W.W. GAERTNER RESEARCH INC.

AD A0 66635

9 FINAL COMPREHENSIVE REPORT.

1 sc

6 ANGULAR LOCATION OF MULTIPLE SIGNAL SOURCES - A DIGITAL APPROACH.

LEVEL II

SPONSORED BY:
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VA 22217

12 110p.

DATE:	11 7 NOVEMBER 1977
CONTRACT NO.:	15 N00014-77-C-0342 new
DATE OF CONTRACT:	77 APRIL 01
NAME OF CONTRACTOR:	W. W. GAERTNER (WW) RESEARCH, INC. 205 SADDLE HILL ROAD STAMFORD, CONNECTICUT 06903 (203) 322-3673
PROJECT SCIENTIST:	DR. S. S. REDDI

388 544 Gm

TABLE OF CONTENTS

SUMMARY

This report describes the research ~~conducted~~ under the Office of Naval Research Contract No. N0014-77-C0342 to investigate algorithms for locating simultaneously multiple signal sources/jammers using linear antenna arrays and digital signal processing. Specifically, the signals impinging on each of the antenna elements are converted to base band and digitized. The digital samples from all the antenna elements are fed to a computer which carries out direction finding, and later beam forming and signal extraction. (Figure S-1).

The problem of multiple signal source location with a linear array ~~can be~~ is formulated. ~~as~~ follows. Consider the linear array with n isotropic antenna elements spaced at intervals of d units. The array is operated in a narrowband mode at a center frequency whose wavelength is γ. There are m signal sources/jammers $J_1, J_2, \ldots, J_m$ located with inclinations of $\theta_1, \theta_2, \ldots, \theta_m$ with the antenna normal (Figure S-2) and complex envelope amplitudes $s_1, s_2, \ldots, s_m$. Let the complex envelope amplitude of the antenna internal noise generated by the ith antenna element be g_i. Assume $E\{g_i\} = 0$, $E\{g_i^* g_j\} = \delta_{ij} g^2$ and $E\{g_i^* s_j\} = 0$. Also assume for simplicity $E\{s_i\} = 0$ and $A = ||E\{s_i^* s_j\}||$. Let z_i be the combined signal output of antenna i due to the signal sources and internal antenna noise and $C = ||E\{z_i^* z_j\}||$. Given C the problem is to evaluate the number of signal sources and their angular locations. (Once this problem is solved, it is a simple matter to extend to the case where two mutually perpendicular linear arrays are used to determine azimuth and elevation angles of signal sources).

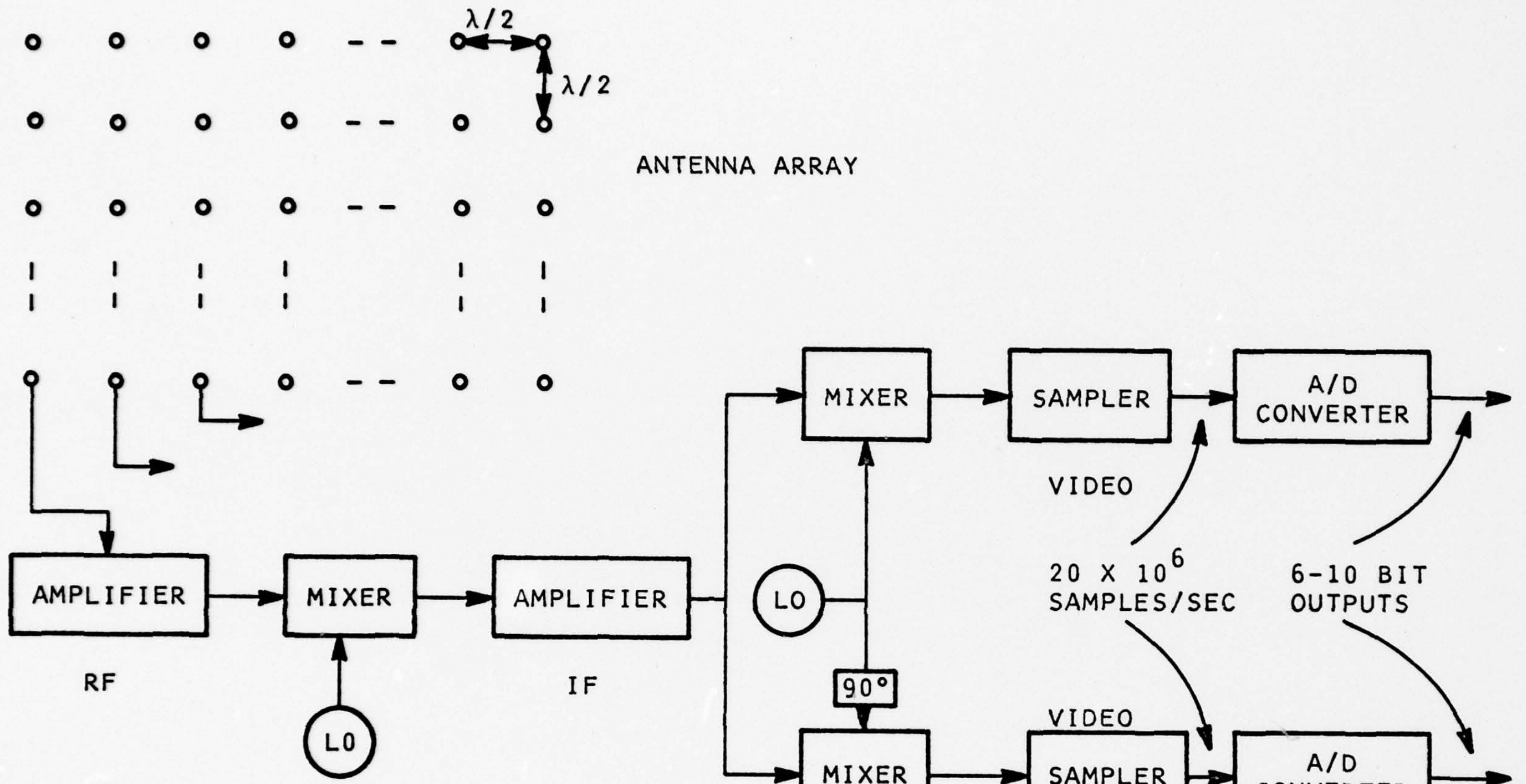


Figure S-1. Block diagram of radar receiver system with digital signal processing.

W.W. GAERTNER RESEARCH INC.

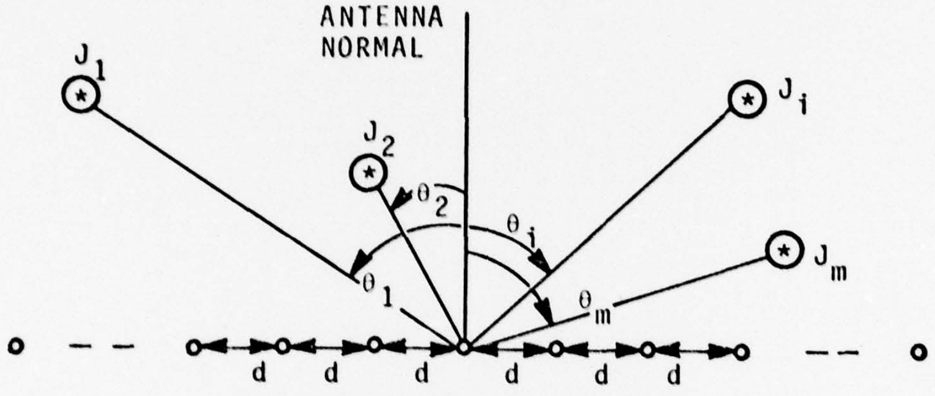


Figure S-2. Geometric details of the linear array and the noise/signal sources.

In order to solve the problem of multiple source locations, a theoretical basis is developed using the properties of Vandermonde determinants and vectors and several algorithms are formulated utilizing this basis. Basically these algorithms consist of three computational phases. In the first phase, eigenvalues/eigenvectors of a complex covariance matrix formed from the sampled output signals of the elements of the linear array are computed. In the second phase, a polynomial is formed from the eigenvectors computed in the first phase and in the final phase the polynomial is solved on the unit circle to find the angular locations of the signal sources.

The proposed algorithms are applicable to cases where the source signal covariance matrix A is singular as well as non-singular. The source signal covariance matrix is singular, for instance, in the presence of the multipath phenomenon. Using a first order perturbation analysis, the effects of errors due to quantization and unidentical variances and non-zero covariances of antenna internal noise signals are discussed and a method is suggested to minimize these effects. Also an upper bound is derived on the computational errors in the estimates of angular locations.

A computational analysis is presented to show that the proposed algorithms have a computational complexity of $O(n^3)$ multiplication. The most complex and generalized version of the proposed algorithms requires approximately 175 million multiplications for n = 100 and around 50 million multiplications for n = 60. Thus if angular location of targets has to be performed every 10 milliseconds, we require a processing rate of approximately 17500 and 5000 million multiplications/second for n = 100 and 60 respectively. (These rates have to be multiplied by 2 if elevation and azimuth angles

of the sources are required). In view of the present technological trend in computing, it is feasible to implement the algorithms for real time angular location on high speed parallel processing computers within the next decade when n is in the range 10 - 100.

Simulations are performed to investigate the behavior of the algorithms in terms of their accuracy, resolution and convergence. The signal sources/jammers and the internal noises are simulated by generating independent normally distributed random numbers. Specifically experiments were conducted to illustrate that in general as the number of samples and/or the number of antenna elements are increased, accuracy of angular estimates improves. Also, the experiments illustrate that higher number of elements and/or higher signal power levels result in better angular resolution of signal sources. Preliminary experimental results indicate that for four antenna elements, for widely separated signal sources, an accuracy of approximately 10 milliradians can be expected after 30-40 samples. Also for four elements, a resolution of approximately 7 degrees may be expected with the source signal to antenna internal noise ratios around 10 db.

In order to assess the effectiveness of the algorithms under real life conditions, actual test data tapes were obtained from the Naval Research Laboratory and the algorithms were applied to process data. These tapes contained covariance matrices obtained from a 4-element linear array. The number of sources varied from 0 to 10 and an analog-to-digital conversion scheme with 512 samples was used. To compute the covariance matrices, 1024 samples were used.

The covariance matrices for the cases where the number of sources is from 1 to 3 were processed. The results are encouraging for one and two sources (except for certain two source cases) and the angular errors are below 5 degrees. However for three sources, the algorithm performs less satisfactorily in terms of resolution coalescing two or three sources which are separated approximately by 10 degrees into one. This is mostly due to the reasons that the data is accurate only up to the second decimal digit and that there are only four antenna elements.

The proposed entirely digital approach to multiple source location (as opposed to the hybrid analog-digital approach developed by Berni and Swarner of the Ohio State University[1]) should be of interest to radar system engineers who are faced with the problem of detecting targets in the presence of heavy jamming and to electronic navigation specialists.

[1] Swarner, W.G. and Berni, A.J., An Adaptive Antenna Array for Angle of Arrival Estimation System for Sensor Communications, Report No. ESL3435-2, Electro Science Laboratory, Department of Electrical Engineering, The Ohio State University, Columbus, Ohio. Also see Berni, A.J., "Angle of Arrival Estimation Using An Adaptive Antenna Array", IEEE Trans. Aerospace and Electronic Systems, Vol. AES-11, No. 2, March, 1975.

1. INTRODUCTION

This report describes the results of a research study conducted to find and explore algorithms for angular location of multiple signal sources employing digital signal processing and linear antenna arrays. A theoretical basis is presented using the properties of Vandermonde determinants and vectors and several algorithms are developed using the basis for simultaneous angular location of multiple signal sources. For the algorithms it is assumed that the internal antenna noise signals in the individual elements of the linear array have identical variances and are uncorrelated. Basically the algorithms consist of determining the eigenvalues/eigenvectors of a complex covariance matrix obtained from the sampled output signals of the linear antenna array and solving a polynomial equation whose coefficients are obtained from the eigenvectors.

The proposed algorithms are applicable to cases where the source signal covariance matrix is singular as well as nonsingular. (The source signal covariance matrix is singular when the multipath phenomenon is present). The effects of errors due to quantization and unidentical variances and nonzero covariances of antenna internal noise signals are discussed using a first order perturbation analysis. A computational analysis is presented to show that the proposed algorithms have a complexity of $O(n^3)$ multiplications where n is the number of elements in the linear array and that it is feasible to implement them on high speed parallel processing computers for real time angular location in the next decade. Simulations are performed to investigate the behavior of the proposed algorithms in terms of their accuracy, convergence and angular resolution. Actual test data was obtained from the Naval Research Laboratory to test the algorithms and the results are reported. The report is concluded with suggestions for future research in this area.

2. GENERAL THEORY OF MULTIPLE SOURCE LOCATION

The problem of simultaneous angular location of multiple sources with a linear antenna array can be formulated as follows. Consider the linear array with n isotropic antenna elements spaced at intervals of d units. The array is operated in a narrow band mode at a center frequency whose wavelength is ν. There are m signal sources/jammers $J_1, J_2, \ldots, J_m$ located with inclinations of $\theta_1, \theta_2, \ldots, \theta_m$ with the antenna normal(Fig. 2-1) and complex envelope amplitudes $s_1, s_2, \ldots, s_m$. Let the complex envelope amplitude of the internal noise generated by the ith antenna element be g_i. The following first and second order statistics are assumed for the signal and antenna noise waveforms:

$$E\{g_i\} = 0$$

$$E\{g_i^* g_j\} = \delta_{ij} g^2$$

$$E\{g_i^* s_j\} = 0$$

Assume for simplicity $E\{s_i\}=0$ and let A be the source signal covariance matrix:

$$A = ||a_{ij}|| = ||E\{s_i^* s_j\}||$$

Associate a direction vector ξ_i with J_i as follows:

W. W. GAERTNER RESEARCH INC.

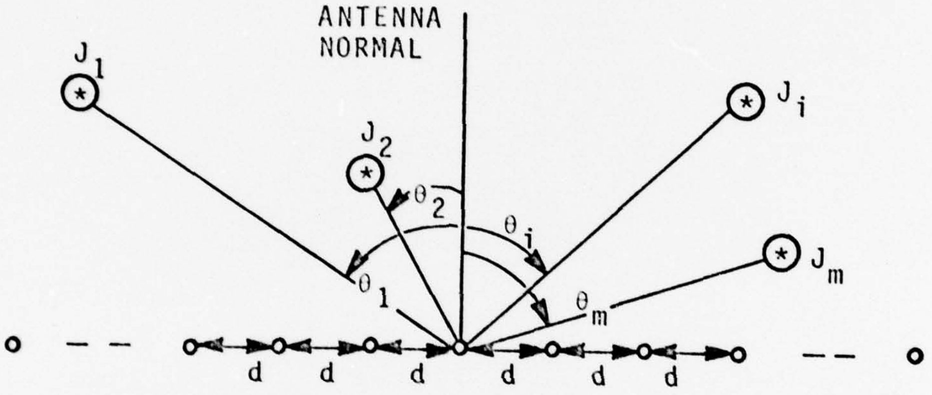


Figure 2-1. Geometric details of the linear array and the noise/signal sources.

$$\xi_i = \begin{bmatrix} 1 \\ \exp\{j(2\pi d/\nu)\sin\theta_i\} \\ \exp\{j(4\pi d/\nu)\sin\theta_i\} \\ \vdots \\ \exp\{j(n-1)(2\pi d/\nu)\sin\theta_i\} \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ \sigma_i \\ \sigma_i^2 \\ \vdots \\ \sigma_i^{n-1} \end{bmatrix}$$

Let $z^T = (z_1, z_2, \ldots, z_m)$ where z_i is the combined signal output of antenna i due to the signal sources and internal antenna noise and $g^T = (g_1, g_2, \ldots, g_m)$. Then z can be written as:

$$z = g + \sum_i s_i \, \xi_i^*$$

and the covariance matrix of the combined signal vector as:

$$C = ||E\{z_i^* z_j\}||$$

$$= g^2 I + \sum_{i,j} a_{ij} \xi_i \xi_j^\dagger \qquad (1)$$

Given C the problem is to evaluate the number of signal sources and their angular locations using the relation (1).

To solve this problem we make use of certain remarkable properties of the vectors of the form $(1, \alpha, \alpha^2, \ldots, \alpha^{k-1})$ referred to as Vandermonde vectors in the following. (Note the direction vectors are Vandermonde vectors.) We denote such vectors by $V_k(\alpha)$ or simply $V(\alpha)$. We need the following properties of Vandermonde vectors for our development.

Lemma 1: Vectors $V_k(\alpha_i)$, $i = 1, 2, \ldots, \ell$, are linearly independent for $k \geq \ell$ and $\alpha_i \neq \alpha_j$, $i \neq j$.

The lemma is a consequence of the properties of Vandermonde determinants (NERI 67). This simple lemma has certain interesting consequences. Let $\{x_1, x_2, \ldots, x_\ell\}$ be a set of linearly independent vectors with k components, $k \geq \ell$, and S be the space by these vectors. Assume that S possesses a basis V which consists entirely of Vandermonde vectors, $V = \{V_k(\alpha_1), V_k(\alpha_2), \ldots, V_k(\alpha_\ell)\}$. Such a basis will be referred to as Vandermonde basis and it can be seen from Lemma 1 that if a Vandermonde basis exists, it is unique.

Theorem 1: Let $\{X_1, X_2, \ldots, X_\ell\}$ be a set of linearly independent k-tuples with Vandermonde basis $\{V_k(\alpha_1), V_k(\alpha_2), \ldots, V_k(\alpha_\ell)\}$, $k > \ell$. Let

$$F = [X_1 | X_2 | \ldots | X_\ell] = \begin{bmatrix} w_1^T \\ \hline w_2^T \\ \hline \vdots \\ \hline w_k^T \end{bmatrix} = \begin{bmatrix} W \\ \hline w_k^T \end{bmatrix}$$

Construct polynomials:

$$G(x) = w_k^T \, (W^+W)^{-1}W^+V_{k-1}(x) - x^{k-1}$$

$$g_i(x) = w_i^T \, (W^+W)^{-1}W^+V_{k-1}(x) - x^{i-1},$$

$$i = 1, 2, \ldots, k-1$$

Then $f(x) = \text{g.c.d.}\{G(x), g_1(x), \ldots, g_{k-1}(x)\}$ possess $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ as its only roots.

Proof: The existence of Vandermonde basis implies the existence of ℓ $(\ell+1)$-tuples $(\alpha, y_1, y_2, \ldots, y_\ell)$ satisfying the following k nonlinear equations:

$$V_k(\alpha) = \sum_i y_i X_i$$

i.e. $$V_k(\alpha) = Fy \qquad (2)$$

where $y = (y_1, y_2, \ldots, y_\ell)^T$. Rewrite (2) as:

$$Wy = V_{k-1}(\alpha)$$

$$w_k^T y = \alpha^{k-1} \qquad (3)$$

Since a consistent solution y exists, y can be written as $y = (W^\dagger W)^{-1} W^\dagger V_{k-1}(\alpha)$ and hence (3) becomes

$$G(\alpha) = w_k^T (W^\dagger W)^{-1} W^\dagger V_{k-1}(\alpha) - \alpha^{k-1}$$

$G(\alpha)$ possesses as its roots $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ and $(k - \ell - 1)$ extraneous roots. To eliminate the extraneous roots, we note for the desired roots:

$$Wy = V_{k-1}(\alpha)$$

i.e., $$W(W^\dagger W)^{-1} W^\dagger V_{k-1}(\alpha) = V_{k-1}(\alpha)$$

$$(W(W^\dagger W)^{-1} W^\dagger - I)\, V_{k-1}(\alpha) = 0$$

from which one can obtain polynomials $g_1(\alpha), g_2(\alpha), \ldots, g_{k-1}(\alpha)$. Thus $f(x) = \gcd\{G(x), g_i(x)\}$ has as its roots $\alpha_1, \alpha_2, \ldots, \alpha_\ell$. It can be seen that if α^* is a root of $f(x)$, then y can be found so that $Fy = V_k(\alpha^*)$ and in view of Lemma 1, $\alpha^* \epsilon \{\alpha_1, \alpha_2, \ldots, \alpha_\ell\}$. Thus the only roots of $f(x)$ are $\alpha_1, \alpha_2, \ldots, \alpha_\ell$.

We refer to $G(x)$ as the principal polynomial and $g_i(x)$ as auxiliary polynomials in the sequel.

Corollary 1: For $k = \ell + 1$, $g_i(x) = 0$ and hence $f(x) = G(x)$. Hence a set of l.i. $(\ell+1)$-tuples $\{X_1, X_2, \ldots, X_\ell\}$ always possess a unique Vandermonde basis (for $f(x)$ is of degree ℓ).

The corollary is interesting since it uniquely characterizes ℓ vectors by means of an ℓ-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_\ell)$.

Corollary 2: If vectors $X_1, X_2, \ldots, X_\ell$ are orthonormal, then

$$G(x) = w_k^T W^\dagger V_{k-1}(x) - (1-\lambda)x^{k-1}$$

$$g_i(x) = w_i^T [I + (1-\lambda)^{-1} w_k^* w_k^T] W^\dagger V_{k-1}(x) - x^{i-1}$$

where $\lambda = |w_k|^2$.

This follows by noting $W^\dagger W = (I - w^* w^T)$ and applying Sherman-Morrison formula (DAHL 74) to obtain its inverse.

The implication of Corollary 2 is that it is computationally simpler to obtain $f(x)$ when the vectors $X_1, X_2, \ldots, X_\ell$ are orthonormal.

2.1 Multiple Source Location for Non-Singular Source Signal Covariance Matrix

The problem of multiple source location when the source signal covariance matrix A is non-singular is simpler and will be considered first. We show that there are exactly m eigenvalues of C which are greater than g^2 and that for their corresponding m eigenvectors, the m source direction vectors form a basis. Then we apply Theorem 1 to obtain the direction (Vandermonde) vectors and consequently angular locations.

Consider the combined signal covariance matrix C which can be written as:

$$C = g^2 I + \sum_{i,j} a_{ij} \xi_i \xi_j^\dagger$$
$$= g^2 I + \Xi A \Xi^\dagger$$

where $\Xi = [\xi_1 | \xi_2 | \xi_3 | \dots | \xi_m]$. To show that C has m eigenvalues greater than g^2, we show that the received signal covariance matrix $B = \Xi A \Xi^\dagger$ has m positive eigenvalues and n-m zero eigenvalues. We also show that the eigenvectors of B (and hence C) can be expressed as $\sum_i z_i \xi_i$ where $z^T = (z_1, z_2, \dots, z_m)$ is an eigenvector of $A\Xi^\dagger\Xi = A\Lambda$.

Let $\eta = \sum_i z_i \xi_i$. In order that η is an eigenvector of B, we should have:

$$B\eta = \lambda\eta$$

i.e., $\Xi A \Xi^{\dagger} \Xi z = \lambda \Xi z$

$$\Xi A \Lambda z = \lambda \Xi z$$

$$\Xi (A\Lambda - \lambda I) z = 0$$

$\Longleftrightarrow \quad (A\Lambda - \lambda I) z = 0$

Thus the eigenvalues of B are the same as those of $A\Lambda$. It may be noted that Λ is Gramian of Ξ (GANT 60) and has a rank m and since A also has a rank m (for it is non-singular) it follows that B has m nonzero eigenvalues. It also follows that these eigenvalues are positive since A and Λ^{-1} are both positive definite (WILK 65).

Let $\zeta_1, \zeta_2, \ldots, \zeta_m$ be the eigenvectors of B that have eigenvalues greater than g^2. Since these eigenvectors can be chosen to be orthonormal (for B is Hermitian), we apply Corollary 2 to obtain the direction (Vandermonde) vectors. (Note $n > m$ for the theorem and corollary to be applicable). We first obtain G(x) and then solve the polynomial equation to obtain its roots. In this case since the roots happen to be of the form $\exp\{-j(2\pi d/\nu) \sin\theta\}$, we restrict our search for the roots to points on the unit circle. This simplifies matters considerably. (It is our experience that the extraneous roots of the principal polynomial G(x) rarely lie on the unit circle. Thus the auxiliary polynomials and their roots are not usually computed. Later on we consider the entire set of polynomials $\{G(x), g_i(x)\}$ as an overdetermined nonlinear system (in one variable) and compute least squared error solutions.)

Example 1: Five antenna elements are assumed spaced at half wavelengths. To simplify matters the covariance matrix of antenna internal noise is assumed to be an identity matrix. Three sources are simulated at inclinations of 0, .1π and -.1π to the antenna normal. The source signals are assumed to be samples from white Gaussian noise and generated by using Box-Muller's transformation (DAHL 74). The signal variances are assumed to be 5 power units. After 10 samples, the received signal covariance matrix is evaluated and an identity matrix is added to obtain the combined signal covariance matrix. The eigenvalues of the combined signal covariance matrix are 32.8, 14.34, 8.67, 1.0, 1.0 and the eigenvectors corresponding to eigenvalues having a value greater than 1 are:

$$X_1 = \begin{bmatrix} .02368 + j.37984 \\ -.09574 + j.11358 \\ -.26674 - j.33110 \\ -.34044 - j.56700 \\ -.25265 - j.38871 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} -.42832 - j.38035 \\ -.53024 - j.37964 \\ -.41012 - j.17065 \\ -.17255 + j.06464 \\ -.02439 + j.12137 \end{bmatrix}$$

$$X_3 = \begin{bmatrix} -.52973 + j.03854 \\ .16043 - j.01170 \\ .40649 - j.0096 \\ -.00581 + j.04302 \\ -.71746 + j.10033 \end{bmatrix}$$

The principal polynomial G(x) is computed as:

$$G(x) = .19458 - .16938x - .10721x^2 + .32695x^3 - .24493x^4$$

There are three roots $e^{j\theta}$, θ = 0, .97081, 5.31238, for which G(x) vanishes and the three angular locations are obtained with seven digit accuracy.

Now we consider the situation where there is 'noise' present in the system of non-linear equations due to such factors as receiver noise and the non-whiteness of antenna noise. In this case the desired roots of G(x) and $g_i(x)$ do not exactly lie on the unit circle and we seek to find the least squared error solution satisfying the system of non-linear equations. We write the system of nonlinear equations as the following matrix equation:

$$FV_k(x) = 0$$

We minimize $\phi = V_k^\dagger(x)F^\dagger F V_k(x)$ subject to the constraint that x lies on the unit circle. Let $x = e^{j\theta}$ and $F^\dagger F = ||q_{ik}e^{j\theta}ik||$.

Then $\phi = \sum_{i,k} q_{ik}\exp\{j[\theta_{ik}-(i-k)\theta]\}$. To find minima, we set $\partial\phi/\partial\theta = 0$ and $\partial^2\phi/\partial\theta^2>0$ to obtain:

$$\phi'(\theta) = 2 \sum_{i>k} q_{ik}(i-k)\sin(\theta_{ik}-(i-k)\theta) = 0 \text{ and}$$

$$\phi''(\theta) = -2 \sum_{i>k} q_{ik}(i-k)^2\cos(\theta_{ik}-(i-k)\theta)>0.$$

In next section, we explain a method which does not entirely rely on the values of eigenvalues.

2.2 Multiple Source Location for Singular Source Signal Covariance Matrix

The problem of multiple source location becomes complex if the source signal covariance matrix A happens to be singular. The singularity of A may be because of the multipath phenomenon or the deliberate enemy control and transmission of jamming or signal waveforms. The complexity of the problem is due to the fact that there will be only $m'<m$ eigenvalues of C which are greater than g^2 and the corresponding m' eigenvectors do not form a basis for the m source direction vectors; here m' is the rank of A. Our approach to the problem is to generate eigenvectors for different covariance matrices obtained by considering n,n+1,... antenna elements and select independent vectors from these sets of eigenvectors. These vectors form a basis for the source direction vectors and hence Theorem 1 can be applied to obtain angular locations. It should be noted that this approach is applicable when A is non-singular and the eigenvalues of $A\Lambda$ are small and comparable to g^2, thus making it difficult to apply the algorithms suggested in the preceding section.

Consider:

$$C = g^2 I + \Xi A \Xi^{\dagger}$$

We show that C has m' eigenvalues greater than g^2 where $m' = \text{rank}(A)$ by proving that $B = \Xi A \Xi^{\dagger}$ has m' postive eigenvalues and $n-m'$ zero eigenvalues. As before, we can prove that $\sum_i z_i \xi_i$ is an eigenvector of B with an eigenvalue of λ if $(z_1, z_2, \ldots, z_m)^T$ is an eigenvector of $A\Lambda$ with an eigenvalue of λ. Since A is of rank m' and Λ has a rank m, there are m' non-zero positive eigenvalues of $A\Lambda$ and consequently m' positive eigenvalues of B. Also, one may note that any eigenvalue of B which has a non-zero eigenvalue is expressible as $\sum z_i \xi_i$.

Let $\psi_1, \psi_2, \ldots, \psi_{m-m'}$ be the vectors $\sum_i z_i \xi_i$ where $(z_1, z_2, \ldots, z_m)^T$ is in the null space of $A\Lambda$ and $\phi_1, \phi_2, \ldots, \phi_{n-m}$ be the vectors orthogonal to the m direction vectors $\xi_1, \xi_2, \ldots, \xi_m$. Also let $\rho_1, \rho_2, \ldots, \rho_{n-m'}$ be the eigenvectors of B (C) with an eigenvalue of zero (g^2). Then it follows $\{\psi_1, \psi_2, \ldots, \psi_{m-m'}, \phi_1, \phi_2, \ldots, \phi_{n-m}\}$ forms a basis for $\{\rho_1, \rho_2, \ldots, \rho_{n-m'}\}$. It should be noted that ψ_i is orthogonal to ϕ_j, $i = 1, 2, \ldots, m-m'$ and $j = 1, 2, \ldots, n-m'$.

Now we consider the situation where two sample covariance matrices C_n and C_{n+1} are generated from n and n+1 antenna elements and let the gramians of the sets of direction vectors be Λ_n and Λ_{n+1} respectively. It is clear that $\Lambda_{n+1} = \Lambda_n + \Sigma^{+}\Sigma$ where $\Sigma = (\sigma_1^n, \sigma_2^n, \ldots, \sigma_m^n)$ and hence that the eigenvectors of C_n and C_{n+1} are in general different. Because of the nature of the direction vectors it is also clear that the two vectors v_1 and v_2 formed from the first and last n components of an eigenvector of C_{n+1} with an eigenvalue greater than g^2 belong to the linear space spanned by the m direction vectors (with n components). This suggests our algorithm. Generate eigenvectors for C_n and C_{n+1} and select those eigenvectors whose eigenvalues are greater than T ($>g^2$). From the thus selected set of eigenvectors of C_{n+1}, generate two sets of vectors V_1, V_2 by choosing the first and last n components from each eigenvector. Augment the selected eigenvectors of C_n with vectors from V_1 and V_2 using Gram-Schmidt orthogonalization procedure. Once this is done we apply the techniques developed in 2.1 to obtain estimates for angular locations.

2.3 Effect of Errors on Computational Accuracy

There are two primary sources of error which affect accuracy of estimates of angular locations. The first source is the A/D converter which introduces errors due to finite levels of quantization. The second source is the antenna elements whose internal noise waveforms may not be uncorrelated white Gaussian noises with identical variances after a finite number of samples. The effect of this error is that the antenna internal noise covariance matrix will not be a diagonal matrix with identical elements but a matrix with unequal diagonal elements and non-zero off-diagonal elements.

A simple first order perturbation analysis (WILK 65) yields useful insight into the effect of errors. Initially, we will only consider the effects of errors on eigenvectors and show how to minimize them. Let A be a Hermitian matrix with eigenvectors $x_1, x_2, \ldots, x_n$ whose corresponding eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_n$. For the perturbed matrix $(A + \varepsilon B)$ let $x_i(\varepsilon)$ and $\lambda_i(\varepsilon)$, $i = 1, 2, \ldots, n$ be the eigenvectors and eigenvalues. For a first order perturbation analysis we assume $\lambda_i(\varepsilon) = \lambda_i + k_i\varepsilon$ and $x_i(\varepsilon) = x_i + \varepsilon \{ \sum_{i \neq j} s_{ij}x_j \}$. Then it follows:

$$(A + \varepsilon B)x_i(\varepsilon) = \lambda_i(\varepsilon)x_i(\varepsilon)$$

collecting first order terms in ε we have:

$$A\{\sum_{j\neq i} s_{ij}x_j\} + Bx_i = \lambda_i\{\sum_{j\neq i} s_{ij}x_j\} + k_ix_i$$

$$\sum_{i\neq j} s_{ij}(\lambda_j - \lambda_i)x_j + Bx_i = k_ix_i$$

Since for a Hermitian matrix we can choose $x_1, x_2, \ldots, x_n$ to be orthonormal we have

$$s_{ij} = x_j^\dagger Bx_i/(\lambda_i - \lambda_j)$$

i.e., $$x_i(\varepsilon) = x_i + \varepsilon \sum_{j\neq i} \beta_{ji}x_j/(\lambda_i - \lambda_j)$$

where $\beta_{ij} = x_i^\dagger Bx_j$. Thus the eigenvectors are sensitive to the separation of eigenvalues.

In our case it may be noted that we are interested in eigenvectors that are linear combinations of direction vectors and concerned only when the vectors orthogonal to the direction vectors can contaminate the desired eigenvectors. This immediately suggests that in general the larger the eigenvalue, the more accurate its corresponding eigenvector will be. Thus in the algorithm described in 2.2 where multiple eigenvector sets are computed and a threshold T is set to select eigenvectors, T reflects the accuracy of eigenvectors containing the desired information.

Now we consider the effects of errors on the roots of the principal polynomial G(x) using first order perturbation analysis. We assume that the obtained eigenvectors of the combined signal covariance matrix C is orthonormal and Corallary 2 is applied to find G(x). We indicate the first order errors in the eigenvectors as

$$F + \varepsilon F_e = \left[\begin{array}{c} W + \varepsilon W_e \\ \hline w_k^T + \varepsilon w_{ke}^T \end{array}\right]$$

and in λ as $\lambda + \varepsilon\lambda_e$ and obtain the following perturbed principal polynomial G_p:

$$G_p(x) = G(x) + \varepsilon G_e(x)$$

where $G_e(x) = (w_{ke}^T W^\dagger + w_k^T W_e^\dagger)\, V_{k-1}(x) + \lambda_e x^{k-1}$. Note the coefficients of $G_e(x)$ cannot exceed in magnitude 2ℓ. Now we consider the roots of $G_p(x)$ and their relation to those of G(x). We have by Taylor's theorem

$$\begin{aligned} G_p(x+\Delta) &= G(x+\Delta) + \varepsilon G_e(x+\Delta) \\ &\simeq G(x) + \Delta G'(x) + \varepsilon G_e(x) \\ &\quad + \varepsilon\Delta G_e'(x) \end{aligned}$$

If x_R and $x_R+\Delta$ are roots of G(x) and $G_p(x)$ respectively, we have

$\Delta = -\varepsilon G_e(x_R)/[G'(x_R) + \varepsilon G_e'(x_R)]$ and since $|x_R| = 1$,
$\Delta \leq 2k\ell\varepsilon/G'(x_R)$.

2.4 Computational Analysis for Multiple Source Location

In this section we perform a computational analysis of the algorithms presented in preceding sections in order to assess the feasibility of implementing digital multiple source location in real time. These algorithms have the following three phases of computation:

1. Eigenvalue/Eigenvector Computation

2. Polynomial Formation

3. Polynomial Solution on the Unit Circle

For computational analysis we select two specific algorithms which are described in the following. In the first algorithm only one eigenvalue/eigenvector computation is performed (for an nxn Hermitian matrix) and the principal polynomial G(x) is computed and solved on the unit circle. To find the zeros of the principal polynomial on the unit circle, the quantities $|G[\exp(jk\theta)]|^2$, $\theta = 2\pi/K_D$, $k = 0, 1, \ldots, k_D-1$, are computed to locate the approximate ranges of the minima of $|G(x)|^2$. Then a Fibonaccian scheme with K_F searches (ORTE 70) is conducted to locate minima of $|G(x)|^2$ and hence zeros of G(x).

In the second algorithm which is more general than the first, eigenvalue/eigenvector computations are performed for nxn and (n+1) x (n+1) Hermitian matrices. Eigenvectors whose eigenvalues exceed a preset threshold $T(>g^2)$ are selected and Gram-Schmidt orthogonalization procedure is performed. Then principal and auxiliary polynomials are computed and least squared error solutions are determined (Section 2.1).

For eigenvalue/eigenvector computations, one may use Householder's transformations to tri-diagonalize the Hermitian matrix and apply QR transformations to obtain eigenvalues/eigenvectors (WILK 65, DAHL 74). Assuming two QR-iterations per eigenvalue (DAHL 74) an eigenvalue/eigenvector computation for an nxn Hermitian matrix requires $20n^3/3 + O(n^2)$ complex multiplications or approximately $28n^3$ real multiplications. For polynomial solution in the first algorithm, we require approximately $(K_D + K_F)n$ complex or $4(K_D + K_F)n$ real multiplications. Assuming that the algorithm has to be performed R times every second, we require a processing rate of $[28n^3 + 4(K_D + K_F)n]R$ multiplications/second for real time implementation. (We retain the linear term in n since K_D is usually large, say around 1000). Figure 2.4-1 shows this processing rate in MMPS (million multiplications/second) against the number of elements for $K_D = 1000$, $K_F = 60$ and R = 1, 10, 100, 1000.

For the second algorithm a similar computational analysis reveals that we require approximately $[68n^3 + (K_D + K_F) \cdot (2 + k_{trig})\, n^2]R$ real multiplications/second if the algorithm has to be performed R times every second. k_{trig} is the ratio of computing times to perform the sine function and multiplication. Figure 2.4-2 shows the required processing rate against the number of elements for $K_D + K_F = 1060$, $k_{trig} = 8$ and R = 1, 10, 100, 1000.

From our computational analysis and the present and future trends of computing technology (TURN 74), it is clear that real time implementation of digital multiple source location procedures is feasible within this decade. (The only technological obstacle may lie in economic construction of A/D converters with required precision). Also the algorithms underlying the procedures are amenable to

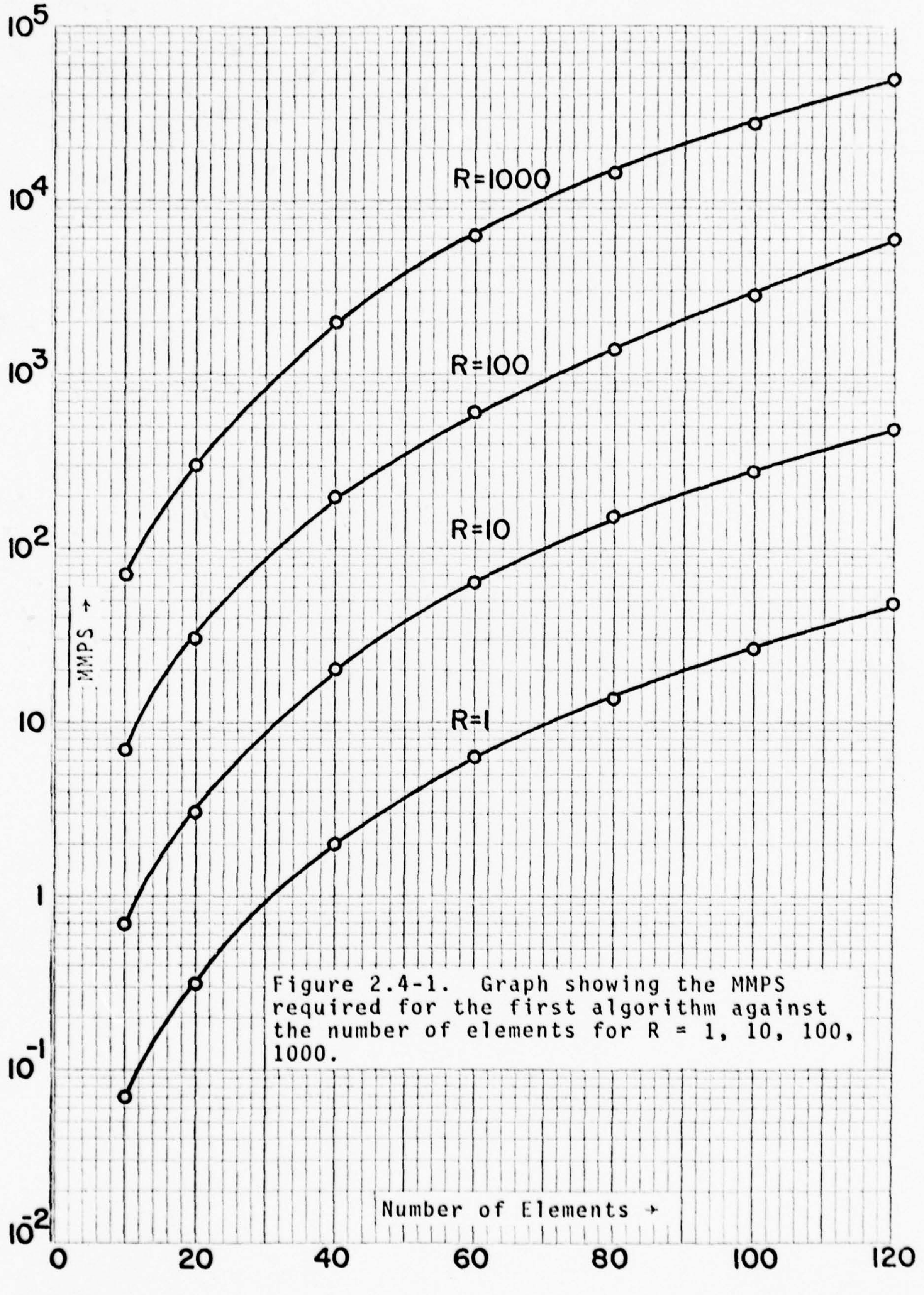


Figure 2.4-1. Graph showing the MMPS required for the first algorithm against the number of elements for R = 1, 10, 100, 1000.

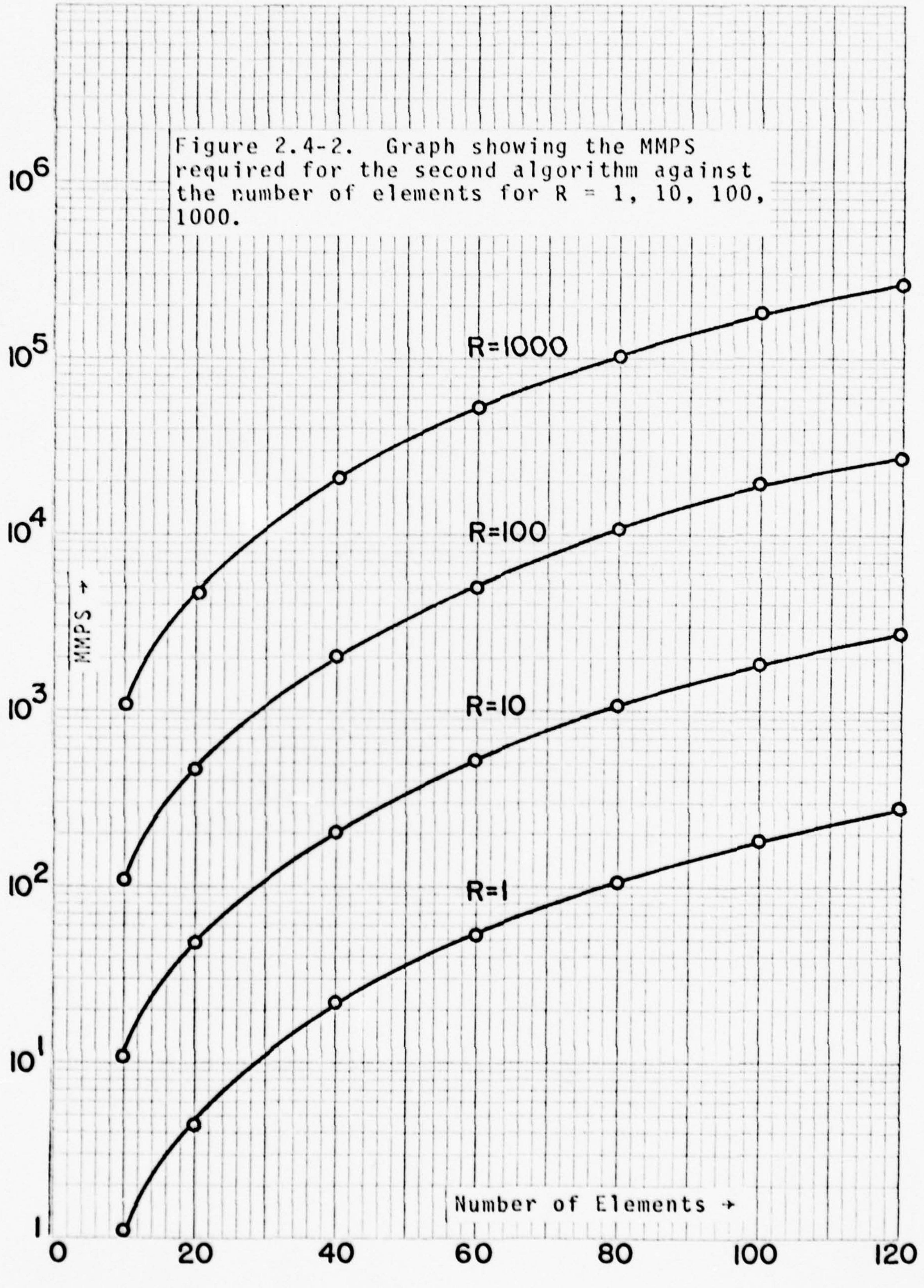


Figure 2.4-2. Graph showing the MMPS required for the second algorithm against the number of elements for R = 1, 10, 100, 1000.

parallel processing and Appendices 2 and 3 show how to exploit parallelism inherent in matrix operations on a particular parallel processing computer described in Appendix 1 to speed up computation. A simple computational analysis shows that for the algorithms proposed for multiple source location, a speed up proportional to p may be expected using parallel processing where p is the number of processors in the computing system and is in the range 20 - 100.

3. COMPUTER IMPLEMENTATION AND NUMERICAL EXPERIENCE

A FORTRAN program was written to implement the multiple source problem on a PDP-11/35. Only the first algorithm described in 2.4 has been completely implemented and in this section we report on the numerical experience gained by using the algorithm. The signal sources/jammers and the internal antenna noises are simulated by generating independent normally distributed random numbers (using Box-Muller's transformation on uniformly distributed numbers in [0, 1] (DAHL 74)) with zero mean and prespecified variances. The program was written in double precision arithmetic (56 bits of mantissa, 8 bits of exponent) and can handle up to 16 antenna elements and 10 sources. A listing of the program is attached as Appendix 4 of this report.

Various numerical experiments were conducted to observe the behavior of the algorithm in terms of its accuracy in determining signal source locations. In the first experiment, a linear array of 4 antenna elements with interelement spacing of half wavelengths is assumed and two sources are simulated at angular locations $.25\pi$ and $-.3\pi$ radians. The signal source variances are assumed to be 10 power units and the internal antenna noise power in each element is taken to be 1 unit. Table 3-1 shows the accuracies obtained as the number of samples is varied from 10 to 100. The inaccuracies in estimates of angular locations are due to the fact that the internal antenna noise covariance matrix is not an identity matrix as assumed. (In the simulation if the internal noise covariance matrix is assumed to be an identity matrix, exact angular locations were obtained.) Table 3-2 shows the internal antenna noise covariance matrices after 10 and 50 samples and the eigenvalues of these matrices.

Number of Antenna Elements: 4

Antenna Array Interelement Spacing: Half Wavelength

Number of Signal Sources: 2

Angular Locations of Signal Sources in Radians/Pi: .25, -.3

Signal Source Variances: 10, 10

Internal Antenna Noise Variance: 1

Number of Samples	Eigenvalues	Threshold for Selecting Eigenvectors	Estimated Angular Locations in Radians/Pi	Maximum Absolute Angular Error in Milliradians/Pi	Average Absolute Angular Error in Milliradians/Pi
10	31.91, 19.17, 1.28, 0.73	2.5	.2554011, -.2661293	33.871	16.935
20	36.86, 22.08, 2.15, 1.19	2.5	.2577914, -.2958452	7.7914	5.9731
30	39.53, 31.11, 1.98, 1.39	2.5	.2546202, -.2997183	4.6202	2.451
40	43.12, 32.48, 1.88, 1.50	2.5	.2489519, -.2980431	1.9569	1.5025
50	41.83, 34.94, 1.97, 1.50	2.5	.2477441, -.2959968	4.0032	3.1296
100	45.37, 37.32, 2.06, 1.73	2.5	.2490604, -.2966534	3.3466	2.1431

Table 3-1. Experimental results illustrating the effects of number of samples on angular accuracy.

Antenna Internal Noise Covariance Matrix and its Eigenvalues after 10 Samples:

$$\begin{bmatrix} 1.1576 & .0910 -j.0132 & -.3488 +j.0425 & -.0370 +j.1718 \\ & 1.1953 & -.3961 +j.4894 & -.1480 -j.0631 \\ & & 2.1693 & -.1293 +j.2903 \\ & & & 1.5779 \end{bmatrix}$$

Eigenvalues: 2.6448, 1.6267, 1.0229, .8057

Antenna Internal Noise Covariance Matrix after 30 Samples:

$$\begin{bmatrix} 2.0050 & .1037 -j.1166 & .0041 +j.0279 & .2110 +j.0489 \\ & 1.6448 & -.0937 +j.2775 & .0539 -j.1150 \\ & & 1.7531 & -.0865 +j.0112 \\ & & & 1.8685 \end{bmatrix}$$

Eigenvalues: 2.2139, 2.0383, 1.6498, 1.3693

Table 3-2. Antenna internal noise covariance matrices and their eigenvalues after 10 and 30 samples.

In the next experiment, the effect of the number of antenna elements on the accuracy of estimates is investigated. The number of samples is fixed at 30 and three signal sources are simulated at $.1\pi$, $.2\pi$ and $.3\pi$ radians with mean power levels at 10 units. The internal antenna noise power is again fixed at 1 unit. Table 3-3 shows the results obtained as the number of elements is varied from 4 to 8. One can immediately see that increasing the number of antenna elements results in improved estimates of angular locations.

In the final experiment the angular resolution capabilities of the algorithm are observed as the source signal powers and the number of antenna elements are changed. Two signal sources are assumed and the number of samples is fixed at 30. Initially the signal powers are fixed at 10 units and the angular locations at $.1\pi$ and $.2\pi$ radians. Table 3-4 shows the results of this experiment. As the second source is moved closer to the first, the algorithm coalesces both sources into one. (The extraneous minima of $|G(x)|^2$ on the unit circle are marked by asterisks). Thus one can see from the table that when the sources are located at $.1\pi$ and $.13\pi$ or closer, the algorithm cannot resolve the sources satisfactorily. However, increasing the power levels of the sources or the number of elements results in the sources being resolved.

In Table 3-5 we show an example of coalescing two sources into one and the results of choosing various thresholds. Two sources are simulated at .1 and $.12\pi$ with power levels fixed at 10 units. As before the internal antenna noise power is assumed to be 1 unit and a four element linear array is assumed. By setting threshold

Antenna Array Interelement Spacing: Half Wavelength

Number of Signal Sources: 3

Angular Locations of Signal Sources in Radians/Pi: .1, .2, .3

Signal Source Variances: 10, 10, 10

Internal Antenna Noise Variance: 1

Number of Samples: 30

Number of Elements	Eigenvalues	Threshold for Selecting Eigenvectors	Estimated Angular Locations in Radians/Pi	Maximum Absolute Angular Error in Milliradians/Pi	Average Absolute Angular Error in Milliradians/Pi
4	83.15, 46.27, 4.63, 1.51	2.5	.0775283, .1842676, .2890970	22.472	16.369
5	98.70, 49.92, 11.26, 1.56 0.97	2.5	.0902723, .1657553, .2875937	34.245	18.793
6	95.95, 43.80, 28.75, 2.17 2.02, 1.12	2.5	.0944212, .2063342, .3012792	6.334	4.397
7	101.92, 68.80, 38.08, 2.70 2.23, 1.30, 1.11	3.0	.0999310, .2029826, .3015170	2.983	1.523
8	100.75, 64.42, 45.13, 2.20 1.67, 1.39, 1.30, 0.64	3.0	.1008242, .2011048, .2995207	1.105	.803

Table 3-3. Experimental results illustrating the effects of number of antenna elements on angular accuracy.

Antenna Array Interelement Spacing: Half Wavelength

Number of Signal Sources: 2

Internal Antenna Noise Variance: 1

Number of Samples: 30

Number of Elements	Signal Source Variances	Signal Source Angular Locations in Radians/Pi	Eigenvalues	Threshold for Selecting Eigenvectors	Estimated Angular Locations in Radians/Pi	Maximum Absolute Angular Error in Milliradians/Pi	Average Absolute Angular Error in Milliradians/Pi
4	10, 10	.1, .2	53.69, 16.01 2.03, 0.95	2.5	.1115245 .2009900	11.525	6.257
4	10, 10	.1, .15	57.72, 8.39 2.27, 1.08	2.5	.0956387 .1518497	4.361	3.106
4	10, 10	.1, .14	79.73, 5.68 2.32, 1.62	2.5	.0879738 .1551497	15.15	13.588
4	10, 10	.1, .13	64.07, 3.50 2.65, 1.09	3.0	.1244523 .1923786*	-	-
4	10, 10	.1, .12	74.64, 3.58 1.58, 1.36	2.5	.1161760 -.1788954*	-	-
4	20, 20	.1, .13	118.46, 7.20 2.32, 1.10	2.5	.1335662 -.2292814*	-	-
4	50, 50	.1, .13	344.14, 8.78 2.02, 1.00	2.5	.1140111 .1348432	4.843	4.427
4	100, 100	.1, .13	780.47, 19.69 3.25, 1.21	3.5	.1049023 .1388154	8.815	6.859
8	10, 10	.1, .13	128.89, 12.59 3.56, 2.46 2.26, 1.73 1.01, 0.77	4.0	.0956865 .1366953	6.695	5.504

Table 3-4. Experimental results illustrating the effects of signal power levels and the number of antenna elements on resolution.

Number of Elements: 4

Number of Signal Sources: 2

Signal Source Variances: 10, 10

Internal Antenna Noise Variance: 1

Number of Samples: 30

Angular Locations of Signal Sources in Radians/Pi	Eigenvalues	Number of Eigenvectors Selected for Polynomial Formation	Estimated Angular Location in Radians/Pi
.1, .12	72.723, 2.489 2.022, 0.907	3	.1203, -.1496*
		2	.1089, .2308*
		1	.1068
.25, -.3	39.529, 31.105 1.983, 1.390	3	.2573, -.3136
		2	.2546, -.2997

Table 3-5. Experimental results illustrating coalescing and the effect of selecting different numbers of eigenvectors for polynomial formation.

at various levels, the number of eigenvectors selected is varied from one to three. The extraneous minimas are marked with an asterisk. The same experiment is repeated with sources located at $.25\pi$ and $-.3\pi$ and as can be noted, no extraneous minima appeared when three eigenvectors are selected. Though extraneous minima can be eliminated theoretically by considering auxiliary polynomials, it is not clear whether such is the case when the data is contaminated by error noise.

In order to test the effectiveness of the algorithm when data was obtained from an actual antenna array, magnetic tapes were obtained from NRL through the courtesy of Mr. Fred Staudaher and processed. These tapes contained covariance matrices obtained from a 4-element linear array. The number of sources varied from 0 to 10 and an analog-to-digital conversion scheme with 512 samples was used. To compute the covariance matrices, 1024 samples were used.

The covariance matrices for the cases where the number of sources is from 1 to 3 were processed and the results are shown in Tables 3-6 to 3-9. As can be seen, the results are encouraging for one source where the angular errors are below 4 degrees (Table 3-6). For two sources except for two cases (File Numbers 17 and 18 in Table 3-7), the angular errors are below 5 degrees. For File Numbers 17 and 18, it appears that the algorithm coalesces two sources into one. These cases and File Numbers 14 and 15 were run again selecting different numbers of eigenvectors for polynomial formation (Table 3-8). (The extraneous minima as before are marked with an asterisk). For the three source cases (Table 3-9), the algorithm appears to

Number of Antenna Elements: 4

Antenna Array Interelement Spacing: Half Wavelength

Number of Sources: 1

Number of Samples: 1024

NRL Mag Tape File Number	Source Angular Location, in Degrees	Eigenvalues	Threshold for Selecting Eigenvectors	Estimate of Angular Location Degrees	Absolute Angular Error in Degrees
4	0	7.06352, 0.01363 0.00087, 0.00048	1.0	-1.10	1.10
5	0	7.92217, 0.01718 0.00067, 0.00049	1.0	0.16	0.16
6	-41	9.37944, 0.02561 0.00206, 0.00090	1.0	-44.58	3.58
7	-45	12.25328, 0.02759 0.00245, 0.00108	1.0	-47.49	2.49
8	-32	5.47412, 0.01151 0.00119, 0.00058	1.0	-34.06	2.06
9	16	4.48534, 0.00839 0.00144, 0.00083	1.0	19.90	3.90
10	30	7.36320, 0.01029 0.00222, 0.00099	1.0	29.87	1.13
11	30	6.61269, 0.00824 0.00204, 0.00093	1.0	31.93	1.93
12	43	4.34818, 0.00820 0.00144, 0.00069	1.0	40.62	2.38
13	51	4.98364, 0.00961 0.00138, 0.00077	1.0	47.45	3.55

Table 3-6. Results of processing NRL magnetic tapes with one source.

Number of Antenna Elements: 4

Antenna Array Interelement Spacing: Half Wavelength

Number of Sources: 2

Number of Samples: 1024

NRL Mag Tape File Number	Source Angular Location in Degrees	Eigenvalues	Threshold for Selecting Eigenvectors	Estimate of Angular Location Degrees	Absolute Angular Error in Degrees
14	16, -32	6.12452, 3.95238 0.00780, 0.00181	0.5	19.99, -33.73	3.99
15	51, 16	4.17308, 2.84038 0.00773, 0.00411	0.5	49.58, 20.34	4.34
16	16, -45	9.00405, 3.18835 0.02218, 0.00361	0.5	17.96, -49.64	4.64
17	30, 51	6.17605, 0.87517 0.00850, 0.00269	0.5	31.52, -18.96*	-
18	-45, -32	15.8755, 1.50405 0.00534, 0.00326	0.5	23.14*, -33.85	-
19	-45, -51	11.1644, 3.91715 0.02223, 0.00351	0.5	-45.41, 53.03	2.03

Table 3-7. Results of processing NRL magnetic tapes with two sources.

NRL Mag Tape File Number	Threshold for Selecting Eigenvectors	Number of Eigenvectors Above Threshold	Estimates of Angular Locations in Degrees
17	1.0	1	39.79
17	.003	3	29.59, 53.94, -26.74*
18	2.0	1	-41.56
18	.004	3	38.28*, -32.47
14	.002	3	19.82, -30.99
15	.005	3	19.60, 59.16, -38.52*

Table 3-8. Results of processing NRL magnetic tapes with two sources and selecting different numbers of eigenvectors.

Number of Antenna Elements: 4

Antenna Array Interelement Spacing: Half Wavelength

Number of Sources: 3

Number of Samples: 1024

NRL Mag Tape File Number	Source Angular Locations In Degrees	Eigenvalues	Threshold for Selecting Eigenvectors	Number of Eigenvectors Above Threshold	Estimates of Angular Locations
20	30, 43, 51	7.16162	.01	3	28.81, 50.48, -26.66*
		0.83685	.5	2	30.81, -19.52*
		0.01087	1.0	1	39.99
		0.00456			
21	-41, -45, -32	15.38885	.01	3	-25.44
		1.01585	1.0	2	23.86*, -33.36
		0.01678	10.0	1	-42.06
		0.00538			
22	-41, 30, 51	7.25728	.01	1	-43.39, 27.00, 67.35
		4.07460			
		0.61344			
		0.00508			
23	30, 43, 51	5.98311	.005	3	30.04, 55.03, -26.85*
		0.56175	.05	2	31.92, -18.82*
		0.00812	1.0	1	40.62
		0.00282			
24	16, 30, 30	6.24163	.005	3	23.33, -23.01*
		0.79472	.5	2	22.32, -30.34*
		0.00802	5.0	1	27.17
		0.00443			

Table 3-9. Results of processing NRL magnetic tapes with three sources.

perform less satisfactorily. These cases were run under the assumption of one, two and three sources and as Table 3-9 shows, in only one case (File Number 22) were the three sources resolved satisfactorily.

The main source of computational errors is in the A/D converter which uses only 512 levels. Hence the data is accurate only up to the second decimal digit and this seems to introduce significant errors in estimates. Also, the number of antenna elements (4) is quite small to resolve sources close to each other satisfactorily. It is hoped that we may obtain in the near future data from an antenna receiver system with more elements and precision.

4. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

This report presented algorithms for locating simultaneously multiple signal sources by means of linear arrays and digital signal processing. Computational complexity analysis of these algorithms shows that it is possible to implement them in real time on high speed parallel processing computer systems within the next decade. Simulations of the algorithms yield satisfactory results after a small number of input samples (30-50) and preliminary results indicate that one can be optimistic in expecting high resolution and good accuracy when a relatively small number (16-32) of antenna elements is used.

At this time it is appropriate to compare the proposed algorithms with that of Berni (BERN 75). For Berni's algorithm, one is forced to restrict the number of elements to m+1 where m is the number of sources to be located. As indicated in Section 3, better accuracies and resolution can be expected by having a large number of antenna elements (refer to Tables 3-3 and 3-4) and thus Berni's algorithm may not be preferable in this sense. Also it may be clear from the description of his algorithm that it is not applicable when the sources are correlated and the signal source covariance matrix is singular.

For future research, it is suggested that a thorough experimental investigation be conducted to study the behavior of the proposed algorithms in terms of accuracy, resolution and convergence at various signal source power levels and angular locations and at various numbers of signal sources. The future research should concentrate on evaluating the effects of errors (due to A/D conversion as well as the non-whiteness of antenna internal noise) on computational accuracy and methods of minimizing these effects. The intriguing problem of extraneous minima (mentioned in Section 3) should also be examined to see whether they can be completely eliminated. The effectiveness of the proposed algorithms in locating signal sources that have a singular covariance matrix is an area that is yet to be explored and the future research may focus on the methods of using the proposed algorithms to combat the tracking errors caused by multipath propagation. It would also be of interest to investigate whether circular and conformal arrays may be used instead of linear arrays for multiple signal location. Finally the problem of recovering multiple signals with different angles of arrival should be of major interest in the fields of communication and defense and hence be pursued in the future.

REFERENCES

BERN 75

Berni, A.J., "Target Identification by Natural Resonance Estimation", IEEE Trans. Aerospace and Electronic Systems, Vol. AES-11, No. 2, March 1975, p. 147-154.

DAHL 74

Dahlquist, G., et al., Numerical Methods, Prentice-Hall, NY, 1974.

GANT 60

Gantmacher, F.R., The Theory of Matrices, Vol. 1, Chelsea, NY, 1960.

NERI 67

Nering, E.D., Linear Algebra and Matrix Theory, Wiley, NY, 1967.

WILK 65

Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Glossary of Symbols

No. of Antennas: n

No. of Signal Sources: m

Signal Sources/Jammers: $J_1, J_2, \ldots, J_m$

Angular Locations of Sources: $\theta_1, \theta_2, \ldots, \theta_m$

Wavelength of the Carrier Signal: ν

Direction Vectors: $\xi_1, \xi_2, \ldots, \xi_m$

$$\xi_i = \begin{bmatrix} 1 \\ \exp\{ j(2\pi d/\nu) \sin \theta_i \} \\ \exp\{ j(4\pi d/\nu) \sin \theta_i \} \\ \vdots \\ \exp\{ j(n-1)(2\pi d/\nu) \sin \theta_i \} \end{bmatrix}$$

$$\Xi = [\xi_1 | \xi_2 | \ldots | \xi_m] \qquad \Lambda = \Xi^\dagger \Xi$$

$$\Pi = \Lambda^{-1}$$

Complex Envelope Amplitudes of Signal Sources:

$$s_1, s_2, \ldots, s_m$$

Source Signal Covariance Matrix:

$$A = ||a_{ij}|| = ||E(s_i^* s_j)||$$

Received Signal Covariance Matrix:

$$B = ||b_{ij}|| = \sum_{i,j} a_{ij} \xi_i \xi_j^\dagger$$

Antenna Internal Noise Covariance Matrix:

$$g^2 I$$

Combined Signal Covariance Matrix:

$$C = ||c_{ij}|| = g^2 I + B$$

Eigenvectors of C: $\zeta_1, \zeta_2, \ldots, \zeta_i, \ldots$

$\rho_1, \rho_2, \ldots, \rho_i, \ldots$

Eigenvalues of B: $\beta_1, \beta_2, \ldots, \beta_i, \ldots$

Eigenvalues of C: $\gamma_1, \gamma_2, \ldots, \gamma_i, \ldots$

APPENDIX 1

A BRIEF DESCRIPTION OF G-471, A PARALLEL PROCESSING COMPUTER

W. W. GAERTNER
RESEARCH INC.

THE G-471

A FOURTH-GENERATION COMPUTER SYSTEM WITH VASTLY SUPERIOR PROCESSING POWER

205 SADDLE HILL ROAD, STAMFORD, CONNECTICUT 06903 • (203) 322 3673

KEY FEATURES OF THE G-471, A NEW PARALLEL/ASSOCIATIVE COMPUTER

1. INTRODUCTION

The G-471 is a 4th generation parallel/associative computer whose architecture and technology produce significant improvements in performance and maintainability over earlier designs, at considerably lower cost. It is described in more detail in Chapter 2.

Its most important advantages are:

Extremely high processing rates (64 - 4096 MIPS)

Extremely high memory bandwidth (512 Mbytes/sec for every 32 Processing Elements, 16,384 Mbytes/sec for 1024 PEs)

Main-stream technology

Much lower cost than conventional scientific computers

Minimal learning time for programming

Microprogrammability

Higher-order languages (FORTRAN, BASIC, COBOL)

Multiprocessing for higher speed and resource utilization

Parallel pipeline capability

Operation as a stand-alone unit or as a peripheral to other computers

Operation with off-the-shelf disks and tapes as mass memory

High reliability through fault tolerance and easy maintainability.

2. DETAILED DESCRIPTION

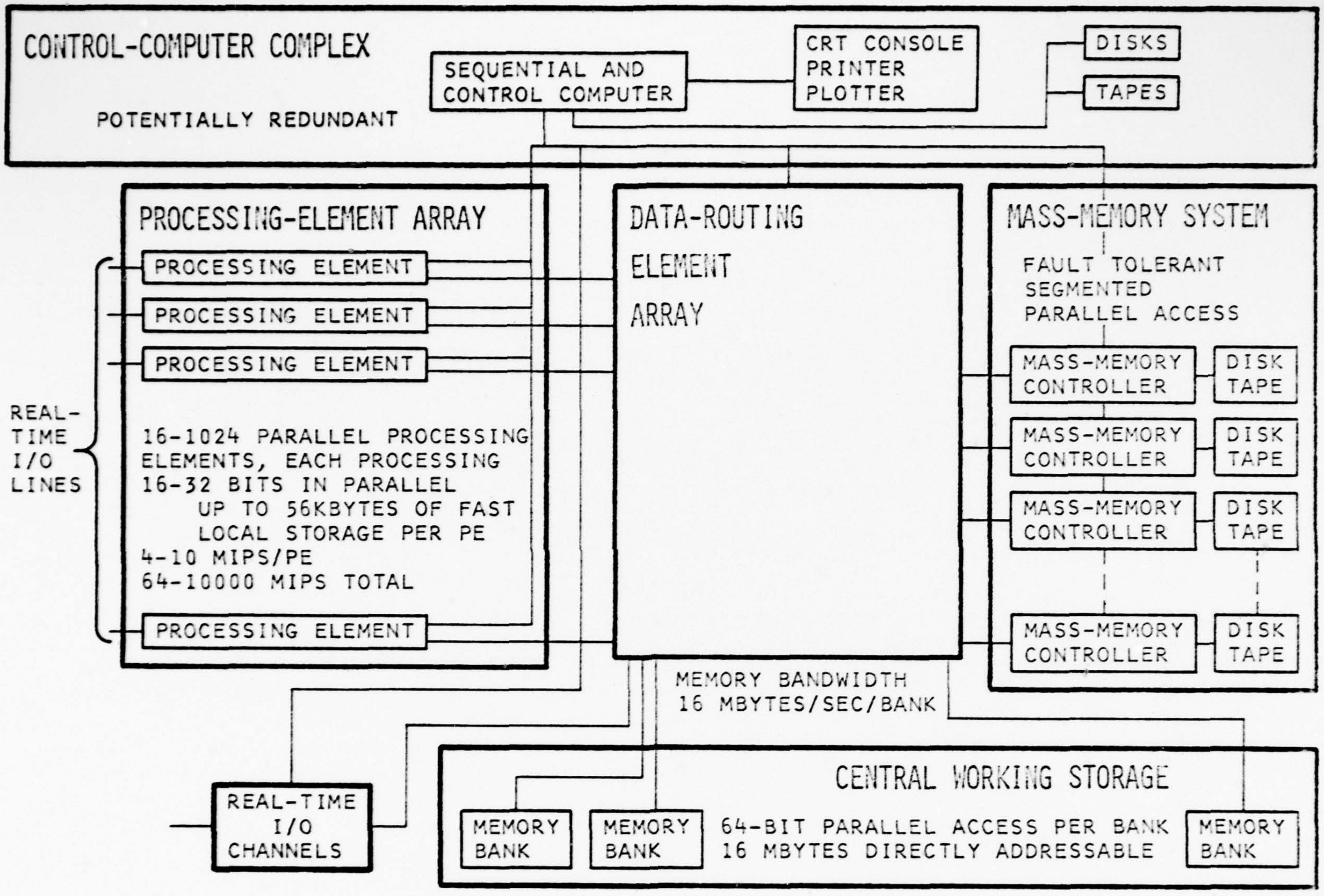
The figure on the following page shows the block diagram of the G-471. The Sequential and Control Computer (SCC) is a standard or militarized minicomputer or a larger machine if desired. It controls the operations of the Processing-Element and Data-Routing Element Arrays (PE&DREA), and it also performs sequential computations. The optimum size of this Sequential and Control Computer depends on the size of the array to be controlled (16 to 1024 Processing Elements); on the number of control actions required (if the processing elements perform mostly routine operations which are programmed in the PE local memories the demand on the control computer is small; if the PE programs require frequent changes, the demand on the control computer is higher); and on how many sequential operations must be performed in addition to the array-control functions. The Sequential and Control Computer can be supplied with the overall system, or it can be a customer-owned computer for which only the necessary interfaces to the PE and Data-Routing Element Arrays and the mass-memory controllers are supplied. The Sequential and Control Computer has standard peripherals such as a CRT terminal and high-speed printer, and it normally runs under a real-time disk-operating system.

The Processing Elements (PEs) in the PE Array contain standard minicomputers, each of which processes 16 bits in parallel. The local memory associated with each PE is expandable to at least 56 kbytes. This storage area can be assigned to data or program, RAM or ROM, in any mix.

The PE designs have been enhanced significantly as compared to standard minicomputers to make the most frequent operations in the PE Array particularly fast. For example, DMA (direct-memory access) to portions of the PE memory is possible while other portions are simultaneously accessed by the CPU. Each microprogrammable high-speed arithmetic enhancement processor carries out 32-bit floating-point multiplications at a rate of 4 MIPS (128 MIPS for 32 PEs).

PEs can be readily paralleled, with arrays ranging from 16 to 1024 PEs (256 to 16,384 bits in parallel).

Each PE can directly address up to 16 Mbytes of semiconductor RAM Central Working Storage (CWS). The CWS is partitioned into at least as many memory banks as there are PEs to permit parallel access. Each PE and each Memory Bank is provided with a 64-bit wide memory transfer channel allowing a transfer rate per channel of 16 Mbytes/sec. Each group of 32 PEs therefore has a memory bandwidth of 512 Mbytes/sec.



Block Diagram of the G-471 Parallel/Associative Computer System

W.W.GAERTNER RESEARCH INC.

Each PE also has access to the mass memory, typically a bank of disks, using controllers which are operated by the Sequential and Control Computer.

In some applications the data to be processed may not be supplied from mass memory but from real-time sensors with a high data rate, such as video sensors, radar, sonar, large distributed sensor arrays etc. For this purpose Real-Time I/O Channels, each with a transfer rate of 16 Mbytes/sec, are provided.

The Data-Routing-Element Array performs the communication function among the PEs, the Real-Time I/O Channels, the CWS Memory Banks and the Mass-Memory Modules. It is basically a programmable cross-point switch whose switch settings are determined dynamically by the memory addresses from the PEs, the Real-Time I/O Channels, the Mass-Memory Controllers, and the Control Computer. The DRE Array operation is transparent to the programs.

The overall operation of the PE and Data-Routing-Element Arrays is controlled by the Control Computer. The control functions are as follows:

a. Activate any single PE or any group of PEs.

b. Transmit instructions to any PE, singly or in a broadcast mode (instructions can also be loaded from CWS and Mass Memory).

c. Transmit data to the PEs (in the majority of applications most of the data would come from the Mass Memory and from the Real-Time I/O Channels).

d. Receive data and "response signals" from the PEs.

e. Control the system reconfiguration to achieve fault tolerance.

f. Allocate tasks and subtasks to available processing elements.

g. Set up protection registers to limit the PEs to permissible portions of the Central Working Storage (CWS).

A striking feature of the design is the large amount of off-the-shelf hardware used, specifically the control computer, the disk controllers, the mass memory (disk, drum, tape), the control console and printer, and the processing elements themselves (except for the high-speed hardware enhancement options).

A major side benefit of this architecture is that it runs under standard real-time disk operating systems and has all higher language capabilities, as well as console and printer software. Remote access via modem could also be standard.

3. ADVANTAGES OVER EARLIER DESIGNS

Extremely High Processing Rates (64 - 4096 MIPS)

The minimum configuration of 16 PEs will process up to 256 Mbytes/sec. The corresponding number for the "jumbo" 1024-PE array is 16384 Mbytes/sec.

Extremely High Memory Bandwidth (512 Mbytes/sec Per 32-PE Array)

The G-471 design recognizes that the high processing power of the PEs must be supported by the ability to move data very rapidly in and out of a large Central Working Storage (CWS). Thus it is possible for each PE to directly address 16 Mbytes of CWS and to transfer data at a rate of 16 Mbytes/sec. Since PEs can operate in parallel, the effective memory bandwidth for a 32-PE Array is 512 Mbytes/sec.

Main Stream Technology

The G-471 is the first parallel/associative processor in main-stream technology using off-the-shelf microprocessor assemblies, minicomputers, controllers and disks. The amount of custom circuitry is minimized. The benefits of millions of dollars of minicomputer R&D in hardware, software, diagnostics and education are incorporated into the G-471.

Low Cost

Because it utilizes many fully designed and debugged system modules which are in mass production, its hardware and software development costs are drastically lower than a machine designed from the gate level up. With the packaging schemes for most of the components worked out, and with spares readily available, the manufacturing and maintenance costs are also greatly reduced.

The cost reduction over other machines with similar capability amounts to a factor between 6 and 10.

Minimal Learning Time For Programming

The G-471 is an associative/parallel processor computer which "you can program on Day One". Rather than requiring its own fundamental machine language, the instruction repertoire of the G-471 is derived from the instruction repertoires of the minicomputers used. Thus, a programmer familiar with minicomputer software can use the G-471 with a minimum of learning time. The execution times of all instructions are known.

Microprogrammability

The processing elements in the array are individually microprogrammable, thus allowing the ready tailoring of the G-471 to a given application.

Higher-Order Languages

Programs written in FORTRAN, BASIC, and COBOL can be run on the G-471.

Multiprocessing for Higher Speed and Resource Utilization

Different processors in the G-471 can carry out different programs simultaneously, a major difference from conventional parallel processors which usually perform only one instruction at a time on all data in parallel.

Fault Tolerance and Easy Maintainability

The special architecture of the G-471 with its high degree of modularity, and the large amount of diagnostics available from the minicomputer systems components, makes the G-471 highly fault tolerant and maintainable, producing a much greater availability than is customary in machines of this complexity.

Easy Expandability

The G-471 is expandable in four directions:

a. Expansion in the number of parallel processing elements (up to 1024 PEs, processing 16K bits in parallel).

b. Expansion of the fast local semiconductor memory associated with each parallel processing element up to 56 kbytes (58 Mbytes in 1024 PE machine).

c. Expansion of Central Working Storage to 16 Mbytes per 32 PEs.

d. Expansion of Mass Memory by adding parallel disks and controllers.

Operation As a Stand-Alone Unit Or As a Peripheral to Other Computers

The G-471 is configured to operate as a stand-alone unit, or the Processing-Element and Data-Routing-Element Array, the Central Working Storage and the Mass Memory can be interfaced with another computer to operate as a peripheral which carries out specific parallel or associative functions. The software is separable in the same way as the hardware. This is of advantage when the associative capability is to be added to an operational computer system with a large amount of other fully debugged software which must not be disturbed.

Operation With Off-The-Shelf Disks As Mass Memory

The G-471 has been configured to operate with standard off-the-shelf disk units as mass memory. This has the advantage that standard disk controllers can be used with their debugged diagnostics and maintenance procedures. It also allows programming of the G-471 on other machines, and the transfer of the programs via removable disk cartridge.

4. TYPICAL APPLICATION AREAS

The advantages of the G-471 come into play in applications which require very high processing rates and large fast memory, and where the processing algorithms allow a high degree of parallelism.

A few typical examples in this area are:

Image (picture) processing, especially restoration and enhancement, including non-standard algorithms.

Radar signal processing.

Sonar signal processing.

Fast Fourier Transforms (FFT) and other array processes.

Real-time beam forming.

Post processing of image, radar, sonar signal-processor output data.

Seismic signal processing.

Correlation of signals from large distributed sensor arrays.

Weather prediction.

EW/ECM problems which involve high data rates.

Mapping algorithms.

Character string (associative) searches of large data bases.

Multicriteria searches of large files.

Missile-tracking and air traffic control problems.

Monte-Carlo analysis.

High-speed simulation.

Partial differential equations.

Matrix manipulations.

High-speed simulation.

The G-471 is not only a data processor but can also be used as a large array controller since the outputs from the individual PEs can individually control a variety of external devices.

5. RELIABILITY/MAINTAINABILITY

The G-471 has been specifically designed for high reliability, maintainability and availability, using a combination of performance monitoring, fault location, self-repair, redundancy and on-line maintenance. The design is taking advantage of many years of pioneering expertise in fault-tolerant digital system development at W. W. Gaertner Research, Inc.

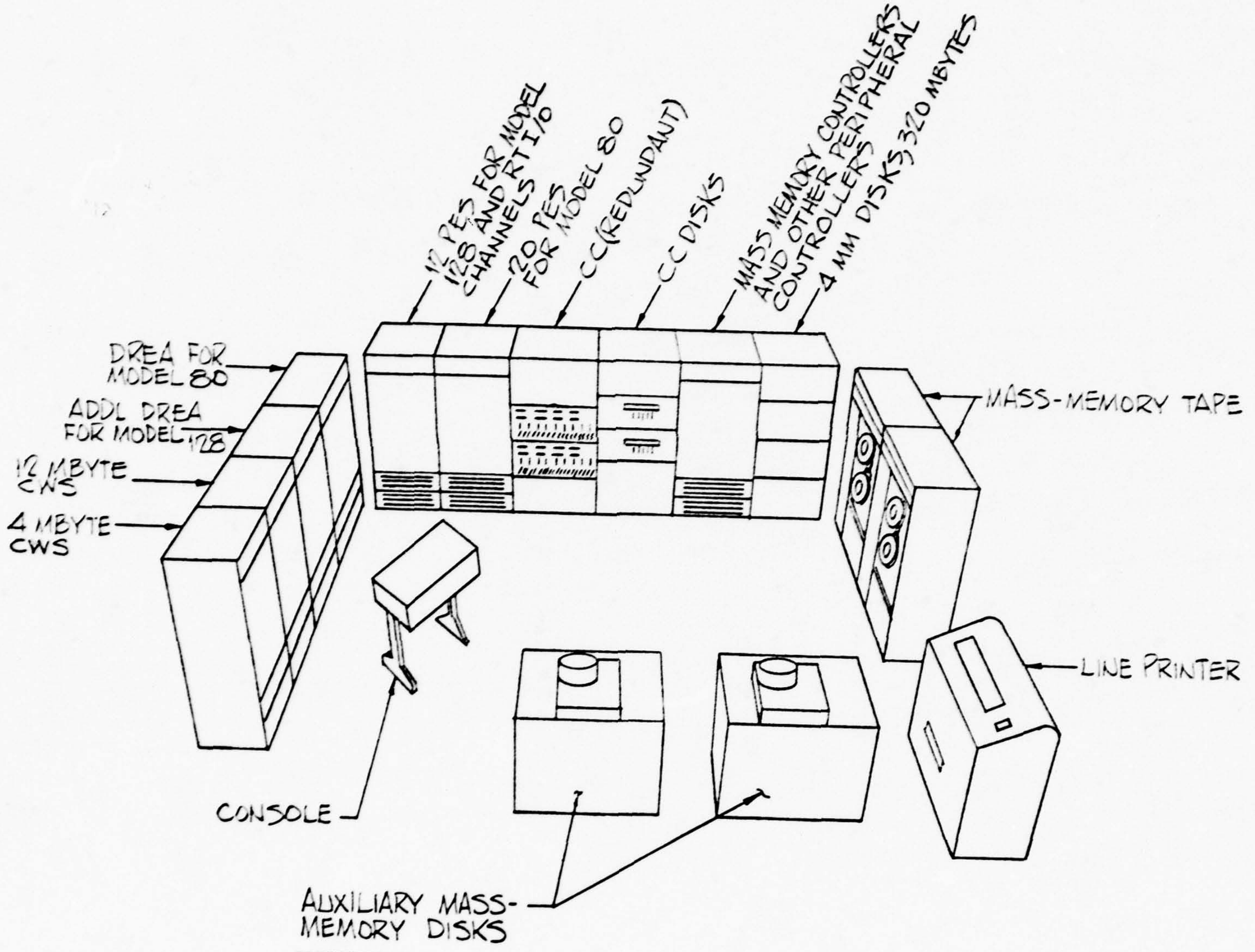
The Control-Computer Complex can be made dual or triple redundant, with voting if necessary. The mass memory (bank of disks) and its controllers are electrically and mechanically segmented allowing for redundant storage of data and programs where desirable. The PEs and their memories as well as the portion of the Data-Routing Network which services the particular PE are mounted on individual boards which can be electrically bypassed and mechanically removed. The Central Working Storage is partitioned into separate memory banks and it can be electronically reconfigured to neutralize hardware faults.

Performance monitoring and fault location software are provided, and on-line maintenance is possible (i.e. replacement of a faulty module can take place with the rest of the system operating).

6. SIZING OF A SPECIFIC G-471 SYSTEM

A G-471 system is sized by analyzing the algorithms which are used in the particular application area (e.g. image processing) and by determining the required maximum execution rate and therefore the number of processing elements, the amount of high-speed local storage, semiconductor Central Working Storage and Mass Memory (disk, tape), real-time I/O facilities, as well as the desired peripherals for the control computer.

W. W. Gaertner Research, Inc. is available on a consulting basis to analyze customer applications and to recommend an optimum configuration for a G-471.



G-471 System Configuration

APPENDIX 2

MATRIX MULTIPLICATION ON A PARALLEL PROCESSING COMPUTER SYSTEM

1. MATRIX MULTIPLICATIONS

Matrix multiplications possess a large degree of parallelism and lend themselves to parallel processing. Consider the multiplication of two square matrices A and B of order nxn. Partition the matrices into k submatrices:

$$\left[\begin{array}{c} A_1 \\ \hline A_2 \\ \hline \vdots \\ \hline A_k \end{array}\right] \left[\begin{array}{c|c|c|c} B_1 & B_2 & -- & B_k \end{array}\right] = \left[\begin{array}{c|c|c|c} A_1B_1 & A_1B_2 & -- & A_1B_k \\ \hline A_2B_1 & A_2B_2 & -- & A_2B_k \\ \hline \vdots & \vdots & \cdots & \vdots \\ \hline A_kB_1 & A_kB_2 & -- & A_kB_k \end{array}\right]$$

It can be seen that k^2 independent matrix multiplications have to be performed and each matrix multiplication requires n^3/k^2 multiplications and additions.

Assuming there is no conflict in accessing A_i and B_j, the time required to perform the matrix multiplication on p processors is

$$(n^3/k^2) \lceil k^2/p \rceil \simeq n^3/p \text{ mtu}$$

where mtu (multiplication time unit) is the time required to perform a single real multiplication. (We estimate the computational complexities in terms of mtu's).

In matrix multiplication the main problem lies in storage organization of matrix elements to ensure conflict-free access rather than the exploitation of computational parallelism. In the following we consider multiplication of 1000x1000 matrices and 10,000x10,000 matrices to illustrate the organizational problems one encounters. To simplify the presentation of algorithms we assume the following computer configuration:

CWS: 10 banks $BK_1, BK_2, \ldots, BK_{10}$ of 256 Kbytes each. (2 Mbytes of Central Storage).

PEs: 10 PEs each with a local storage of 48 Kbytes in three memory banks.

Disk Units: 1 - 2 units with a transfer rate of 10 Mbytes/second.

<u>1000 x 1000 Matrix Multiplication</u>

Partition the matrices as follows:

$$\begin{array}{cc} & 1000 \\ 2 & \left[\begin{array}{c} A_1 \\ \hline A_2 \\ \hline \\ \\ \hline A_{500} \end{array}\right] \\ 2 & \\ & \\ 2 & \end{array} \quad \begin{array}{cccc} 100 & 100 & & 100 \\ \left[B_1 \right. & \left| B_2 \right| & --- & \left| B_{10} \right] \end{array} \; 1000 = \left[\begin{array}{c|c|c|c} A_1B_1 & A_1B_2 & -- & A_1B_{10} \\ \hline A_2B_1 & A_2B_2 & -- & A_2B_{10} \\ \hline \vdots & \vdots & \cdot\cdot & \vdots \\ \hline A_{10}B_1 & A_{10}B_2 & -- & A_{10}B_{10} \end{array}\right]$$

1. Load B_1, B_2, .., B_{10} into the ten banks of CWS.

2. Set i = 0.

3. Load A_{i+1}, A_{i+2}, ..., A_{i+10} into the local storage (PELS) of the ten processors.

4. Perform 10 computational steps described in the following. (The procedure is described for i = 0).

	P_1	P_2	P_3		P_{10}
step 1	A_1B_1	A_2B_2	A_3B_3	----	$A_{10}B_{10}$
step 2	A_1B_{10}	A_2B_1	A_3B_2	----	$A_{10}B_9$
⋮	⋮	⋮	⋮	----	⋮
step 10	A_1B_2	A_2B_3	A_3B_4	----	$A_{10}B_1$

In step 1, P_i performs A_iB_i and stores the result in the bank containing B_i; in step 2, P_i performs $A_iB_{i-1 \pmod{11}}$ and stores the result in the bank containing $B_{i-1 \pmod{11}}$ and so on.

5. Transfer the partial products accumulated in CWS to the disk unit(s).

6. If i = 490, stop. Otherwise $i \leftarrow i + 10$ and go to step 2.

Estimation of Time Requirements

t_M: Time in nanoseconds to perform a single real multiplication ($\sim$ 250).

t_C: Time in nanoseconds to store (fetch) in (from) CWS without conflicts ($\sim$ 500).

r_t: Disk transfer rate in Mbytes/second ($\sim$ 10).

It is assumed $2t_M \geq t_C$ and that each matrix element is 8 bytes long.

To perform $A_i B_j$, we fetch (1000x100) matrix elements and store (2x100) elements. We execute (2x1000x100) multiplications. Since $2t_M \geq t_c$, the process is computational bound.

Time $T^{(1000)}_{MATMUL}$ required for the entire multiplication can be given as

$$T^{(1000)}_{MATMUL} = (t_M/10) + (24/r_t) \text{ seconds.}$$

Hence for typical $t_M (\sim 250)$ and $r_t (\sim 10)$ we have

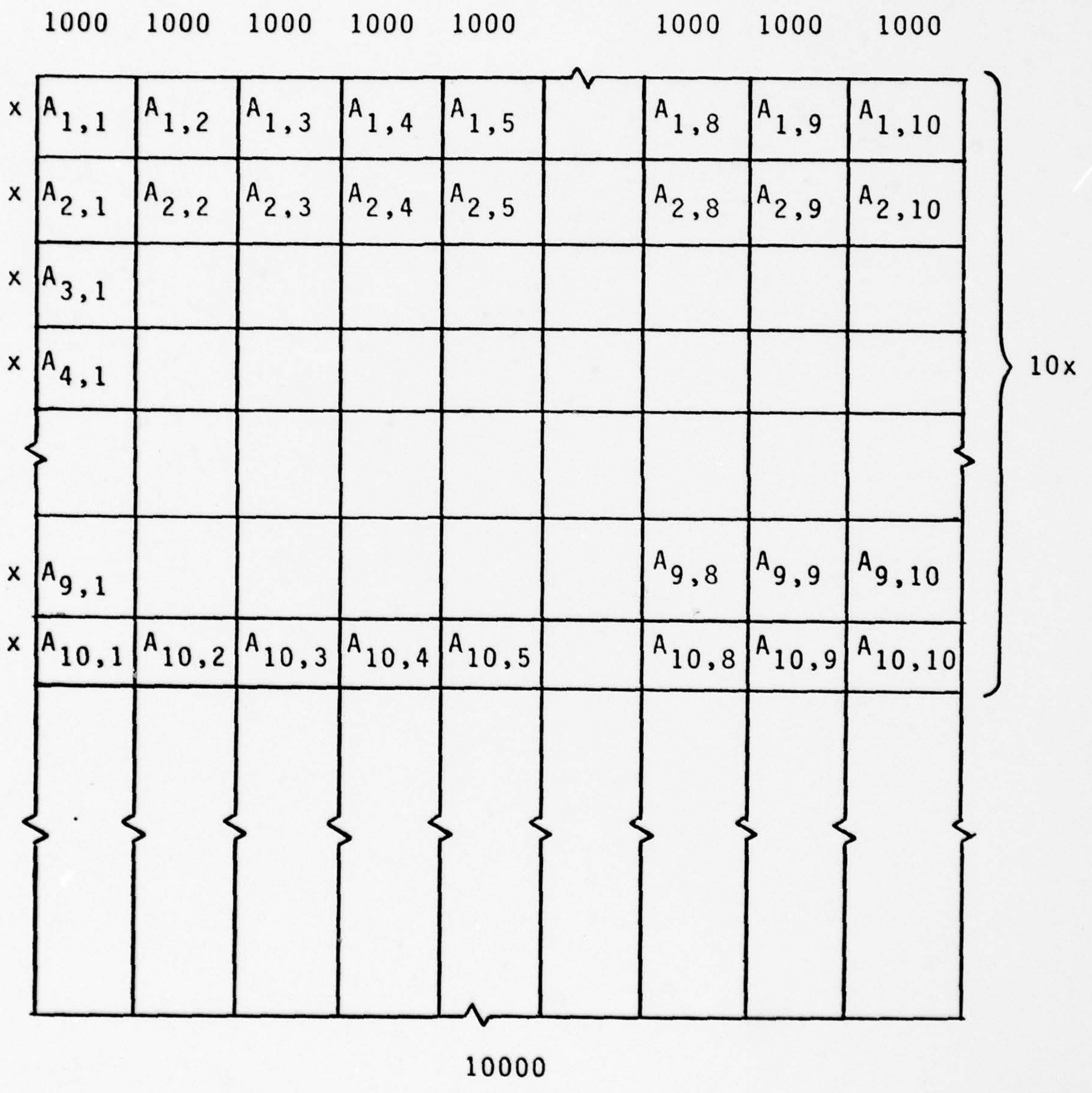
$$T^{(1000)}_{MATMUL} \sim 30 \text{ seconds.}$$

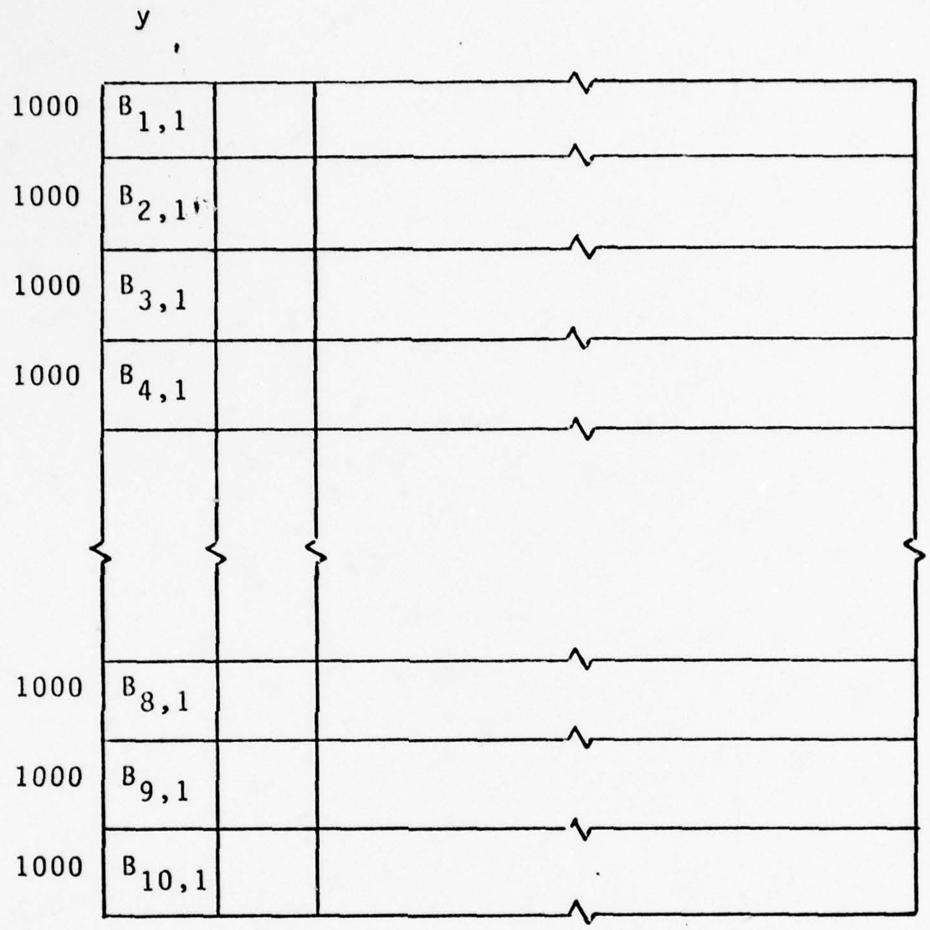
In general if we have p processors and equal number of banks in CWS,

$$T^{(1000)}_{MATMUL} = (t_M/p) + (24/r_t) \text{ seconds.}$$

10,000 x 10,000 Matrix Multiplication

Partition the matrices as follows:


1000 1000 1000 1000 1000 1000 1000 1000
x $A_{1,1}$ $A_{1,2}$ $A_{1,3}$ $A_{1,4}$ $A_{1,5}$ $A_{1,8}$ $A_{1,9}$ $A_{1,10}$
x $A_{2,1}$ $A_{2,2}$ $A_{2,3}$ $A_{2,4}$ $A_{2,5}$ $A_{2,8}$ $A_{2,9}$ $A_{2,10}$
x $A_{3,1}$
x $A_{4,1}$
x $A_{9,1}$ $A_{9,8}$ $A_{9,9}$ $A_{9,10}$
x $A_{10,1}$ $A_{10,2}$ $A_{10,3}$ $A_{10,4}$ $A_{10,5}$ $A_{10,8}$ $A_{10,9}$ $A_{10,10}$
10x
10000



The values for x and y will be determined based on disk transfer rates. The algorithm for matrix multiplication can be described as follows:

1. Transfer B_{i1} to bank BK_i of CWS.

2. Transfer A_{ii} to BK_i of CWS.

3. Perform steps a) and b) in parallel.

   a) Assign processor PE_i to BK_i and compute $A_{ii}B_{i1}$ (of dimensions x X y) using the outer product expansion:

$$X = \begin{array}{c} \\ x \end{array}\left[\begin{array}{c|c|c|c} 1 & 1 & & 1 \\ X_1 & X_2 & & X_{1000} \end{array}\right]$$

$$Y = \begin{array}{c|c} & 1 \\ y & Y_1 \\ \hline y & Y_2 \\ \hline & \vdots \\ \hline y & Y_{1000} \end{array} \qquad XY = X_1Y_1 + X_2Y_2 + \cdot + X_{1000}\,Y_{1000}$$

   b) Transfer $A_{i,\ i+1\ (\mathrm{mod}\ 10)}$ from the disk to $BK_{i+1\ (\mathrm{mod}\ 10)}$.

Figure 1-1 shows the contents and assignment of the banks.

4. Perform steps a) and b) in parallel.

   a) Assign P_i to BKi+1 (mod 10) to compute $A_{i,\ i+1\ (\mathrm{mod}\ 10)} \cdot B_{i+1\ (\mathrm{mod}\ 10),\ 1}$ using outer product expansion.

   b) Transfer $A_{i,\ i+2\ (\mathrm{mod}\ 10)}$ from the disk to $BK_{i+2\ (\mathrm{mod}\ 10)}$.

Figure 1-2 shows the contents and assignment of the banks.

•
•
•
•
•
•

W. W. GAERTNER RESEARCH INC.

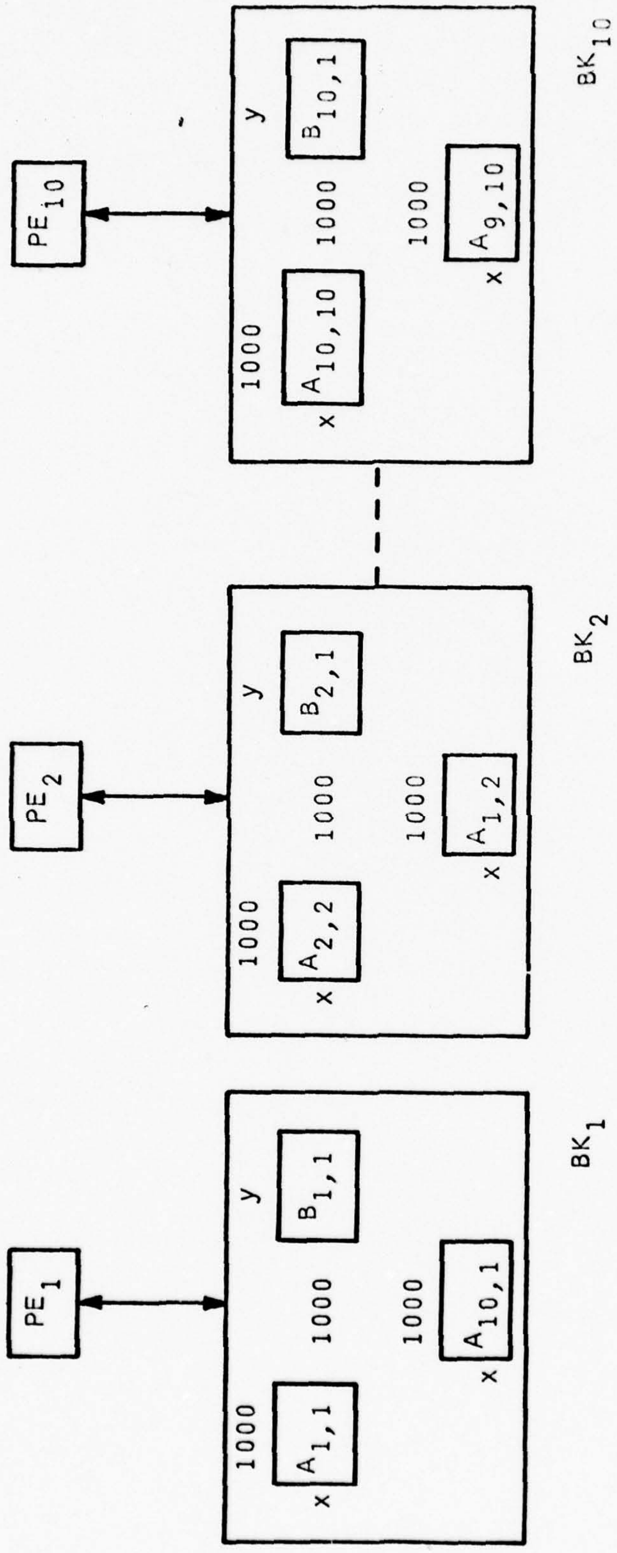


Figure 1-1. Processor-Bank Assignments and Bank Contents at Step 3.

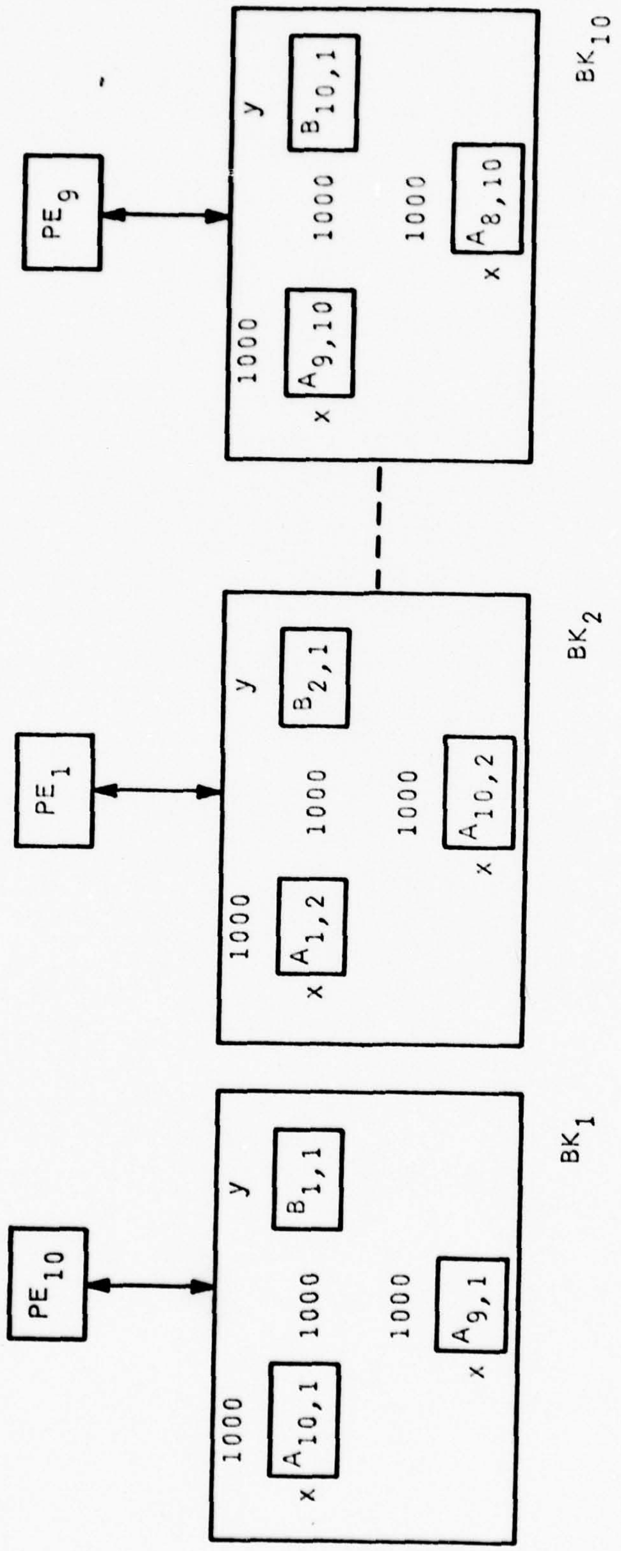
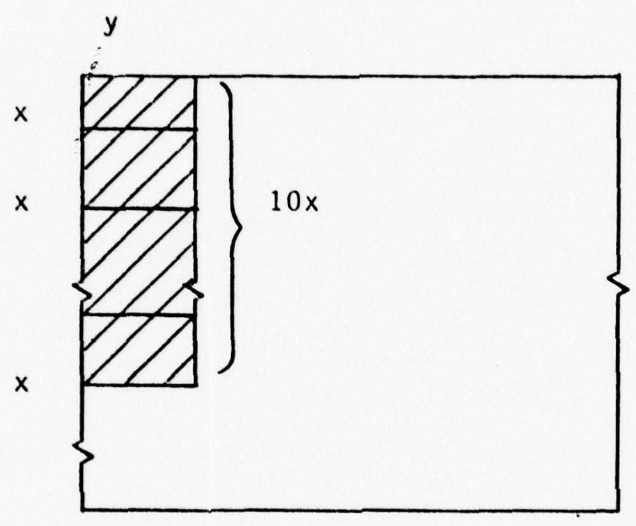


Figure 1-2. Processor-Bank Assignments and Bank Contents at Step 4.

After the processors compute the x X y products (shown hatched) proceed to the next 10x rows of A and repeat the process.



AB

•
•
•
•

After exhausting the rows, repeat the process by selecting next y columns until B is completely processed.

<u>Determination of x and y and Time Estimates</u>

x and y are determined as follows:

$$\left.\begin{array}{l}\text{Number of multiplications}\\ \text{performed by a processor}\\ \text{in each stage}\end{array}\right\} : \quad 1000\,xy$$

$$\left.\begin{array}{l}\text{Number of memory accesses}\\ \text{necessary to compute the product}\end{array}\right\} : \; 1000\,(x + y)$$

$$\left.\begin{array}{l}\text{Number of fetches from the}\\ \text{disc to buffer the elements of A}\end{array}\right\} : \; 1000\,px$$

For complete overlap of disc transfers:

$$(1000\,xy)\ t_M \geq 1000\,(x+y)\ t_c + [(1000\,px)\ t_D/D].$$

Here D is the number of disks and t_D is the time in nanoseconds to transfer a byte. To accommodate three submatrices in each bank:

$$8(2x+y) \geq M_B$$

where M_B is the capacity of each bank in Kbytes (assumed to be 256).

For $t_M = 250$, $t_C = 500$, $t_D = 800$ and 10 processors, complete overlap of disk transfers can be achieved for D = 3. For this case x and y can be given as:

x	y
5	21, 22
6	19, 20
7	18

and time for matrix multiplication:

$$t_{MATMUL}^{(10,000)} \simeq 10^3\ t_M/p \text{ seconds}$$

$$\simeq 7 \text{ hours.}$$

For $p = 20$ and $D = 6$,

$$t_{MATMUL}^{(10,000)} \simeq 3.5 \text{ hours.}$$

For $D/p < \cdot 3$ (p: no. of processors)

$$t_{MATMUL}^{(10,000)} \simeq \text{MAX} \left\{10^3 t_M/p,\ 10^3(t_C+t_D/D)/y\right\}$$

For $x = 6$, $y = 20$ and $D = 1$:

$$t_{MATMUL}^{(10,000)} \simeq 16 \text{ hours.}$$

APPENDIX 3

MATRIX EIGENVALUE/EIGENVECTOR COMPUTATION ON A PARALLEL PROCESSING COMPUTER

3. MATRIX EIGENVALUE/EIGENVECTOR COMPUTATION

We consider real symmetric matrices and analyze the speed-ups achievable in eigenvalue/eigenvector computations by means of parallel processing. (Our analysis is applicable to Hermitian matrices since they can be converted to real symmetric matrices [1].) There are two main steps involved in finding the eigenvalues/eigenvectors of a symmetric matrix:

1) Reduce the matrix to tri-diagonal form by Householder transformations. This requires approximately $2n^3/3$ multiplications for a nxn matrix.

2) Apply Francis-Kublanovskaja QR transformations iteratively to diagonalize the tri-diagonal matrix and obtain eigenvalues. Each iteration takes 3n multiplications and 3n divisions, and experiments show that 2 QR iterations are required per eigenvalue [2].

First, we explain Householder transformations briefly, and show how the computation in Step 1 can be speeded up by a factor of approximately p, the number of processors. In Step 1, n-2 Householder transformations are applied and after r-1 transformations, we have

$$A_{r-1} = P_{r-1}A_{r-2}P_{r-1}$$

$$A_{r-2} = P_{r-2}A_{r-3}P_{r-2}$$

$$\vdots$$

$$A_1 = P_1A_0P_1$$

$$A_{r-1} = \left[\begin{array}{cccc|cccc}
* & * & & & & & & \\
* & * & * & & & & & \\
 & * & * & * & & & & \\
\cline{5-8}
 & & * & * & * & \circledast & \circledast & \circledast \\
\hline
 & & \multicolumn{1}{|c}{} & * & * & * & * & * \\
 & & \multicolumn{1}{|c}{} & \circledast & * & * & * & * \\
 & & \multicolumn{1}{|c}{} & \circledast & * & * & * & * \\
 & & \multicolumn{1}{|c}{} & \circledast & * & * & * & *
\end{array}\right]
\begin{array}{l}
\left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} r \\
\left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} n-r
\end{array}$$

$$= \left[\begin{array}{cc|c}
\multicolumn{2}{c|}{\multirow{2}{*}{C_{r-1}}} & 0 \\
\cline{3-3}
 & & b^T_{r-1} \\
\hline
0 & \multicolumn{1}{|c|}{b_{r-1}} & B_{r-1}
\end{array}\right]$$

P_r is selected to be symmetric and orthogonal and in the product $P_rA_{r-1}P_n$, the circled elements in A_{r-1} are annihilated. Note if

$$P_r = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & Q_r \end{array}\right] \begin{array}{l} r \\ n-r \end{array}$$

then

$$A_r = P_rA_{r-1}P_r = \left[\begin{array}{cc|c}
\multicolumn{2}{c|}{\multirow{2}{*}{C_{r-1}}} & 0 \\
\cline{3-3}
 & & C^T_r \\
\hline
0 & \multicolumn{1}{|c|}{c_r} & Q_rB_{r-1}Q_r
\end{array}\right]$$

where $c_r = Q_rb_{r-1}$.

Householder showed that for any given column

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_r \end{bmatrix}$$

There exists a symmetric, orthogonal matrix $P = I-2ww^T$ such that w is a nx1 vector of unit length (i.e., $w^Tw = 1$) and

$$Pa = \begin{bmatrix} * \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

where the non-zero element is $(a_1^2 + a_2^2 + . . + a_n^2)^{\frac{1}{2}}$. Thus Q_r can be selected to be a Householder transformation matrix. (For an excellent discussion of Householder transformations, refer to Wilkinson [1].)

The major computation involved in each stage is the evaluation of the product $Q_rB_{r-1}Q_r$. Thus we consider:

$$A = (I-2ww^T)B(I-2ww^T)$$

and analyze the algorithmic aspects of evaluating A. Let

$$C = Bw \qquad D = 2C-2w(C^Tw)$$

Then it follows:

$$A = B-wD^T-Dw^T$$

If A and B are mxm matrices, computation of C = Bw and wD^T requires $2m^2$ multiplications. Hence, in the r th stage there are approximately $2(n-r)^2$ multiplications, and Step 1 requires $2n^3/3$ multiplications on a sequential processor. However, in the present computer with p processors, the number of mtus required to perform computation in the rth stage is:

$$2\left\lceil \frac{n-r}{p} \right\rceil (n-r)$$

and hence Step 1 requires:

$$2\sum_{r=1}^{n-2}\left\lceil\frac{n-r}{p}\right\rceil(n-r) \simeq 2n^3/3p+O(n^2) \text{ mtu}$$

In Step 2, QR transformations are applied to diagonalize the tri-diagonal matrix. In s th stage we decompose

$$A_s = Q_sR_s$$

where Q_s is orthogonal and R_s is upper-triangular, and recombine to obtain

$$A_{s+1} = R_sQ_s$$

since $R_s = Q_s^{-1}A_s = Q_s^TA_s$, we have $A_{s+1} = Q_s^TA_sQ_s$. This process is repeated until A_s converges to a diagonal matrix. In each stage, it may be necessary to introduce shifts to accelerate convergence, as in the following fashion:

$$A_s-k_sI = Q_sR_s$$

$$R_sQ_s+k_sI = A_{s+1}$$

Ortega and Kaiser [1] gave a method for simultaneous decomposition and recombination in QR algorithm for tri-diagonal matrices. Let

$$A_s = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & 0 & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & & \cdots & & \\ & 0 & & & \cdots & \\ & & & & \beta_n & \alpha_n \end{bmatrix}$$

$$A_{s+1} = \begin{bmatrix} a_1 & b_2 & & & & \\ b_2 & a_2 & b_3 & & 0 & \\ & b_3 & a_3 & b_4 & & \\ & & & \cdots & & \\ & 0 & & & \cdots & \\ & & & & b_n & a_n \end{bmatrix}$$

Then

$$\gamma_i = \alpha_1 - u_{i-1}$$

$$p_i = \gamma_1^2/c_{i-1} \quad (\text{if } c_{i-1} \neq 0)$$

$$= c_{i-2}/\beta_i^2 \quad (\text{if } c_{i-1} = 0)$$

$$b_1^2 = s_{i-1}(p_i + \beta_{i+1}^2) \quad (i \neq 1)$$

$$s_i = \beta_{i+1}^2/(p_i + \beta_{i+1}^2); \; c_i = p_i/(p_i + \beta_{i+1}^2)$$

$$u_i = s_i(\gamma_i + \alpha_{i+1})$$

$$a_i = \gamma_i + u_i$$

These equations are executed seriatim, for i + 1,2,..., n, starting with $c_0 = 1$ and $u_0 = 0$. Also the value of β_n and

α_{n+1} are both taken as zero.

At present, there is no known method for exploiting parallelism in non-linear recursion equations and hence we do not expect a speed-up of p in this case. However, as Step 1 requires $O(n^3)$ operations as compared to $O(n^2)$ operations in Step 2, the overall speed-up appears to be primarily decided by the speed-up of Step 1. Since a speed-up of p is achievable in the first step, we can conclude that, in general, the speed-up for obtaining eigenvalues is p.

A similar analysis applies for finding eigenvectors. Since matrix multiplications of the form

$$(I-2w_r w_r^T)(I-2w_{r-1} w_{r-1}^T) \cdot \cdot \cdot \cdot \cdot$$

are involved, we expect a speed-up of p in the first step. In the second step, we perform multiplications:

$$Q_s^T \, Q_{s-1}^T \cdot \cdot \cdot B \qquad .$$

Thus, using the techniques presented for matrix multiplication, we can achieve a speed-up of p.

References:

[1] Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

[2] Dahlquist, G., et al, Numerical Methods, Prentice-Hall, New Jersey, 1974.

APPENDIX 4

LISTING OF COMPUTER PROGRAM FOR MULTIPLE SOURCE LOCATION

```
C       MAIN ROUTINE
C       LINK
C       SIGL4M=SIGL4M,SYSLIB/F/C
C       SIG011/O:1/C
C       SIG012/O:1/C
C       SIG013/O:1/C
C       SIG015/O:1
C       SIG016,SIG014/O:1/C
C       SIG017/O:1/C
C       SIG018/O:1
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        LOGICAL*1 TIMEIS(8)
        INTEGER CASNUM,REINIT,UPDATE
        COMMON/ANSWER/ COMPUT(10,10),ICOMP1,ICOMP2
        COMMON/ SAVE / DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
        1,SIGLOC(16),WLIMIT
        COMMON/COMCOE/COEFF(16,2),THRESH,ANTNIS
        COMMON/ JACO /XMU(32,32),EIGEN(32)
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG,MSET,METHOD
        COMMON/NRLTAP/IFILNO,IFILN2
C
        CASNUM = 0
        IRND = 26949
        JRND = -19461
        PI = 3.14159265358979D0
        TWOPI=2.0D0*PI
        ONEOPI=1.0D0/PI
C       TYPE 10
C10     FORMAT(' INPUT FILE FOR POLYNOMIAL COMPUTATIONS ON THE UNIT
C       1 CIRCLE?',/)
C       CALL ASSIGN(3,'COEFF.FIL',9)
C
        TYPE 5000
5000    FORMAT(' TYPE 1 FOR INPUT FROM TERMINAL OR 2 FOR INPUT FROM
        1FILE.')
        ACCEPT 5001,ITTY
5001    FORMAT(I3)
20      TYPE 5005
5005    FORMAT(///,' SELECT ONE OF THE FOLLOWING ALGORITHMS:',//
        1,' TYPE 1 FOR ALGORITHM THAT SELECTS EIGENVECTORS
        2 BASED ON EIGENVALUES.',/
        3,' TYPE 2 FOR ALGORITHM THAT SELECTS EIGENVECTORS
        4 BASED ON INNER PRODUCTS.',/
        5,' TYPE 3 FOR ALGORITHM THAT SELECTS INDEPENDENT VECTORS FROM'
        6,' MULTIPLE',/,8X,'EIGENVECTOR SETS BY GRAM-SCMIDT PROCEDURE.')
        ACCEPT 5001,METHOD
C
        IFLAG=1
C
        IF(IFLAG.NE.1) GOTO 30
         CALL SUBA
30      IF(ITTY.EQ.4) GOTO 9990
```

```
        IF(ITTY.EQ.2) GOTO 40
        CALL SUBB(IFLAG)
        IF(IFLAG.NE.1) GOTO 40
        CALL SIGOPT
        GOTO 20
40      CONTINUE
        CALL SIGO12
C
C
        IF(METHOD.EQ.1) CALL SIGO18
        IF(METHOD.EQ.2) CALL SIGO17
        IF(METHOD.EQ.3) CALL SIGO16
C
C
        CALL SUBCIR(NUMANT-1)
        GOTO 20
9990    CONTINUE
        STOP
        END
        BLOCK DATA
        COMMON/NRLTAP/IFILNO,IFILN2
        DATA IFILNO/-1/
        END
        SUBROUTINE CMPLXM(A,B,C,D,E,F)
        DOUBLE PRECISION A,B,C,D,E,F
        E=A*C-B*D
        F=A*D+B*C
        RETURN
        END
```

```
      SUBROUTINE SUBA
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      REAL FLOAT
      INTEGER REINIT,UPDATE,CASNUM
      COMMON/ANSWER/ COMPUT(10,10),ICOMP1,ICOMP2
      COMMON/ SAVE / DIFVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
     1,SIGLOC(16),WLIMIT
      COMMON/ COMCOE / COEFF(16,2),THRESH,ANTNIS
      COMMON/JACO/XMU(32,32),EIGEN(32)
      COMMON/ CONSNT / PI,TWOPI,ONEOPI
      COMMON/HH/ AIN(16,2),SIG(16,2),SN(16,2)
     1,  XI(16,16,2)
      COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
     1,NUMSIG,MSET,METHOD
      COMMON/NRLTAP/IFILNO,IFILN2
C
      GOTO (1000,1200,1300,9990) ITTY


1000  CONTINUE
      TYPE 3000
3000  FORMAT(/////,1X,'NUMBER OF ANTENNA ELEMENTS?')
      ACCEPT 3001,NUMANT
3001  FORMAT(I7)
C
C     SET FOR MULTIPLE ANALYSIS
C
      IF(METHOD.GT.1)NUMANT=NUMANT+1
      IF(NUMANT.GT.15) GOTO 1000
C
      TYPE 3002
3002  FORMAT(1X,'DELAY FACTOR 2*D/LAMBDA ?')
      ACCEPT 3010,DELFAC
3010  FORMAT(F14.7)
      TYPE 3003
3003  FORMAT(1X,'NUMBER OF SIGNAL SOURCES?')
      ACCEPT 3001,NUMSIG
      TYPE 3004
3004  FORMAT(1X,'SIGNAL SOURCE LOCATIONS THETA/PI?')
      ACCEPT 3005,(SIGLOC(I),I=1,NUMSIG)
3005  FORMAT(5(F14.7))
      TYPE 3006
3006  FORMAT(1X,'SIGNAL SOURCE VARIANCES?')
      ACCEPT 3005,(SIGVAR(I),I=1,NUMSIG)
      TYPE 3007
3007  FORMAT(1X,'UPDATE INTERVAL IN NUMBER OF SAMPLES?')
      ACCEPT 3001,UPDATE
      TYPE 3008
3008  FORMAT(1X,'REINITIALIZATION INTERVAL IN NUMBER OF SAMPLES?')
      ACCEPT 3001,REINIT
      TYPE 3307
3307  FORMAT(' INTERNAL ANTENNA NOISE FACTOR?')
      ACCEPT 3010,ANTNIS
      ITHRSS=2
      IF(METHOD.EQ.2) GOTO 3530
```

```
3505    TYPE 50550
        ACCEPT 50551,ITHRSS
3630    IF(ITHRSS.LT.1. OR. ITHRSS.GT.2) GOTO 3505
        IF(ITHRSS.EQ.1) THRESH=0.0D0
        IF(ITHRSS.EQ.2) TYPE 50510
C3308   FORMAT(' THRESHOLD - ANTENNA INTERNAL NOISE POWER?')
        IF(ITHRSS.EQ.2)ACCEPT 3010,THRESH
        IF(METHOD.LT.3) GOTO 3530
        TYPE 3330
3330    FORMAT(/,' TOLERANCE FOR GRAM-SCHMIDT ORTHOGONALIZATION',
        1' PROCEDURE ?')
        ACCEPT 3010,WLIMIT
3530    CASNUM = CASNUM+1
        DELFAC = DELFAC*PI
        GOTO 1400


C1200   EXPANSION FOR FILE INPUT.
1200    CONTINUE
1300    CONTINUE
C
        NUMANT=4
        DELFAC=1.0D0
        DELFAC=DELFAC*PI
        IF(IFILNO.GT.0) GOTO 50500
13010   TYPE 1301
1301    FORMAT(////,' MAG TAPE FILE NUMBERS FROM ... TO ...?')
        ACCEPT 50001,IFILN1,IFILN2
50001   FORMAT(2I5)
50549   TYPE 50550
50550   FORMAT(' TYPE 1 FOR INTERACTIVE THRESHOLD SELECTION.',/
        1,' TYPE 2 FOR NON-INTERACTIVE THRESHOLD SELECTION.')
        ACCEPT 50551,ITHRSS
50551   FORMAT(I1)
        IF(ITHRSS.LE.0.OR.ITHRSS.GT.2) GOTO 50549
        IF(ITHRSS.EQ.1) THRESH=0.0D0
        IF(ITHRSS.EQ.2) TYPE 50510
50510   FORMAT(' THRESHOLD FOR SELECTING EIGENVECTORS?')
        IF(ITHRSS.EQ.2) ACCEPT 50511,THRESH
50511   FORMAT(D14.7)
        IFILNO=IFILN1-1
50500   IFILNO=IFILNO+1
        IF(IFILNO.GT.IFILN2) GOTO 13010
        TYPE 50003,IFILNO
50003   FORMAT(///,' NRL MAG TAPE FILE NUMBER : ' ,I2)
C
        TYPE 50004,NUMANT,DELFAC/PI
50004   FORMAT(///,' ANTENNA ELEMENTS  : ' ,I1,/,
        1          ' DELFAC 2*D/LAMBDA :' ,1PD8.1,/,
        2          ' NUMBER OF SAMPLES : 1024')
        CALL SIGMTR
        RETURN
C
1400    CONTINUE
        IF(REINIT/UPDATE .LE.10) GOTO 1420
        STOP 'TOO MANY STEPS'
1420    IF(NUMSIG.LE.10) GOTO 1440
        STOP 'TOO MANY SIGNAL SOURCES'
1440    CONTINUE
        WRITE(2,4002) CASNUM
4002    FORMAT(1H1,///,' SIMULATION OF MULTI-SOURCE DIRECTION FINDING
        1 PROBLEM',///,' CASE NUMBER : ',I5,///)
        WRITE(2,4003)NUMANT
4003    FORMAT(' ANTENNA PARAMETERS :',//,'   NUMBER OF ELEMENTS =',
        1I2)
        WRITE(2,4004)DELFAC*ONEOPI
4004    FORMAT('    DELAY FACTOR         =',1PD10.3)
        WRITE(2,4005)
4005    FORMAT('    THE COVARIANCE MATRIX OF INTERNAL ANTENNA
        1 NOISES IS ASSUMED TO BE',/,'    AN IDENTITY MATRIX.')
        WRITE(2,4006),THRESH
4006    FORMAT(///,' ALGORITHM PARAMETER :',//,3X,'THRESHOLD -
        1 ANTENNA INTERNAL NOISE POWER = ',1PD10.3)
C
1500    ILOOP = 0
        ICOMP1=0
        ICOMP2=0
        DO 4 I = 1, NUMSIG
4       SIGLOC(I) = PI*SIGLOC(I)
        DO 5 I = 1, NUMANT
        DO 5 J = 1, NUMANT
        DO 5 K = 1,2
5       XM(I,J,K) = 0.0D0

C       MODULE DIR-VEC-GEN.  GENERATES DIRECTION VECTORS OF SIGNAL
C       SOURCES.
        DO 10 IDIR = 1, NUMSIG
        XX = -DELFAC*DSIN(SIGLOC(IDIR))
        DO 10 JDIR = 1, NUMANT
        XY = DBLE(FLOAT(JDIR-1))*XX
        DIRVEC(IDIR,JDIR,1) = DCOS(XY)
        DIRVEC(IDIR,JDIR,2) = DSIN(XY)
10      CONTINUE

        MFLAG=0
        RETURN
9990    ITTY=4
        RETURN
        END
        SUBROUTINE SUBB(IFLAG)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        REAL RAN,FLOAT
        INTEGER REINIT,UPDATE,CASNUM
        COMMON/COMCOE/ COEFF(16,2),THRESH,ANTNIS
        COMMON/ SAVE / DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
        1,SIGLOC(16)
        COMMON/JACO/XMU(32,32),EIGEN(32)
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        COMMON/HH/ AIN(16,2),SIG(16,2),SN(16,2)
        1,  XI(16,16,2)
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG
C
        IFLAG=0
2000    ILOOP = ILOOP + 1

C       MODULE ANT-INT-NOISE.  GENERATES ANTENNA INTERNAL NOISE SAMPLES.
        DO 20 IANT = 1, NUMANT
        ANT1 = DBLE(RAN(IRND,JRND))
        ANT2 = DBLE(RAN(IRND,JRND))
        CALL GAUS(ANT1,ANT2,AIN(IANT,1),AIN(IANT,2),ANTNIS)
20      CONTINUE

C       MODULE SIG-GEN.  GENERATES RECEIVED SIGNAL SAMPLES.
        DO 30 ISIG = 1, NUMSIG
        SIG1 = DBLE(RAN(IRND,JRND))
        SIG2 = DBLE(RAN(IRND,JRND))
        CALL GAUS(SIG1,SIG2,SIG(ISIG,1),SIG(ISIG,2),0.5D0*
        1SIGVAR(ISIG))
30      CONTINUE
        DO 40 ISIG = 1, NUMANT
        SN(ISIG,1) = 0.0D0
        SN(ISIG,2) = 0.0D0
        DO 40 JSIG = 1, NUMSIG
        CALL CMPLXM(SIG(JSIG,1),SIG(JSIG,2),DIRVEC(JSIG,ISIG,1),
        1DIRVEC(JSIG,ISIG,2),A1,A2)
        CALL CMPLXA(SN(ISIG,1),SN(ISIG,2),A1,A2,SN(ISIG,1),
        1SN(ISIG,2))
40      CONTINUE

C       MODULE COMBINE.  COMBINES ANTENNA INTENAL NOISE AND RECEIVED
C       SIGNALS.
        DO 50 ICOM = 1, NUMANT
        SN(ICOM,1) = AIN(ICOM,1) + SN(ICOM,1)
        SN(ICOM,2) = AIN(ICOM,2) + SN(ICOM,2)
50      CONTINUE

C       MODULE COM-COVAR-MAT.  COMPUTES THE COVARIANCE MATRIX OF THE
C       COMBINED SIGNAL SAMPLES.
        DO 60 ICOM = 1, NUMANT
        DO 60 JCOM = 1, NUMANT
        CALL CMPLXM(SN(ICOM,1),-SN(ICOM,2),SN(JCOM,1),SN(JCOM,2),
        1A1,A2)
        XM(ICOM,JCOM,1) = XM(ICOM,JCOM,1) + A1
        XM(ICOM,JCOM,2) = XM(ICOM,JCOM,2) + A2
60      CONTINUE

C       MODULE UPDATE-COVAR-MAT.  UPDATES COVARIANCE MATRIX OR
C       REINITIALIZES TO ZERO MATRIX.
        IF (MOD(ILOOP,REINIT).NE.0) GOTO 3000
        IFLAG=1
        RETURN
3000    CONTINUE
        IF (MOD(ILOOP,UPDATE).NE.0) GOTO 2000
        DO 70 IUPD = 1, 2*NUMANT
        DO 70 JUPD = 1, 2*NUMANT
70      XMU(IUPD,JUPD) = 0.0D0
        DFLOOP=DBLE(FLOAT(ILOOP))
D       TYPE 75,(((XM(IUPD,JUPD,KUPD)/DFLOOP,KUPD=1,2),JUPD=1,NUMANT),
        1IUPD=1,NUMANT)
75      FORMAT(8(1X,F7.4))
        DO 80 IUPD = 1, NUMANT
        DO 80 JUPD = 1, NUMANT
        XMU(IUPD,JUPD) = XM(IUPD,JUPD,1)/DFLOOP
        XMU(IUPD+NUMANT,JUPD+NUMANT)=XM(IUPD,JUPD,1)/DFLOOP
        XMU(IUPD,JUPD+NUMANT) = -XM(IUPD,JUPD,2)/DFLOOP
80      CONTINUE
        IFLAG=0
        RETURN
        END
        SUBROUTINE CMPLXA(A,B,C,D,E,F)
        DOUBLE PRECISION A,B,C,D,E,F
        E=A+C
        F=B+D
        RETURN
        END
        SUBROUTINE CMPLXD(A,B,C,D,E,F)
        DOUBLE PRECISION A,B,C,D,E,F,G
        G=C*C+D*D
        E=A*C+B*D
        F=B*C-A*D
        E=E/G
        F=F/G
        RETURN
        END
        SUBROUTINE CMPLXT(A,B,C,D)
        DOUBLE PRECISION A,B,C,D
        C=DSQRT(A*A+B*B)
        D=DATAN2(B,A)
        RETURN
        END
        SUBROUTINE GAUS(A1,A2,B1,B2,SIG)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        A=-2.0D0*SIG*DLOG(A1)
        A=DSQRT(A)
        B1 = A*DCOS(TWOPI*A2)
        B2 = A*DSIN(TWOPI*A2)
        RETURN
        END
        SUBROUTINE CMPLXC(A,B,C,D)
        DOUBLE PRECISION A,B,C,D
        C=A*DCOS(B)
        D=A*DSIN(B)
        RETURN
        END
        SUBROUTINE SIG012
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        REAL FLOAT
C
        COMMON/SAVE/DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
        1,SIGLOC(16)
        COMMON/ JACO/XMU(32,32),EIGEN(32)
C
        INTEGER CASNUM,REINIT,UPDATE
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT
        1,ILOOP,NUMSIG,MSET,METHOD
C
C       USE XMU2 TO GEN EIGEN2 VIA JACOBI
C
        CALL JACOBI(2*NUMANT,50,1.0D-10,1.0D-10,1.0D-10)
        IF(METHOD.EQ.1) RETURN
        CALL ASSIGN(1,'SIGP2.TMP',9)
        WRITE(1,10)NUMANT
10      FORMAT(I3)
        WRITE(1,15)((XMU(I,J),J=1,2*NUMANT),I=1,2*NUMANT)
15      FORMAT(1X,4D15.8)
        WRITE(1,15)(EIGEN(I),I=1,2*NUMANT)
        CALL CLOSE(1)
C
C       GEN PASS 1 DATA NOW
C
        DFL=DBLE(FLOAT(ILOOP))
        NMANT1=NUMANT-1
        DO 80 I=1,NMANT1
        DO 80 J=1,NMANT1
        XMU(I,J)=XM(I,J,1)/DFL
        XMU(I+NMANT1,J+NMANT1)=XM(I,J,1)/DFL
        XMU(I,J+NMANT1)=-XM(I,J,2)/DFL
80      CONTINUE
C
        CALL JACOBI(2*NMANT1,50,1.0D-10,1.0D-10,1.0D-10)
        RETURN
        END
        SUBROUTINE JACOBI(N,ITMAX,EPS1,EPS2,EPS3)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        COMMON/JACO/A(32,32),EIGEN(32)
        DIMENSION T(32,32),AIK(32),NO(32)
        LOGICAL FINE
C       TYPE 6666,A
D6666   FORMAT(' JACO',/(5(1PE9.3)))
        DO 4 I = 1, N
        DO 4 J = 1, N
4       T(I,J) = 0.0D0
        NM1=N-1
        SIGMA1=0.0
        OFFDSQ=0.0
        DO 5 I=1,N
        SIGMA1=SIGMA1+A(I,I)**2
        T(I,I)=1.0
        IP1=I+1
```

```
          IF(I.GE.N) GOTO 6
          DO 5 J=IP1,N
5         OFFDSQ=OFFDSQ+A(I,J)**2
6         S=2.0*OFFDSQ+SIGMA1
          DO 26 ITER=1,ITMAX
          DO 20 I=1,NM1
          IP1=I+1
          DO 20 J=IP1,N
          Q=DABS(A(I,I)-A(J,J))
          IF (Q.LE.EPS1) GOTO 9
          IF (DABS(A(I,J)).LE.EPS2) GOTO 20
          P=2.0*A(I,J)*Q/(A(I,I)-A(J,J))
          SPQ=DSQRT(P*P+Q*Q)
          CSA=DSQRT((1.0+Q/SPQ)/2.0)
          SNA=P/(2.0*CSA*SPQ)
          GOTO 10
9         CSA=1.0/DSQRT(2.0D0)
          SNA=CSA
10        CONTINUE
          DO 11 K=1,N
          HOLDKI=T(K,I)
          T(K,I)=HOLDKI*CSA+T(K,J)*SNA
11        T(K,J)=HOLDKI*SNA-T(K,J)*CSA
          DO 16 K=I,N
          IF (K.GT.J) GOTO 15
          AIK(K)=A(I,K)
          A(I,K)=CSA*AIK(K)+SNA*A(K,J)
          IF (K.NE.J) GOTO 14
          A(J,K)=SNA*AIK(K)-CSA*A(J,K)
14        GOTO 16
15        HOLDIK=A(I,K)
          A(I,K)=CSA*HOLDIK+SNA*A(J,K)
          A(J,K)=SNA*HOLDIK-CSA*A(J,K)
16        CONTINUE
          AIK(J)=SNA*AIK(I)-CSA*AIK(J)
          DO 19 K=1,J
          IF (K.LE.I) GOTO 18
          A(K,J)=SNA*AIK(K)-CSA*A(K,J)
          GOTO 19
18        HOLDKI=A(K,I)
          A(K,I)=CSA*HOLDKI+SNA*A(K,J)
          A(K,J)=SNA*HOLDKI-CSA*A(K,J)
19        CONTINUE
20        A(I,J)=0.0
          SIGMA2=0.0
          DO 21 I=1,N
          EIGEN(I)=A(I,I)
21        SIGMA2=SIGMA2+EIGEN(I)**2
          IF (1.0-SIGMA1/SIGMA2.LT.EPS3) GOTO 100
26        SIGMA1=SIGMA2
          TYPE 200
200       FORMAT(1X,'CONVERGENCE DID NOT OCCUR')
          RETURN
C         PROGRAM TO SORT THE EIGENVALUES IN DESCENDING ORDER.
100       DO 500 I = 1, N
```

```
500     NO(I) = I
501     FINE = .TRUE.
        DO 510 J = 2, N
        IF (EIGEN(J-1).GE.EIGEN(J)) GOTO 510
        FINE = .FALSE.
        TMP = EIGEN(J-1)
        EIGEN(J-1) = EIGEN(J)
        EIGEN(J) = TMP
        NTMP = NO(J-1)
        NO(J-1) = NO(J)
        NO(J) = NTMP
510     CONTINUE
        IF (.NOT.FINE) GOTO 501
        TYPE 700,N/2,(EIGEN(I),I = 1,N)
700     FORMAT(//,' EIGENVALUES FOR ',I3,' ANTENNA ELEMENTS :',/
        1,(1X,5F14.7))
        DO 650 I = 1, N
        DO 650 J = 1, N
650     A(J,I) = T(J,NO(I))
        RETURN
        END
```

```
      SUBROUTINE SUBCIR(N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL FLOAT
      INTEGER CASNUM,REINIT,UPDATE
      COMMON/ANSWER/ COMPUT(10,10),ICOMP1,ICOMP2
      COMMON/ COMCOE / COEFF(16,2),THRESH
      COMMON/ CONSNT / PI,TWOPI,ONEOPI
      COMMON/SAVE/ DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
     1,SIGLOC(16),WLIMIT
      COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
     1,NUMSIG,MSET
      REAL A(1024,2)
      DIMENSION RESLT(15),PHIVAL(15)
C
D     TYPE 5500,MSET
D5500 FORMAT(' SUBCIR, MSET=',I3)
      INDX=0
      ICOMP1=ICOMP1+1
      ICOMP2=0
      TPIDNU=TWOPI/1024.
      CALL ASSIGN(3,'COEFF.FIL',9)
      REWIND 3
      READ(3)A
      CALL CLOSE(3)
      X1=DBLE(A(1024,1))
      X2=DBLE(A(1024,2))
      RESSAV=CALFUN(X1,X2,N)
      SLPSAV=1.0D0
      DO 300 LOOP = 1, 1024
      X1 =DBLE (A(LOOP,1))
      X2 =DBLE (A(LOOP,2))
      R1 = COEFF(N+1,1)
      R2 = COEFF(N+1,2)
      DO 10 I = N, 1, -1
      R11 = R1*X1 - R2*X2
      R12 = R1*X2 + R2*X1
      R1 = COEFF(I,1) + R11
      R2 = COEFF(I,2) + R12
10    CONTINUE
      RES= R1*R1 + R2*R2
      SLOPE=RES-RESSAV
      IF(LOOP.EQ.1) SLPFST=SLOPE
      IF(SLOPE.LT.0.0D0) GOTO 290
      IF(SLOPE.EQ.0.0D0) GOTO 200
      IF(SLPSAV.GE.0.0D0) GOTO 290
200   DFL=DBLE(FLOAT(LOOP-1))
      DFLM2=DFL-2.0D0
210   CONTINUE
      X1=TPIDNU*DFLM2
      X2=TPIDNU*DFL
      INDX=INDX+1
      CALL FBONAC(X1,X2,RESLT(INDX))
      X1=DCOS(RESLT(INDX))
      X2=DSIN(RESLT(INDX))
      PHIA=CALFUN(X1,X2,N)
```

```
D       TYPE 6066,PHIA,RESLT(INDX)
D6066   FORMAT(' PHIA=',1PD16.8,' RESLT=',D16.8)
C
        PHIVAL(INDX)=PHIA
        IF(INDX.LE.MSET) GOTO 290
        DO 240 I=1,MSET
        IF(PHIA.GE.PHIVAL(I)) GOTO 240
        PHIVAL(I)=PHIA                         !FOUND A LESSER VALUE
        RESLT(I)=RESLT(INDX)
        GOTO 245
240     CONTINUE
C
C
245     INDX=INDX-1
        GOTO 290
251     CONTINUE
D       TYPE 6029,RESLT(INDX),PHIA
D6029   FORMAT(' A=',1PD16.9,', PHI OF A=',D15.9)
290     SLPSAV=SLOPE
        RESSAV=RES
300     CONTINUE
        IF(SLOPE.GE.0.0) GOTO 310
        IF(SLPFST.LE.0.0) GOTO 310
        DFL=0.0
        DFLM2=-2.0D0
        GOTO 210
310     CONTINUE
C
        TYPE 3390
3390    FORMAT(//,' ESTIMATES OF ANGULAR LOCATIONS :')
        DO 340 I=1,INDX
        A1=RESLT(I)/DELFAC
        IF(A1.GT.1.0D0) A1=A1-2.0D0
        A1=DATAN(A1/DSQRT(1.0D0-A1*A1))*ONEOPI
C       ICOMP2=ICOMP2+1
C       COMPUT(ICOMP2,ICOMP1)=A1
        SQRPHI=DSQRT(PHIVAL(I))
        TYPE 345,I,A1,SQRPHI
345     FORMAT(/,5X,'SOURCE ',I2,' :',/
        1,    5X,'   ESTIMATED ANGULAR LOCATION IN RADIANS/PI :',F11.7,/
        2,       5X,'    RESIDUAL ERROR ON UNIT CIRCLE           :',1PD11.4)
340     CONTINUE
C
        RETURN
        END
        DOUBLE PRECISION FUNCTION CALFUN(X1,X2,N)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON/ COMCOE / COEFF(16,2),THRESH
        R1 = COEFF(N+1,1)
        R2 = COEFF(N+1,2)
        DO 10 I = N, 1, -1
        R11 = R1*X1 - R2*X2
        R12 = R1*X2 + R2*X1
        R1 = COEFF(I,1) + R11
        R2 = COEFF(I,2) + R12
```

```
10      CONTINUE
        CALFUN= R1*R1 + R2*R2
        RETURN
        END
        SUBROUTINE FBONAC(AIN,BIN,RESLT)
C
C       FIBONACCI SEQUENCE
C
C       INPUTS AIN,BIN
C
C       OUTPUT RESLT
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INTEGER CASNUM,REINIT,UPDATE
        COMMON/ COMCOE / COEFF(16,2),THRESH
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG
        DIMENSION T1(60),T2(60),TAU(60),A(60),B(60)
        DATA IRSTFL/0/
        DATA MAX /60/
C
C       COMPUTE FIBONACCI SEQUENCE
C
        IF(IRSTFL.NE.0) GOTO 29
        IRSTFL=1
C
        TAU(1)=1.0D0
        TAU(2)=1.0D0
C
        DO 20 K=3,MAX                   !COMPUTE TAU
20      TAU(K)=TAU(K-1)+TAU(K-2)
C
C
29      A(1)=AIN
        B(1)=BIN
C
        N=NUMANT-1
C       .
        DO 40 K=1,MAX-2
        T1(K+1)=(TAU(MAX-1-K)/TAU(MAX+1-K))*(B(K)-A(K))+A(K)
        T2(K+1)=(TAU(MAX-K)/TAU(MAX+1-K))*(B(K)-A(K))+A(K)
C
        X1=DCOS(T1(K+1))
        X2=DSIN(T1(K+1))
        R1 = COEFF(N+1,1)
        R2 = COEFF(N+1,2)
        DO 10 I = N, 1, -1
        R11 = R1*X1 - R2*X2
        R12 = R1*X2 + R2*X1
        R1 = COEFF(I,1) + R11
        R2 = COEFF(I,2) + R12
10      CONTINUE
        PHI1= R1*R1 + R2*R2
C
        X1=DCOS(T2(K+1))
```

AD-A066 635 GAERTNER (W W) RESEARCH INC STAMFORD CONN F/G 17/2
ANGULAR LOCATION OF MULTIPLE SIGNAL SOURCES - A DIGITAL APPROAC--ETC(U)
NOV 77 N00014-77-C-0342
UNCLASSIFIED NL

2 OF 2
AD
A066635

END
DATE
FILMED
5-79
DDC

```
        X2=DSIN(T2(K+1))
        R1 = COEFF(N+1,1)
        R2 = COEFF(N+1,2)
        DO 11 I = N, 1, -1
        R11 = R1*X1 - R2*X2
        R12 = R1*X2 + R2*X1
        R1 = COEFF(I,1) + R11
        R2 = COEFF(I,2) + R12
11      CONTINUE
        PHI2= R1*R1 + R2*R2
C
        IF(PHI1.GE.PHI2) GOTO 30
        A(K+1)=A(K)
        B(K+1)=T2(K+1)
        GOTO 39
C
30      A(K+1)=T1(K+1)
        B(K+1)=B(K)
C
39      CONTINUE
C       TYPE 300,K,A(K),B(K),T1(K+1),T2(K+1),PHI1,PHI2,A(K+1),B(K+1)
D300    FORMAT(' K=',I3
D       1,' A(K)=',1PD15.9,' B(K)=',D15.9
D       2,' T1(K+1)=',D15.9,' T2(K+1)=',D15.9,/
D       3,' PHI1=',D15.9,' PHI2=',D15.9
D       4,' A(K+1)=',D15.9,' B(K+1)=',D15.9)
C
40      CONTINUE
C
        RESLT1=A(MAX-1)
        RESLT2=B(MAX-1)
        RESLT=(RESLT1+RESLT2)*0.50D0
        RETURN
        END
        SUBROUTINE SIGABT
C
C       ABOVE THRESHOLD EIGEN SELECTION
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        REAL FLOAT
        INTEGER REINIT,UPDATE,CASNUM
        COMMON/COMCOE/ COEFF(16,2),THRESH
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        COMMON/016/ AIN(16,2),SIG(16,2),SN(16,2)
        1, XI(16,16,2)
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG,MSET
C
C       COMPUTE LAMBDA, SUM OF THE SQUARES OF W
C
D       TYPE 5,NUMANT,MSET
D5      FORMAT(' SIGABT-NUMANT=',I3,' MSET=',I3)
D       TYPE 6,(((XI(I,J,K),K=1,2),J=1,MSET),I=1,NUMANT)
D6      FORMAT(' SIGABT-XI=',/,(1X,4D15.8))
        DO 6170 I=1,NUMANT-1
        DO 6170 J=1,MSET
        XI(I,J,2)=-XI(I,J,2)
6170    CONTINUE
        SUMSQW=0.0D0
        I=NUMANT
        DO 6180 J=1,MSET
        SUMSQW=SUMSQW+XI(I,J,1)*XI(I,J,1)+XI(I,J,2)*XI(I,J,2)
6180    CONTINUE
D       TYPE 6181,SUMSQW
D6181   FORMAT(' SUMSQW=',1PD20.12)
        IF(DABS(1.0D0-SUMSQW).LT.1.0D-5) TYPE 6185
6185    FORMAT(/,' WARNING ABSOLUTE VALUE OF 1-SUMSQW IS LESS THAN 1.0D-
        15')
C
C       COMPUTE COEFF
C
        COEFF(NUMANT,1)=1.0D0
        CF=-(1.0D0-SUMSQW)
        COEFF(NUMANT,2)=0.0D0
        MP1=NUMANT
        DO 6199 I=1,NUMANT-1
        COEFF(I,1)=0.0D0
        COEFF(I,2)=0.0D0
        DO 6190 J=1,MSET
        COEFF(I,1)=COEFF(I,1)+XI(MP1,J,1)*XI(I,J,1)
        1                            -XI(MP1,J,2)*XI(I,J,2)
        COEFF(I,2)=COEFF(I,2)+XI(MP1,J,1)*XI(I,J,2)
        1                            +XI(MP1,J,2)*XI(I,J,1)
6190    CONTINUE
        COEFF(I,1)=COEFF(I,1)/CF
        COEFF(I,2)=COEFF(I,2)/CF
6199    CONTINUE
C
D       TYPE 6195,((COEFF(I,J),J=1,2),I=1,NUMANT)
```

```
D6195   FORMAT(' COEFF',/, 2(F20.15))
C
        RETURN
        END
```

```
      SUBROUTINE SIGOPT
C
C     REPORT THE SIMULATION RESULTS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER CASNUM,REINIT,UPDATE
      COMMON/CONSNT/PI,TWOPI,ONEOPI
      COMMON/ANSWER/ COMPUT(10,10),ICOMP1,ICOMP2
      COMMON/SAVE/DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
     1,SIGLOC(16)
      COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
     1,NUMSIG,MSET
      DIMENSION IORD(16,17),IRB(16),ILB(16)
C
D     TYPE 1000,((COMPUT(I,J),I=1,ICOMP2),J=1,ICOMP1)
D1000 FORMAT(5(1PD14.7))
C
      DO 20 I=1,NUMSIG
      SIGLOC(I)=SIGLOC(I)*ONEOPI
20    CONTINUE
      CALL SORTD(SIGLOC,NUMSIG,IORD(1,17),IRB,ILB)
C
      DO 40 I=1,ICOMP1
      CALL SORTD(COMPUT(1,I),NUMSIG,IORD(1,I),IRB,ILB)
40    CONTINUE
C
      DO 50 I=1,NUMSIG
      WRITE(2,2010)I
2010  FORMAT(///,' SIGNAL SOURCE ',I2,' PARAMETERS :',/)
      WRITE(2,2020)SIGLOC(IORD(I,17)),SIGVAR(IORD(I,17))
2020  FORMAT('   ANGULAR LOCATION    =',F10.7,' * PI RADIANS',
     1/,'   VARIANCE (POWER)    =',1PD10.3,//)
      WRITE(2,2030)
2030  FORMAT('   COMPUTED VALUES FOR ANGULAR LOCATION :',//
     1,'   SAMPLES            THETA/PI           ERROR/PI',/
     2,'                      RADIANS            RADIANS',/)
C
      NUMTMS=REINIT/UPDATE
      NSAM=0
      DO 70 J=1,NUMTMS
      NSAM=NSAM+UPDATE
C     ACC=COMPUT(I,IORD(J,I))-SIGLOC(IORD(I,17))
D     TYPE 2074,IORD(J,I),IORD(I,J)
D2074 FORMAT(' IORD(J,I)=',I3,', IORD(I,J)=',I3)
C     ACC=COMPUT(I,IORD(I,J))-SIGLOC(IORD(I,17))
      ACC=COMPUT(IORD(I,J),J)-SIGLOC(IORD(I,17))
      WRITE(2,2070)NSAM,COMPUT(IORD(I,J),J),ACC
2070  FORMAT(2X,I5,10X,F10.7,6X,F10.7)
70    CONTINUE
50    CONTINUE
C
      RETURN
      END
      SUBROUTINE SORTD(X,M,IORD,IRB,ILB)
      DOUBLE PRECISION X
```

```
              DIMENSION X(M),IORD(M),IRB(M),ILB(M)
      D       TYPE 100,X
      D100    FORMAT(' X=',3F10.7)
      C
              IF(M.GT.1) GOTO 1
              IORD(1)=1
              RETURN
      1       CONTINUE
              ILB(1)=0
              IRB(1)=0
              DO 14 I=2,M
              ILB(I)=0
              IRB(I)=0
      2       J=1
      4       IF(X(I).GT.X(J)) GOTO 10
      6       IF(ILB(J).EQ.0) GOTO 8
              J=ILB(J)
              GOTO 4
      8       IRB(I)=-J
              ILB(J)=I
              GOTO 14
      10      IF(IRB(J).LE.0) GOTO 12
              J=IRB(J)
              GOTO 4
      12      IRB(I)=IRB(J)
              IRB(J)=I
      14      CONTINUE
              L=1
      16      J=1
              GOTO 20
      18      J=ILB(J)
      20      IF(ILB(J).GT.0) GOTO 18
      22      IORD(L)=J
              L=L+1
      24      IF(IRB(J)) 28,30,26
      26      J=IRB(J)
              GOTO 20
      28      J=-IRB(J)
              GOTO 22
      30      CONTINUE
      D       TYPE 31,X
      D31     FORMAT(' XS=',3F10.7)
              RETURN
              END
```

```
        SUBROUTINE SIG016
C
C       ABOVE THRESHOLD EIGEN SELECTION
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        INTEGER REINIT,UPDATE,CASNUM
        COMMON/COMCOE/ COEFF(16,2),THRESH
        COMMON/JACO/XMU(32,32),EIGEN(32)
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        COMMON/016/ AIN(16,2),SIG(16,2),SN(16,2)
        1, V(16,16,2)
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG,MSET,METHOD
C
C
        DIMENSION W(16,16,2)
C
C
C       SELECT EIGEN VALUES ABOVE THRESHOLD
C       LET THE NUMBER OF SETS BE MSET
C
3999    IF(THRESH.NE.0.0D0) GOTO 4000
        TYPE 4010
4010    FORMAT(/,' THRESHOLD FOR SELECTING EIGENVECTORS ?',$)
        ACCEPT 4015,THRESH
4015    FORMAT(F10.0)
        GOTO 4020
4000    CONTINUE
        TYPE 61210,THRESH
61210   FORMAT(/,' THRESHOLD FOR SELECTING EIGENVECTORS :',1PD8.1)
4020    CONTINUE
C
C        PASS 1 DATA
C
        NMANT2=NUMANT
        NMANT1=NMANT2-1
        IF(METHOD.EQ.1) NMANT1=NUMANT
C       TYPE 11,NMANT1
C11     FORMAT(' NMANT1=',I4)
C
        MSET1=0
        DO 6100 M=1,2*NMANT1,2
        IF(EIGEN(M).GE.THRESH) MSET1=MSET1+1
6100    CONTINUE
        IF(MSET1.GT.0) GOTO 6120
C
        TYPE 6110,THRESH,(EIGEN(I),I=1,2*NMANT1)
6110    FORMAT(' NO EIGEN VALUES ABOVE THRESH OF ',1PD20.13,
        1,/,' EIGEN VALUES ARE',/,4(D20.13))
        STOP ' THRESH ERROR'
C
6120    CONTINUE
C       TYPE 61211,MSET1
C61211  FORMAT(///,' ESTIMATED NUMBER OF SOURCES :',I3)
C       TYPE 6121,THRESH,MSET1,NMANT1
```

```
C6121   FORMAT(' SIG016- THRESH=',D15.8,' MSET=',I6,', NMANT1=',I5)
        DO 6150 I=1,NMANT1
        K=0
        DO 6150 J=1,2*MSET1,2
        K=K+1
        V(I,K,1)=XMU(I,J)
        V(I,K,2)=XMU(I+NMANT1,J)
6150    CONTINUE
6170    CONTINUE
C       TYPE 6670,(((V(I,J,K),K=1,2),J=1,MSET1),I=1,NMANT1)
C6670   FORMAT(' V=',/,(1X,4D15.8))
        IF(METHOD.EQ.1) GOTO 9000
C
C       READ IN PASS 2 DATA
C
C       TYPE 2
C2      FORMAT(' NOW READ P2 DATA')
        CALL ASSIGN(1,'SIGP2.TMP',9)
        READ(1,10)NMANT2
10      FORMAT(I3)
        READ(1,1)((XMU(I,J),J=1,2*NMANT2),I=1,2*NMANT2)
1       FORMAT(1X,4D15.8)
        READ(1,1)(EIGEN(I),I=1,2*NMANT2)
        CALL CLOSE(1)
C       TYPE 3
C3      FORMAT(' FINISHED READ')
C       TYPE 4,(EIGEN(I),I=1,2*NMANT2)
C4      FORMAT(' EIGEN2',/,4F10.7)
C
C       SELECT PASS TWO DATA GREATER THAN THRESH
C
        MSET2=0
        DO 7100 M=1,2*NMANT2,2
        IF(EIGEN(M).GE.THRESH) MSET2=MSET2+1
7100    CONTINUE
        IF(MSET2.GT.0) GOTO 7120
C
        TYPE 7110,THRESH,(EIGEN(I),I=1,2*NMANT2)
7110    FORMAT(' NO EIGEN VALUES ABOVE THRESH OF ',1PD20.13,
        1,/,' EIGEN VALUES ARE',/,4(D20.13))
        STOP ' THRESH ERROR'
C
7120    CONTINUE
C       TYPE 71211,MSET2
C71211  FORMAT(///,' ESTIMATED NUMBER OF SOURCES :',I3)
C       TYPE 7121,THRESH,MSET2,NMANT2
C7121   FORMAT(' SIG016- THRESH=',D15.8,' MSET2=',I6,', NMANT2=',I5)
        DO 7150 I=1,NMANT2
        K=0
        DO 7150 J=1,2*MSET2,2
        K=K+1
        W(I,K,1)=XMU(I,J)
        W(I,K,2)=XMU(I+NMANT2,J)
7150    CONTINUE
7170    CONTINUE
```

```
C
C
C
        IWCOL=0
        DO 2000 J2=2,2*MSET2+1
        IWS=1
        IF(MOD(J2,2).NE.0) GOTO 500      !RETAIN COLUMN TWICE
        IWCOL=IWCOL+1
        IWS=0
500     CONTINUE
        CALL ORG(W,NMANT1,MSET1,IWCOL,IWS)
2000    CONTINUE
C       TYPE 1020,(((V(I,J,K),K=1,2),J=1,MSET1),I=1,NMANT1)
C1020   FORMAT(' RESULANT V=',/,(1X,4D15.8))
        NUMANT=NMANT1
9000    MSET=MSET1
        TYPE 81211,MSET
81211   FORMAT(///,' ESTIMATED NUMBER OF SOURCES :',I3)
        CALL SIGABT
        RETURN
        END
        SUBROUTINE ORG(W,NMANT,MSETV,IWCOL,IWS)
C
C
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON/SAVE/DIRVEC(16,16,2),XM(16,16,2),SIGVAR(16),DELFAC
        1, SIGLOC(16),WLIMIT
        COMMON/016/A(16,2),B(16,2),SN(16,2),V(16,16,2)
C
        DIMENSION WP(16,2),W(16,16,2)
C
C       TYPE 900,((V(I,1,K),K=1,2),I=1,NMANT)
C900    FORMAT(' ORG-V(1 TO NMANT,1,1 TO 2',/4(1X,1PD15.8))
C       TYPE 910,((W(I,IWCOL,K),K=1,2),I=1+IWS,NMANT+IWS)
C910    FORMAT(' ORG-W',/,4(1X,1PD15.8))
C       XR1=0.0D0
C       XR2=0.0D0
C       DO 920 I=1,NMANT
C       DO 920 J=1,MSETV
C       XR1=XR1+V(I,J,1)
C       XR2=XR2+V(I,J,2)
C920    CONTINUE
C       TYPE 925,XR1,XR2
C925    FORMAT(' ORG-SUM OF V=',2(1X,1PD15.8))
        DO 1000 J=1,MSETV
        A(J,1)=0.0D0
        A(J,2)=0.0D0
        DO 100 I=1,NMANT
        CALL CMPLXM(W(I+IWS,IWCOL,1),W(I+IWS,IWCOL,2),V(I,J,1)
        1,-V(I,J,2),XR1,XR2)
        A(J,1)=A(J,1)+XR1
        A(J,2)=A(J,2)+XR2
100     CONTINUE
1000    CONTINUE
```

```
C
C       SET B = SUM A(J) V(J)
C
        DO  50 I=1,NMANT
        B(I,1)=0.0D0
        B(I,2)=0.0D0
        DO 50 J=1,MSETV
        CALL CMPLXM(A(J,1),A(J,2),V(I,J,1),V(I,J,2),XR1,XR2)
        B(I,1)=B(I,1)+XR1
        B(I,2)=B(I,2)+XR2
50      CONTINUE
C
C       WP=W-SUMOF[ B(J)V(J)]
C
        DO 70 I=1,NMANT
        WP(I,1)=W(I+IWS,IWCOL,1)-B(I,1)
70      WP(I,2)=W(I+IWS,IWCOL,2)-B(I,2)
C       TYPE 305,((WP(I,J),J=1,2),I=1,NMANT)
C305    FORMAT(' WP=',/,(4D12.5))
C
C       INNER PRODUCT
C
C       DO 3000 I=1,MSETV
C       A(I,1)=0.0D0
C       A(I,2)=0.0D0
C       DO 3000 J=1,NMANT
C       CALL CMPLXM(WP(J,1),-WP(J,2),V(J,I,1),V(J,I,2),XR1,XR2)
C       A(I,1)=A(I,1)+XR1
C       A(I,2)=A(I,2)+XR2
C3000   CONTINUE
C       TYPE 3010,((A(I,K),K=1,2),I=1,MSETV)
C3010   FORMAT(' INNER PRODUCT OF WP*V=',/,4(1PD15.8))
C
        WX1=0.0
        DO 400 I=1,NMANT
        WX1=WX1+WP(I,1)*WP(I,1)+WP(I,2)*WP(I,2)
400     CONTINUE
        I=1
        IF(WX1.LT.WLIMIT) I=0
D       TYPE 405,WX1,I
D405    FORMAT(' INNER PRODUCT=',G10.2,', I=',I2)
        IF(I.NE.1) GOTO 2000
C
C       MOVE WP INTO V
C
        MSETV=MSETV+1
        ODSWX1=1.0D0/DSQRT(WX1)
        DO 80 J=1,NMANT
        V(J,MSETV,1)=WP(J,1)*ODSWX1
        V(J,MSETV,2)=WP(J,2)*ODSWX1
80      CONTINUE
C
2000    CONTINUE
        RETURN
        END
```

```
      SUBROUTINE SIG017
C
C     METHOD TWO
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INTEGER CASNUM,REINIT,UPDATE
      COMMON/COMCOE/COEFF(16,2),THRESH,ANTNIS
      COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT
     1,ILOOP,NUMSIG,MSET,METHOD
      COMMON/JACO/XMU(32,32),EIGEN(32)
      DIMENSION XMU2(32,32),EIGEN2(32),A(16,2),A2(16,2)
      DIMENSION V(16,16,2),W(16,2),AS(16),AS2(16)
C
C
C     READ PASS TWO DATA..............W.................
C
      CALL ASSIGN(1,'SIGP2.TMP',9)
      READ(1,10)NMANT2
10    FORMAT(I3)
      READ(1,1)((XMU2(I,J),J=1,2*NMANT2),I=1,2*NMANT2)
1     FORMAT(1X,4D15.8)
      READ(1,1)(EIGEN2(I),I=1,2*NMANT2)
      CALL CLOSE(1)
)(
      NMANT1=NMANT2-1
      DO 6150 I=1,NMANT1
      K=0
      DO 6150 J=1,2*NMANT1,2
      K=K+1
      V(I,K,1)=XMU(I,J)
      V(I,K,2)=XMU(I+NMANT1,J)
6150  CONTINUE
C
      DO 7150 J=1,NMANT2
      W(J,1)=XMU2(J,1)
      W(J,2)=XMU2(J+NMANT2,1)
7150  CONTINUE
)(
C     INNER PRODUCT
C
      DO 100 J=1,NMANT1
      A(J,1)=0.0D0
      A(J,2)=0.0D0
      A2(J,1)=0.0D0
      A2(J,2)=0.0D0
      DO 100 I=1,NMANT1
      CALL CMPLXM(W(I,1),-W(I,2),V(I,J,1),V(I,J,2),XR1,XR2)
      A(J,1)=A(J,1)+XR1
      A(J,2)=A(J,2)+XR2
      CALL CMPLXM(W(I+1,1),-W(I+1,2),V(I,J,1),V(I,J,2),XR1,XR2)
      A2(J,1)=A2(J,1)+XR1
      A2(J,2)=A2(J,2)+XR2
100   CONTINUE
D     TYPE 101,((A(I,K),K=1,2),I=1,NMANT1)
```

```
D101    FORMAT(' INNER PRODUCT AT VECTOR 1',/,(1X,4D15.8))
D       TYPE 102,((A2(I,K),K=1,2),I=1,NMANT1)
D102    FORMAT(' INNER PRODUCT AT VECTOR 2',/,(1X,4D15.8))
        DO 8100 I=1,NMANT1
        AS(I)=DSQRT(A(I,1)*A(I,1)+A(I,2)*A(I,2))
        AS2(I)=DSQRT(A2(I,1)*A2(I,1)+A2(I,2)*A2(I,2))
8100    CONTINUE
        TYPE 8105,(AS(I),I=1,NMANT1)
8105    FORMAT(/,' INNER PRODUCTS WITH TEST VECTOR 1 :',/4(1X,D15.8))
        TYPE 8106,(AS2(I),I=1,NMANT1)
8106    FORMAT(/,' INNER PRODUCTS WITH TEST VECTOR 2 :',/4(1X,D15.8))
        TYPE 3010
3010    FORMAT(//,' NUMBER OF EIGENVECTORS TO BE SELECTED BASED ON',
        1,' INNER PRODUCTS ?')
        ACCEPT 3015,MSET
3015    FORMAT(I2)
C
6120    CONTINUE
        DO 6170  I=1,NMANT1-1
        DO 6170 J=1,MSET
        V(I,J,2)=-V(I,J,2)
6170    CONTINUE
        SUMSQW=0.0D0
        I=NMANT1
        DO 6180 J=1,MSET
        SUMSQW=SUMSQW+V(I,J,1)*V(I,J,1)+V(I,J,2)*V(I,J,2)
6180    CONTINUE
D       TYPE 6181,SUMSQW
D6181   FORMAT(' SUMSQW=',1PD20.12)
        IF(DABS(1.0D0-SUMSQW).LT.1.0D-5) TYPE 6185
6185    FORMAT(/,' WARNING ABSOLUTE VALUE OF 1-SUMSQW IS LESS THAN 1.0D-
        15')
C
C       COMPUTE COEFF
C
        COEFF(NMANT1,1)=1.0D0
        CF=-(1.0D0-SUMSQW)
        COEFF(NMANT1,2)=0.0D0
        MP1=NMANT1
        DO 6199 I=1,NMANT1-1
        COEFF(I,1)=0.0D0
        COEFF(I,2)=0.0D0
        DO 6190 J=1,MSET
        COEFF(I,1)=COEFF(I,1)+V(MP1,J,1)*V(I,J,1)
        1                          -V(MP1,J,2)*V(I,J,2)
        COEFF(I,2)=COEFF(I,2)+V(MP1,J,1)*V(I,J,2)
        1                          +V(MP1,J,2)*V(I,J,1)
6190    CONTINUE
        COEFF(I,1)=COEFF(I,1)/CF
        COEFF(I,2)=COEFF(I,2)/CF
6199    CONTINUE
C
D       TYPE 6195,((COEFF(I,J),J=1,2),I=1,NMANT1)
D6195   FORMAT(' COEFF',/, 2(F20.15))
C
```

```
      NUMANT=NMANT1
      RETURN
      END
```

```
        SUBROUTINE SIGO18
C
C       ABOVE THRESHOLD EIGEN SELECTION
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        INTEGER REINIT,UPDATE,CASNUM
        COMMON/COMCOE/ COEFF(16,2),THRESH,ANTNIS
        COMMON/JACO/XMU(32,32),EIGEN(32)
        COMMON/ CONSNT / PI,TWOPI,ONEOPI
        DIMENSION XI(16,16,2)
        COMMON/H2H/CASNUM,IRND,JRND,REINIT,UPDATE,ITTY,NUMANT,ILOOP
        1,NUMSIG,MSET,METHOD
C
C       SELECT EIGEN VALUES ABOVE THRESHOLD
C       LET THE NUMBER OF SETS BE MSET
C
        IF(THRESH.NE.0.0D0) GOTO 4000
        TYPE 4010
4010    FORMAT(/,' THRESHOLD FOR SELECTING EIGENVECTORS ?',$)
        ACCEPT 4015,THRESH
4015    FORMAT(F10.0)
        GOTO 4020
4000    CONTINUE
        TYPE 61210,THRESH
61210   FORMAT(/,' THRESHOLD FOR SELECTING EIGENVECTORS :',1PD8.1)
4020    CONTINUE
        MSET=0
        DO 6100 M=1,2*NUMANT,2
        IF(EIGEN(M).GE.THRESH) MSET=MSET+1
6100    CONTINUE
        IF(MSET.GT.0) GOTO 6120
C
        TYPE 6110,THRESH,(EIGEN(I),I=1,2*NUMANT)
6110    FORMAT(' NO EIGEN VALUES ABOVE THRESH OF ',1PD20.13,
        1,/,' EIGEN VALUES ARE',/,4(D20.13))
        STOP ' THRESH ERROR'
C
6120    CONTINUE
        TYPE 61211,MSET
61211   FORMAT(///,' ESTIMATED NUMBER OF SOURCES :',I3)
D       TYPE 6121,THRESH,MSET,NUMANT
D6121   FORMAT(' SIGABT- THRESH=',D15.8,' MSET=',I6,', NUMANT=',I5)
        DO 6150 I=1,NUMANT
        K=0
        DO 6150 J=1,2*MSET,2
        K=K+1
        XI(I,K,1)=XMU(I,J)
        XI(I,K,2)=XMU(I+NUMANT,J)
6150    CONTINUE
C
        DO 6170 I=1,NUMANT-1
        DO 6170 J=1,MSET
        XI(I,J,2)=-XI(I,J,2)
6170    CONTINUE
C
```

```
C       COMPUTE LAMBDA, SUM OF THE SQUARES OF W
C
        SUMSQW=0.0D0
        I=NUMANT
        DO 6180 J=1,MSET
        SUMSQW=SUMSQW+XI(I,J,1)*XI(I,J,1)+XI(I,J,2)*XI(I,J,2)
6180    CONTINUE
D       TYPE 6181,SUMSQW
D6181   FORMAT(' SUMSQW=',1PD20.12)
        IF(DABS(1.0D0-SUMSQW).LT.1.0D-5) TYPE 6185
6185    FORMAT(/,' WARNING ABSOLUTE VALUE OF 1-SUMSQW IS LESS THAN 1.0D-
        15')
C
C       COMPUTE COEFF
C
        COEFF(NUMANT,1)=1.0D0
        CF=-(1.0D0-SUMSQW)
        COEFF(NUMANT,2)=0.0D0
        MP1=NUMANT
        DO 6199 I=1,NUMANT-1
        COEFF(I,1)=0.0D0
        COEFF(I,2)=0.0D0
        DO 6190 J=1,MSET
        COEFF(I,1)=COEFF(I,1)+XI(MP1,J,1)*XI(I,J,1)
        1                       -XI(MP1,J,2)*XI(I,J,2)
        COEFF(I,2)=COEFF(I,2)+XI(MP1,J,1)*XI(I,J,2)
        1                       +XI(MP1,J,2)*XI(I,J,1)
6190    CONTINUE
        COEFF(I,1)=COEFF(I,1)/CF
        COEFF(I,2)=COEFF(I,2)/CF
6199    CONTINUE
C
D       TYPE 6195,((COEFF(I,J),J=1,2),I=1,NUMANT)
D6195   FORMAT(' COEFF',/, 2(F20.15))
C
        RETURN
        END
```