 100 dB. This happens because at such a high SNR, the zeros are at more or less fixed locations. Thus, one can expect about the same number of iterations needed to identify them. The study of the polynomial root algorithm sensitiveness to the number of sources at different locations deserves further research and is out of the scope of this work.

The REFINEMENT algorithm spends a total of $3n^3+(1-2r)n^2-rn$ flops, including the orthonormalization necessary to be used in conjunction with the MINNORM. When the problem is located under the line depicted in Figure 16, the updating via a complete RRQR algorithm is preferred in the most probable case. Totalizing, the RRQR, the MINNORM and the REFINEMENT algorithms take a total number of flops of $4n^3+(2.76r+5)n^2+rn$

for each update. This value was developed for a real valued noise-free autocorrelation matrix. To cope with future growth of the program, we might multiply the number of flops by a 4 to 1 factor when implementing it operationally and by a roughly 4 to 1 factor to estimate the case of a complex matrix.

E. INTEGRATING THE RRQR ALGORITHM INTO A REAL-TIME CASE

Suppose we have a signal being received by a linear phased array on the earth surface from a moving object located at 10 NM from the sensors at an angle of 80° from the sensors vertical (see Figure 17).

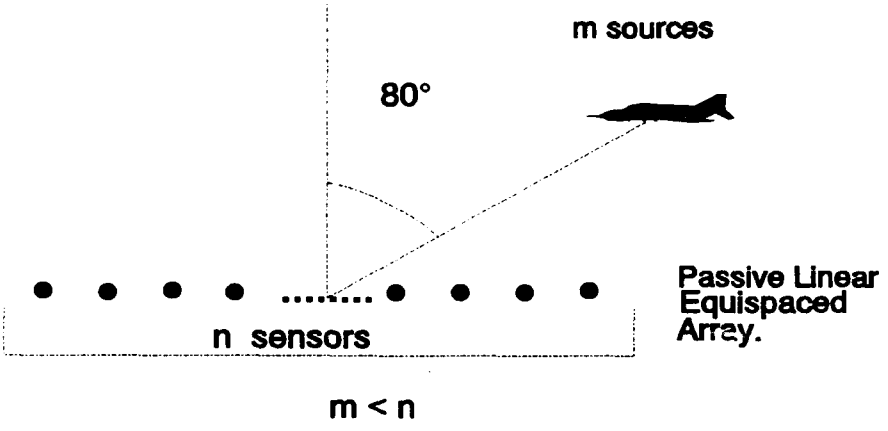


Figure 17: Situation of a signal being received by linear phased array sensors.

Also suppose that this signal is moving at an horizontal velocity v. Based on the simulations run in this thesis, the moving signal changed 5° in its DOA with respect to the array vertical. This was done in 200 snapshots. Each snapshot is taken at an interval of t_s seconds. Therefore, the angular speed of this moving signal is defined as

$$w=\frac{5.\frac{\pi}{180}}{200.ts}\ \frac{rd}{s}. \tag{53}$$

The signal velocity is

$$v=\frac{w\cdot R}{\cos(\theta)}\ , \tag{54}$$

where R is the distance between the signal and the array. Using (53) and (54) leads to:

$$t_s=\frac{5\cdot\pi\cdot R}{36000\cdot v\cdot\cos(\theta)}. \tag{55}$$

Assuming a signal is moving at sound speed as in Figure 18, θ=80°, R=10 NM, a t_s of 136.87 ms would be needed.

For the scenario simulated in the previous section where n=10 and r=8, the number of flops needed for each update would be 16x6788=108.6 Kflops per update. This value is equivalent to 108,600/136.87 ms ≈ 790 Kflops per second of microprocessor computing power. Assuming a reasonable computing power of one flop per clock pulse at 32 bits, a microprocessor would have to operate at approximately 790 KHz. Current commercial

microprocessors sold with a clock of 50 MHz would be able to implement this algorithm in real-time. Note that there is enough room to accommodate the root finding procedure neglected in the MINNORM algorithm flops computation.

VII. CONCLUSIONS

We have presented a fast algorithm to isolate signal and noise subspaces without performing the eigenvector decomposition of the autocorrelation matrix. This method is called the Rank-Revealing QR factorization.

We have investigated the possibility of using the least dominant eigenvector instead of using the least dominant singular vector in step two of the RRQR algorithm. Simulations have shown that the minimum singular vector generated by the inverse iteration method gives better results than those obtained with the approximate smallest eigenvector generated by the same method. The Incremental Condition Estimator algorithm for finding an approximation for the smallest singular vector has also been tested. Simulations have demonstrated that the inverse iteration again yields better results.

The possibility of using a faster RRQR algorithm than the original algorithm with fewer iterations has been investigated. Simulations have shown that reducing the number of iterations worsens the signal/noise subspace estimations. Therefore, the original algorithm is preferred.

Two spectral estimators have been tested to be used with the RRQR algorithm, the MUSIC and the Minimum Norm. A limited number of simulations have indicated that the Minimum Norm

resolved two signals for a lower SNR than the MUSIC method, in agreement with Rao [Ref. 12].

A relatively inexpensive computational refinement algorithm has been presented for the estimation of the RRQR signal subspace. Simulations have shown that improvements at least as high as a factor of 20 are possible to be obtained for the signal angle of arrival, as compared with the original RRQR-based DOA results. Note that this refinement improvement becomes better as the SNR increases.

An adaptive RRQR-based algorithm has been introduced to track the DOA of moving signals. Two options have been evaluated to compute correlation updates. The more adequate option may be determined depending upon the particular problem set up, e.g., the noise-free autocorrelation matrix rank deficiency (r) and the number of sensors (n).

An evaluation of the number of flops required by the adaptive algorithm to find the DOA for two sources present has been determined. The results have shown the feasibility of the algorithm to solve a real-time problem.

Appendix A

```
%
% This function implements Chan's RRQR algorithm.
% Milton P. Ferreira, Sep/1992.
%
% Input parameters:
%
% r= matrix to which we want to apply the RRQR factorization
% (noise-free autocorrelation matrix.
% r1= rank deficiency of the matrix "r".
%
% Output parameters:
%
% Q= orthonormal matrix resulting from the RRQR factorization
% of "r".
% R= upper triangular matrix resulting from the RRQR
% factorization of "r".
% e= permutation matrix (PI) resulting from the RRQR
% factorization of "r".
% nsbsp= noise subspace.
% ssbsp= signal subspace.
%
function [Q,R,e,nsbsp,ssbsp]=rrqr(r,r1)
n=size(r);n=n(1,1);
dd=n-r1;
w1=zeros(n);
[Q R]=qr(r);
e=eye(n);
coun=0;
for i=n:-1:n-r1+1;
  R11=R(1:i,1:i); % FIND THE THE LEADING ixi BLOCK (STEP 1)
  [u,sigmin,v]=ssvd(R11,3); % FIND THE LEAST DOMINANT
        %SINGULAR VECTORS AND SINGULAR VALUE (STEP 2)
  pt=eye(i);
 vinf=norm(v,inf);          % FIND THE MAXIMUM ABSOL. VALUE
    %           ELEMENT OF THE LEAST DOMINANT SINGULAR VECTOR
     vaux1=abs(v);
     ind=find(vaux1==vinf);
     if ind~=i,         % FIND THE POSITION OF THE MAX ELEMENT
          vaux=pt(ind,:);   % MAKE THE PERMUTATION TO PUT THE
   %           MAXIMUM ELEMENT AT THE i th POSITION (STEP 3)
          pt(ind,:)=pt(i,:);
          pt(i,:)=vaux;
      w1(:,i)=[v;zeros(n-i,1)]; % ASSIGN v TO THE ith COLUMN
                                              %OF w  (STEP 4)
     p=pt';
     ptil12=zeros(i,(n-i));
     ptil21=zeros((n-i),i);
```

```
      ptil22=eye((n-i));
      ptil=[p ptil12 ; ptil21 ptil22];
      w=ptil'*w1;                                % STEP 5
      [Q1 R11til]=qrgiv(R11*p);                  % STEP 6
      e=e*ptil;                                  % STEP 7
      R12=R(1:i,(i+1):n);
      if (n-i)==0,
          R22=[];
          else
          R22=R((i+1):n,(i+1):n);
      end;
      R=[R11til Q1'*R12 ; zeros((n-i),i) R22];
      Q12=zeros(i,(n-i));
      Q21=zeros((n-i),i);
      Q22=eye((n-i));
      Q=Q*[Q1 Q12 ; Q21 Q22];
end;
end;
nsbsp=Q(:,n-r1+1:n);
ssbsp=Q(:,1:n-r1);
```

Appendix B

```
function jik = givens1(x,y,n,i,k)
%GIVENS   Givens rotation matrix.
%     G = GIVENS(x,y) returns the complex Givens rotation
matrix
%
%          |  c        s  |                    |  x  |     |  r  |
%     G =  |              |   such that  G *   |     |  =  |     |
%          | -conj(s)  c  |                    |  y  |     |  0  |
%
%     where c is real, s is complex, and c^2 + |s|^2 = 1.

%     Copyright (c) 1987-88 by The MathWorks, Inc.
%     Modified by Milton P. Ferreira Sep/1992
%
% Input parameters:
%
% x= pivoting element [a(i,i)].
% y= element we want to zero out [a(k,i)].
% n= matrix dimension.
% i= column of the element we want to zero out.
% k= row of the element we want to zero out.
%
% Output parameters:
%
% jik= matrix "J". When multiplied by the matrix "a" will zero
% out the element a(i,k).
absx = abs(x);
if absx == 0.0
     c = 0.0; s = 1.0;
else
     nrm = norm([x y]);
     c = absx/nrm;
     s = x/absx*(conj(y)/nrm);
end
jik=eye(n);
jik(i,i)=c;
jik(k,i)=-conj(s);
jik(i,k)=s;
jik(k,k)=c;
```

Appendix C

```
% Computes Givens Rotations to zero out the elements of a
% matrix below its main diagonal. It is used at step 6 of
% the RRQR algorithm instead of a new QR factorization of
% R11*P.
% Milton P. Ferreira Sep/1992.
%
% Input parameters:
%
% w= matrix whose elements below the main diagonal we want to
% zero out.
%
% Output parameters:
%
% q1= new matrix Q, after applying Givens Rotations.
% r1= new matrix R, after applying Givens Rotations.
%
function [q1,r1]=qrgiv(w)
i=size(w);i=i(1,1);
qt=eye(i);
for q=2:i;
   for p=1:min([q-1,i]);
     if w(q,p)~=0,
           jpq=givens1(w(p,p),w(q,p),i,p,q);
           qt=jpq*qt;
           w=jpq*w;
      end;
   end;
end;
q1=-qt';
r1=-w;
```

Appendix D

```
% Algorithm for the adaptive tracking of a moving source
% Main program.
% Milton P. Ferreira Sep/1992.
%
% Input parameters:
%
% db= SNR.
%
% Output parameters:
%
% R0= autocorrelation matrix.
% R1= noise-free autocorrelation matrix.
% Y= each column of Y is one output vector "x" from the "n"
% sensors.
% nest= # of snapshots used to compute the autocorrelation
% matrix.
% nupd= # of updates used to simulate the movement.
% ipp= autocorrelation matrix dimension.
% nsin= number of sources.
%
[R0,R1,Y,nest,nupd,ipp,nsin]=cor4m1(db);
[Q,R,e,nsbsp,ssbsp]=rrqr(R1,ipp-nsin);
for i=nest+1:nest+nupd;
     old=Y(:,i-nest);
     [Q,R]=givqr(Q,R,e,old,-old);
     new=Y(:,i);
     [Q,R]=givqr(Q,R,e,new,new);
     [Q,R,e,nsbsp,ssbsp]=rrqe(Q,R,e,nsin,ipp); % is the RRQR
          % subroutine without the initial QR decomposition
     [mags,angs]=minn(nsbsp,ssbsp,ipp);
     [ma,an]=ident(mags,angs);
     angupd(i-nest)=an(2,1);
end;
plot(i,angupd);
grid;
title(['SIGNAL SUBSPACE - SNR=',int2str(db),' dB']);
ylabel('SOURCE LOCATION');
xlabel('NB OF UPDATES');
text(.7,.5,'SOLID =UPDATE','sc');
```

Appendix E

```
% Computes the rank-one modification of a square matrix.
%
% Input parameters:
%
% Q= orthonormal matrix resulting from the RRQR
% factorization.
% R= upper triangular matrix resulting from the RRQR
% factorization.
% e= permutation matrix (PI), resulting fron the RRQR
% factorization.
% u and v= u and v vectors that modify the matrix to be
% updated.
%
% Output parameters:
%
% Q1= matrix Q after modifying.
% R1= matrix R after modifying.
%
% Milton P. Ferreira Sep. 1992.
%
function [Q1,R1]=givqr(Q,R,e,u,v);
w=Q'*u;
Q1=Q;
H=R;
i=size(w);i=i(1,1);
for q=i-1:-1:1;
     jpq=givens1(w(i,1),w(q,1),i,i,q);
     w=jpq*w;
     H=jpq*H;
     Q1=Q1*jpq';
end;
R1=H+w*v'*e;
for q=1:i-1;
     jpq=givens1(R1(q,q),R1(i,q),i,q,i);
     R1=jpq*R1;
     Q1=Q1*jpq';
end;
```

Appendix F

```
function [mags,angs]=minn(nsbsp,ssbsp,ipp)
% compute the noise and signal zero locations using the TK
% min-norm
% alg.  (version 1.0   10/14/91), Monique P. Fargues.
%
% Input parameters:
%
% ipp= correlation matrix dimension
% nsbsp= noise subspace.
% ssbsp= signal subspace.
%
% Output parameters:
%
% mags= magnitude of the polynomial roots.
% angs= phase of the polynomial roots.
%
%
dn=zeros(1:ipp);ds=zeros(1:ipp);
clear g
g=nsbsp(1,:);              %noise zeros
En=nsbsp(2:ipp,:);
dn(2:ipp)=En*g'/(g*g');
dn(1)=1;
clear g
g=ssbsp(1,:);              %signal zeros
Es=ssbsp(2:ipp,:);
Kg=-1/(1-g*g');
ds(2:ipp)=Kg*(Es*g');
ds(1)=1;
flops(0);
dnr=roots(dn);
dsr=roots(ds);
mags=abs(dsr);
angs=angle(dsr)*180/pi;
```

Appendix G

```
% Computes the "refined" signal subspace
% Version 1.0 08/21/92, Monique P. Fargues.
%
% Input parameters
%
% R= upper triangular matrix resulting from the RRQR
% factorization.
% e= permutation matrix (PI) resulting from the RRQR
% factorization.
% nsin= # of sources.
% ipp= correlation matrix dimension.
%
% Output parameter:
%
% ref= "refined" and reorthonormalized signal subspace.
%
function ref=refine(R,e,nsin,ipp)
pi1=e(:,1:nsin);
pi2=e(:,nsin+1:ipp);
R11=R(1:nsin,1:nsin);
R12=R(1:nsin,nsin+1:ipp);
ref=orth(pi1*R11'+pi2*R12');
```

Appendix H

```
%  identify the signal magnitude and location of the roots
%  version 1.0  10/13/91, Monique P. Fargues.
% Input parameters:
%
% mag= magnitude of the polynomial roots
% ang= phase of the polynomial roots
%
% Output parameters:
%
% mag_r= vector containing the magnitude of the roots
% corresponding to the sources.
% ang_r= idem to the phases.
%
function [mag_r,ang_r]=ident(mag,ang)
[m,n]=size(ang);
k=1;
ang_r0(1:2,1)=[0;0];mag_r0(1:2,1)=[0;0];
for i=1:m
if mag(i)<1.3 & ang(i)<50 & ang(i)>10  %look at both
  ang_r0(k,1)=ang(i) ;
  mag_r0(k,1)=mag(i);
  k=k+1;
end
if k==3, break, end
end
% sort to insure proper separation of sines
[ang_r(:,1),1]=sort(ang_r0(:,1));
mag_r(:,1)=(mag_r0(1,1));
```

Appendix I

```
function [sigmin,x]=icest(R)
%  1/7/92       ******-icest.m-******
%  version 1.0, Monique P. Fargues.
%  incremental condition estimator (modified from Bischoff
%paper)
%  computes an estimate for minimum singular value and
%vector
%  associated
%  with an upper triangular matrix
%  initial matrix      | gamma   v' |
%                      |   0     R  |
% Input parameters:
%
% R= matrix we want to estimate the smallest S. value and
% its S. vectors.
%
% Output parameters:
%
%   sigmin: minimum singular value
%   x:      ------- singular vector
%-------------------------------------------------------------
clear xx x v

[m,n]=size(R);
%if(m~=n), error ('R is not square'), end
%if(any(diag(R)==0))  % matrix is singular
%  smin=0; vmin=zeros(n,1);vmin(min(find(diag(R)==0)))=1;
%  return
%end
%
x(n,1)=1/R(n,n);

for i=n-1:-1:1
  xx(1:n-i,1)=x(i+1:n,1);
  v=R(i,i+1:n)'; gamma=R(i,i);
  alpha=v'*xx;
if alpha~=0
  beta=(abs(gamma)*norm(xx,2))^2 + abs(alpha)^2 -1;
  eta=beta/(2*abs(alpha));
  sqr=sqrt(eta^2+1);
  nu=eta + sqr          ;                  % sqrt(eta^2+1);
  temp=abs(alpha)*nu;
  root=temp+1; sqr=sqrt(nu^2+1);
  s=temp/(alpha*sqr)          ;   % sqrt(nu^2+1));
  c=-1/sqr                              ;  % sqrt(nu^2+1);

else
```

```
  root=(abs(gamma)*norm(xx,2))^2;
  if(root>1), c=1, s=0;
  else, c=0, s=1; end
end

x(i:n,1)=[(c-s*alpha)/gamma;s*x(i+1:n,1)];

end   % of initial loop
xnorm=norm(x(i:n,1),2);
x(1:n,1)=x(1:n,1)/xnorm;
sigmin=1/xnorm;
```

Appendix J

```
function [vsv,sigmin,usv]=ssvd(a,k)

% THIS FUNCTION IS THE IMPLEMENTATION OF THE INVERSE
%ITERATION
% TECHNIQUE TO FIND THE SINGULAR VECTORS AS SHOWN IN THE
%PAPER
% "DEFLATED DECOMPOSITION OF SOLUTIONS OF NEARLY SINGULAR
% SYSTEMS" - TONY F. CHAN - PG 746 - SIAM J. NUMER. ANAL.
% VOL 21 #4 AUG. 1984
% [vsv,sigmin,usv]=ssvd(a,k).
%
% Input parameters:
%
% a= matrix we are looking for the smallest singular
% vectors.
% k= number of iterations desired.
%
% Output parameters:
%
% usv, vsv= right and left smallest S. vectors.
% sigmin= minimum singular value.
%
% Milton P. Ferreira
%
n=size(a);
n=n(1,1);
v=ones(n,1);
for i=1:k;
     util=a\v;
     u=util/norm(util);
     vtil=a'\u;
     v=vtil/norm(vtil);
end;
vsv=v;
util=a\vsv;
usv=util/norm(util);
sigmin=1/norm(util);
```

Appendix K

```
function [R0,R1,Y,nest,nupd,ipp,nsin]=cor4m1(db);
%
% Monique P. Fargues.
%
%  COMPUTE THE CORRELATION FUNCTION ONLY
%  R0: Toeplitz correlation function
%
% Input parameters:
%
% db= SNR.
%
% Output parameters:
%
% R0= autocorrelation matrix.
% R1= noise-free autocorrelation matrix.
% Y= each column of Y is one output vector "x" from the "n"
% sensors.
% nest= # of snapshots used to compute the autocorrelation
% matrix.
% nupd= # of updates used to simulate the movement.
% ipp= autocorrelation matrix dimension.
% nsin= number of sources.
%
clg;format compact
seed1=1042;rand('seed',seed1)

icor=0;      %input('true/est correl 1/0: ');
itop=0;      %input('toeplitz/non toeplitz 1/0: ');
nupd=200;    %input('update nb nupd: ');
%nupd0=input('update nb nupd0 (drop in angle): ');
del_freq=5; % input('del_freq (in percen*freq(1): ');
%if icor==1
%   fprintf('true cor. seq\n')
%else
%   fprintf('est. cor. seq\n')
%end
nest=100;
%gdb=input(' input gdb: [x x] ');
gdb=[db,db];
ang=[30 40];          % source angles
freqt=[7.5 9];        % temporal frequencies
ipp=10;               % number of sensors
nsin=2;               % number of sources
ssig=1.;              % ref. noise variance
jc=sqrt(-1);
sigma=1;
```

```
for i=1:nsin                                    % linear amplitude
  A(i)=sqrt(2.)*(10^(gdb(i)/20.));
end
if icor==1
% compute the true correlation sequence
for k=1:ipp
res=0;
if k==1
 res=ssig^2;
end
  for i=1:nsin
  res=res+(A(i)^2)*exp(jc*(k-1)*ang(i)*pi/180.)/2.;
  end
R(k)=res;
end
R0=toeplitz(R);
else

% compute the estimated correlation seq
% based on nest data points
rand('normal')
%sigm=sqrt(10^((sigma)/10));
for i=1:nsin
  A(i)=sqrt(2)*sigma*(10^(gdb(i)/20));
end
for i=1:ipp

Y(i,1:nest+nupd)=sigma*(rand(1,nest+nupd)+jc*rand(1,nest+nup
d));
end
freq=ang*pi/180;
rand('uniform')              % create uniform variable dist.
                           % in (-pi,pi)
X1=pi*(rand(1,nsin*ipp*(nest+nupd))-.5);

freq0=freq(1);
for j2=1:nest+nupd                                % j2: time
snapshot
%freq(1)=freq0 - (pi/180)*(j2-1)*del_freq/nupd; %del_freq
                            % change in nupd samples
%if j2>nest+nupd0, freq(1)=freq0*del_freq; end %step freq.
% change

  for i=1:nsin                              % i: number of sines
  for j1=1:ipp                              % j1: sensor position
    temp=(j1*freq(i)+j2*freqt(i)+X1(1,j2+(i-1)*ipp));
    Y(j1,j2)=Y(j1,j2)+A(i)*exp(jc*temp);
  end
  end
end
R0=Y(:,1:nest)*Y(:,1:nest)';
```

```
end

%get the noise free version of RT & normalize the matrix
% compute the EVD of the noisy matrix for later computations
% for original correlation function
%[Ve,De]=eig(R0);
% sort the eigenvalues
%[Des,l]=sort(diag(De));l2=flipud(l);
%Ves=Ve(:,l2(:));              % sorting in descending order
%Vnoi=Ves(:,nsin+1:ipp);       % isolate the noise vectors
%Vsig=Ves(:,1:nsin);           % ----------- signal -------
R1=R0-nest*ssig^2*eye(R0);     % noise free correlation
matrix
%R1=R1./R1(1,1);              % potential matrix
                                % normalization

if icor==1,
        R1=R0-ssig^2*eye(R0);     % noise free correlation
matrix
end;
if icor==0,
        R1=R0-nest*ssig^2*eye(R0);
end;
if itop==1,
     R1=toep(R1);
end
```

Appendix L

```
% Transforms a matrix to a Toeplitz matrix. Used with
% "cor4m1.m".
%
% Input parameter:
%
% Matrix to be transformed.
%
% Output parameter:
%
% sum= Toeplitz matrix after transformation.
%
% Milton P. Ferreira Sep/1992.
%
function sum=toep(R1);
n=size(R1);n=n(1,1);
sum=zeros(R1);
for i=-n+1:n-1;
     a=diag(R1,i);
     s=mean(a);
     s1=size(a);
     s1=s1(1,1);
     s2=ones(s1,1)*s;
     sum=sum+diag(s2,i);
end;
```

Appendix M

```
% Computes the smallest eigenvector by inverse iteration.
%
% Input parameters:
%
% a= matrix from which we want to compute an approximation
% of the smallest eigenvector.
% u= # of iterations.
%
% Output parameter:
%
% x= approximation for the smallest eigenvector.
%
% Milton P. Ferreira Sep/1992.
%
function x=eeig(a,u)
n=size(a);
n=n(1,1);
x=ones(n,1);
for i=1:u;
     t=a\x;
     x=t/norm(t,inf);
end;
x=x/norm(x);
```

LIST OF REFERENCES

1. Fast Order-Recursive Hermitian Toeplitz Eigenspace Technique for Array Processing, M.P. Fargues, Ph.D. dissertation, VPI & SU, Blacksburg, VA, Aug. 1988.

2. Modern Spectral Estimation, Theory and Application, Steven M. Kay, Prentice Hall Signal Processing Series, 1988, pp. 431-433.

3. "Tracking Moving Sources Using the Rank-Revealing QR Factorization," M.P. Fargues, proceedings of the 25th Asilomar Conference on Signal, System and Computer, Pacific Grove, Nov. 1-4, 1991, pp. 292-296.

4. "Rank Revealing QR Factorization," T.F. Chan, Linear Algebra Appl., Vol. 88, No. 89, 1987, pp. 67-82.

5. "Direction of Arrival Estimation Using Rank Revealing QR Factorization," S. Prasad and B. Chandna, IEEE Trans. Signal Processing, Vol. SP-38, No. 5, May 1991, pp. 1224-1229.

6. Numerical Linear Algebra and Optimization, Vol.1, Gill P. E. [et al.], Addison-Wesley Publishing Company, 1991, pp. 26.

7. Matrix Computations, Gene H. Golub and Charles F. Van Loan, 1983, The Johns Hopkins University Press.

8. "Deflated Decomposition of Solutions of Nearly Singular Systems," T.F. Chan, SIAM J. Numer. Anal., 21(4), Aug. 1984, pp. 738-754.

9. Probability and Statistics for Engineering and the Sciences, Jay L. Devore, 1991, Brooks/Cole Publishing Company.

10. "Incremental Condition Estimation," C.H. Bischof, SIAM J. Matrix Anal. Appl., Vol. 2, Apr 1990, pp. 312-322.

11. "Estimating the Angles of Arrival of Multiple Plane Waves," Ramdas Kumaresan, Donald W. Tufts, IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-19, No. 1, Jan 1983, pp. 134-138.

12. "Statistical Performance Analysis of the Minimum Norm Method," Bhaskar D. Rao and K.V.S. Hari, IEEE Proceedings ICASSP-89, pp. 2760-3 Vol. 4.

13. "An Updating Algorithm for Subspace Tracking," G. W. Stewart, University of Maryland Computer Science Technical Report CS-TR 2494, 1990, also in IEEE Trans. on Signal Processing, Vol. 40, No. 6, June 1992, pp. 1535-1541.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code EC Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
4.	Professor Monique Fargues, Code EC/Fa Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	2
5.	Professor Ralph Hippenstiel, Code EC/Hi Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
6.	Professor Harold Titus, Code EC/Ts Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
7.	Diretoria de Armamento e Comunicacoes da Marinha Brazilian Navy Rua Primeiro de Marco, 118, 21 andar CEP 20010, Rio de Janeiro, RJ, Brasil	2
8.	LCdr Milton Ferreira Brazilian Navy Rua Primeiro de Marco, 118, 21 andar CEP 20010, Rio de Janeiro, RJ, Brasil	1