CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

PHASE-ONLY ADAPTIVE NULLING

WITH DISCRETE VALUES

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science in

Engineering

by

Elise Tina Oestreich

August, 1979

The Thesis of Elise Tina Oestreich is approved:

Ephraim Mendelovicz

Fred Gruenberger

Edward J. Hriber, Chairman

California State University, Northridge

ACKNOWLEDGMENT

The author wishes to thank Dr. Ephraim Mendelovicz for suggesting the topic for this thesis and for his continuing encouragement and guidance.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	vi
LIST OF TABLES	vii
ABSTRACT	viii
Chapter 1 - INTRODUCTION	1
1.1 Background and Problem Statement	1
1.2 Array, Signal and Noise Models	2
1.3 Derivation of the Signal to Interference Plus Noise Ratio (SIR) for the Fully Adaptive Array . . .	6
1.4 Derivation of the Optimum Weight Vector for the Fully Adaptive Array	9
1.5 The Maximum SIR for an ESA with Discrete Phases . .	11
1.6 Thesis Outline	13
Chapter 2 - BRANCH AND BOUND	14
2.1 Introduction to Branch and Bound	14
2.2 SIR Maximization for the ESA with Discrete Phases .	15
2.3 Tree Structure	16
2.4 A Fundamental Mathematical Relation	19
2.5 Fathoming	20
2.6 Updating the Current Lower Bound	21
Chapter 3 - COMPUTER PROGRAM DESCRIPTION	22
3.1 Introduction to the Computer Program	22
3.2 Fully Adaptive Array Processing	24
3.2.1 MWQ	25
3.2.2 SIRCLB	29
3.3 ESA Processing - Pre-Branch and Bound (SIRCLB) . .	29

	Page
3.4 Branch and Bound: Discrete Weights and Assignment Priority	35
3.5 Branch and Bound: Loop Processing	40
3.6 ARFACT and APLOT	47
3.7 Trench-Zohar Complex Toeplitz Matrix Inversion Subroutine	47
Chapter 4 - RESULTS	48
4.1 Introduction	48
4.2 Results for Four Element Arrays	48
4.3 Results for Eight Element Arrays	55
4.4 Problems with Algorithm	59
Chapter 5 - CONCLUSIONS AND FUTURE WORK	63
REFERENCES	67
APPENDICES	
A. MAXIMIZATION OF THE SIGNAL TO INTERFERENCE RATIO. .	68
B. MAXIMUM SIR FOR A PARTIALLY CONSTRAINED WEIGHT VECTOR	77
C. COMPUTER PROGRAM LISTING	84

LIST OF FIGURES

	Page
1.1 Coherent Combiner for K Channels	3
1.2 Signal Arriving in the Direction θ_s from Broadside	4
2.1 Tree Structure for Branch and Bound	17
3.1 Block Diagram for the Computer Program	23
3.2 Interference Covariance Matrix	27
3.3 Flowchart for SIRCLB ESA Processing	30
3.4 Flowchart for WEIGHT Subroutine, p. 1	32
Flowchart for WEIGHT Subroutine, p. 2	33
3.5 Flowchart for Branch and Bound	36
3.6 Complex Plane Representation of Discrete Set (NBITS = 3)	41
4.1 Adapting Patterns with Jammer in Main Beam, ESA with 3 Bit Phase Shifters	50
4.2 Adapting Patterns with Jammer in First Sidelobe, ESA with 3 Bit Phase Shifters	51
4.3 Antenna Patterns with Jammer in Second Sidelobe, ESA with 3 Bit Phase Shifters	52
4.4 Comparison of Adapting Patterns with Jammer in Main Beam, ESA with 3 Bit Phase Shifters vs. 4 Bit Phase Shifters	53
4.5 Patterns for Eight Element Linear Array	60
5.1 Approximate Branch and Bound Search	66

LIST OF TABLES

	Page
4.1 Signal to (Noise Plus Interference) Ratios (dB), Jammer in 3 Different Positions	54
4.2 Signal to (Noise Plus Interference) Ratios, dB	56
4.3 Difference Between Adapted and Quiescent Patterns in Direction of Interference, dB	57
4.4 Difference Between Adapted and Quiescent Patterns in Direction of Signal, dB	58

ABSTRACT

PHASE-ONLY ADAPTIVE NULLING WITH DISCRETE VALUES

by

Elise Tina Oestreich

Master of Science in Engineering

In this thesis a method, using a branch and bound algorithm, is described for obtaining optimal phase settings for jammer cancellation with an electronically scanning array (ESA) whose only variable parameters are element phases which vary in discrete steps.

For the purpose of this study it is assumed that the interference environment is known. Therefore, no estimation of environmental noise input will be required.

The signal to interference plus noise ratio (SIR) for a linear array with an adaptive complex element weight vector was given in the literature. In particular the optimum weight vector that maximizes the SIR when the weights are unconstrained has been derived.

This solution is not valid when the values allowed for the components of the weight vector are restricted to a discrete set. In particular, in an ESA the only values available are those with constant

magnitude and discrete phases determined by the number of phase shifter bits. Methods of continuous analysis cannot produce a result for this discrete problem and an exhaustive search is impractical in most circumstances.

In contrast to the exhaustive search method, the branch and bound technique needs to examine only a small subset of all possible discrete weight vectors in order to arrive at the best SIR. The method has been tested and the jammer cancellation results in the test cases have been encouraging.

As a minimum, the approach described provides upper bounds for adaptation performance for electronically scanned arrays that are controlled with digital phase shifters.

Chapter 1

INTRODUCTION

1.1 Background and Problem Statement

Radar and communication systems must be designed to cope with directional interference, intentional and unintentional, in order to fill the needs of users. This requirement has led to increasing interest in adaptive antenna techniques which can sense and respond to a time-varying environment. Adaptive arrays have the capacity to null the reception from particular directions and thereby eliminate undesirable signals. An adaptive array can be implemented by associating a coherent receiver channel with each array element. The complex gain of the receiver is equivalent to a complex weight. This weight has magnitude and phase that vary over a continuous range of values. Adaptive arrays are over-equipped to handle electronic scanning since all that is required for that function is variable phase shifters. In many applications electronic scanning is necessary, while adaptive nulling capability is a desirable option, often left out, due to its high cost. Using phase control only reduces this cost and provides the electronic scan function. In addition it should provide an acceptable degree of nulling adaptivity. Arrays having phase control only are typically called electronically scanned arrays (ESA). This thesis provides a way to find the best adaptive nulling performance for an ESA with phases that vary in discrete steps.

Herein the term "fully adaptive array" is reserved for arrays with element weights that can vary continuously in magnitude and phase such as those analyzed by Widrow [1] and Applebaum [2].

1.2 Array, Signal and Noise Models

The signal and noise environment can be represented by a coherent combiner as shown in Figure 1.1. All desirable signals are narrowband and the noise is a narrowband, stationary process, which is additive and Gaussian with zero mean and unit variance. The interferences are also narrowband, zero mean stationary processes which are additive and Gaussian. Signals, noise and interference are represented by complex envelopes.

The desired input signal waveform modeled as a planar wave is represented in Figure 1.2. The horizontal line along the bottom of the drawing depicts a linear array with K elements. The incoming signal arrives at an angle θ_s from broadside. In this linear array it is assumed that the elements are equally spaced by a distance

$$d = \frac{\lambda}{2} \quad (1.1)$$

where λ is the wavelength. This value insures that there will be no grating lobes for any scanning angle θ_s . The equation for the signal in the k^{th} array element is given by

$$s_k = e^{-j \frac{2\pi kd}{\lambda} \sin \theta_s} \quad (1.2)$$

Given that n_i and n_j are the noise and interference components from two channels i and j the covariance of n_i and n_j is given by

$$m_{ij} = E\{n_i^* n_j\} \quad (1.3)$$

where $*$ represents complex conjugate.

Figure 1.1. Coherent Combiner for K Channels

Figure 1.2. Signal Arriving in the Direction θ_s from Broadside

It follows that

$$m_{ji} = E\{n_j^* n_i\} = E\{n_i n_j^*\} = m_{ij}^* \quad (1.4)$$

Thus, the covariance matrix

$$\underline{M} \triangleq [m_{ij}] \quad (1.5)$$

is Hermitian. The power in the i^{th} channel is given by

$$m_{ii} = E\{n_i^* n_i\} = |n_i|^2 \quad (1.6)$$

Note that the covariance matrix is made up of two components

$$\underline{M} = \underline{M}_i + \underline{M}_n \quad (1.7)$$

where $\underline{M}_i$ represents the covariance matrix due to interference alone

and $\underline{M}_n$ denotes the covariance matrix for noise alone,

where

$$\underline{M}_n = \begin{bmatrix} p_n & 0 \\ 0 & p_n \end{bmatrix} = p_n \underline{I}_K \quad (1.8)$$

with $\underline{I}_K \triangleq$ identity matrix of order K

and $p_n \triangleq$ thermal noise power in each array element.

The derivation of the covariance matrix $\underline{M}_i$ is shown next. The voltage due to interference i in element k is shown in Equation 1.9.

$$v_{ki} = v_i e^{-j \frac{2\pi k d}{\lambda} \sin \theta_i} = v_i e^{-jk\beta_i} \quad (1.9)$$

where

$$\beta_i = \frac{2\pi d}{\lambda} \sin \theta_i \quad (1.10)$$

The interference covariance of the k^{th} and ℓ^{th} array elements is given by

$$\begin{aligned} m_{k\ell_i} &= E\{v_{ki}^* v_{\ell i}\} = E\{v_i^* e^{jk\beta_i} v_i e^{-j\ell\beta_i}\} \\ &= e^{j(k-\ell)\beta_i} E\{|v_i|^2\} \\ &= p_i e^{j(k-\ell)\beta_i} \end{aligned} \quad (1.11)$$

The interference covariance matrix may now be given as

$$\underline{M}_i = [m_{k\ell_i}] = \begin{bmatrix} p_i e^{j(k-\ell)\beta_i} \end{bmatrix} \quad (1.12)$$

where $p_i \triangleq$ interference power in each array element.

For the purpose of this study it is assumed that the interference covariance is given and unchanging so that no estimation of the interference environment such as interference power(s) and direction(s) are required. When there are multiple jammers, the interference covariance matrix elements are of the form of a summation of elements described in Equation 1.12.

1.3 Derivation of Signal to Interference Plus Noise Ratio (SIR) for the Fully Adaptive Array

In this section, it will be shown that the signal to interference

plus noise ratio (SIR) is given by

$$\text{SIR} = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^\dagger \underline{M} \tilde{W}} \quad (1.13)$$

where t represents transpose, $\dagger$ is the conjugate transpose, and

$$\tilde{W} = \begin{bmatrix} w_1 \\ \cdot \\ \cdot \\ w_k \end{bmatrix} = \text{adaptive complex array weight vector} \quad (1.14)$$

$$\tilde{W}_q = \begin{bmatrix} w_{q1} \\ \cdot \\ \cdot \\ w_{qk} \end{bmatrix} = \text{unadapted (quiescent) weight vector} \quad (1.15)$$

representing the optimum compromise
between such performance factors as
resolution, gain, and low sidelobes

In order to derive Equation 1.13 it should be noted that the signal output from the combiner (Figure 1.1) is a real scalar value.

$$v_s = \sum_{k=1}^K w_k s_k \quad (1.16)$$

In vector notation it is given as

$$v_s = \tilde{W}^t \tilde{S} = \tilde{S}^t \tilde{W} \quad (1.17)$$

$\tilde{S}$ was modeled as a planar waveform and is completely defined when the direction θ_s is given.

In order to maximize v_s in Equation 1.17 it is sufficient to set

$$\tilde{W}_q = \tilde{S}^* \quad (1.18)$$

so that all the voltages in the array elements add up coherently.

When Equation 1.18 is satisfied, the array illumination is uniform with the attendant high sidelobes.

The signal power can now be obtained from Equations 1.17 and 1.18 as

$$P_s = |v_s|^2 = |\tilde{W}^t \tilde{W}_q^*|^2 \quad (1.19)$$

The noise output may be expressed as the scalar value

$$v_n = \tilde{W}^t \tilde{N} = \tilde{N}^t \tilde{W} \quad (1.20)$$

where

$$\tilde{N} = \begin{bmatrix} n_1 \\ \cdot \\ \cdot \\ n_k \end{bmatrix} = \text{noise and interference vector}$$

The expected noise power can be represented by

$$\begin{aligned} P_N &= E \left\{ |v_n|^2 \right\} = E \left\{ |\tilde{W}^t \tilde{N}|^2 \right\} = E \left\{ (\tilde{W}^t \tilde{N})^* (\tilde{W}^t \tilde{N}) \right\} \\ &= E \left\{ (\tilde{W}^\dagger \tilde{N}^*) (\tilde{N}^t \tilde{W}) \right\} = E \left\{ \tilde{W}^\dagger (\tilde{N}^* \tilde{N}^t) \tilde{W} \right\} \end{aligned}$$

Since the expectation operator only operates on the random noise terms,

$$P_n = \tilde{W}^\dagger \left(E \left\{ \tilde{N}^* \tilde{N}^t \right\} \right) \tilde{W}$$

or

$$P_n = \tilde{W}^\dagger \underline{M} \tilde{W} \quad (1.21)$$

where

$$\underline{M} = E \left\{ \tilde{N}^* \tilde{N}^t \right\} = \left[m_{ij} \right] \quad (1.22)$$

Therefore, the ratio of the signal power to the noise (plus interference) power is given by

$$SIR = \frac{P_s}{P_n} = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^\dagger \underline{M} \tilde{W}} \quad (1.13)$$

1.4 Derivation of the Optimum Weight Vector for the Fully Adaptive Array

The corresponding optimum weight vector which maximizes the SIR is given in this section but its derivation is deferred to Appendix A. This result is due to Applebaum [2]. However, the proof given in the appendix follows other authors.

The optimum weight vector that maximizes Equation 1.13 is given by

$$\tilde{W} = \underline{M}^{-1} \tilde{W}_q \quad (1.23)$$

The numerator in Equation 1.13 is shown in the appendix to be a quadratic form,

$$|\tilde{W}^t \tilde{W}_q^*|^2 = \tilde{W}^\dagger \underline{A} \tilde{W} \quad (1.24)$$

with

$$\underline{A} = \begin{bmatrix} \underline{W} & \underline{W}^\dagger \\ \underline{\sim q} & \underline{\sim q} \end{bmatrix}$$

Similarly, the denominator of Equation 1.13 is a quadratic form.

Since $\underline{M}$ is the covariance matrix for a stationary random process, the denominator is positive definite. The equation for the signal to interference ratio is now given by the ratio of quadratic forms

$$SIR = \frac{\underline{W}^\dagger \underline{A} \underline{W}}{\underline{W}^\dagger \underline{M} \underline{W}} \quad (1.25)$$

Therefore the following theorem from Gantmacher [8] can be used.

Theorem: Given Hermitian matrices $\underline{C}$ and $\underline{D}$ of order $n \times n$ with $\underline{D}$ positive definite, the equation

$$|\underline{C} - \lambda \underline{D}| = 0$$

has in general n distinct values for λ (eigenvalues).

Then:

The largest characteristic value λ_n is the maximum of the ratio of two quadratic forms and is given by

$$\lambda_n = \max_{\underline{W}} \frac{\underline{W}^\dagger \underline{C} \underline{W}}{\underline{W}^\dagger \underline{D} \underline{W}}$$

and this maximum is assumed only for the eigenvector $\underline{W}$ associated with the characteristic value λ_n which satisfies the following homogeneous equation

$$\underline{C} \underline{\tilde{W}} = \lambda_n \underline{D} \underline{\tilde{W}}$$

Applying this theorem to Equation 1.25, it is shown in Appendix A that

- a) The largest eigenvalue λ_M of the linear combination $\underline{A} - \lambda \underline{M}$ is its only nonzero value and is given by

$$\lambda_M = \underline{W}_{\sim q} \underline{M}^{-1} \underline{W}_{\sim q}$$

- b) The eigenvector corresponding to λ_M is

$$\underline{W}_M = \underline{M}^{-1} \underline{W}_{\sim q}$$

Thus λ_M provides the maximum obtainable SIR for the array and $\underline{W}_M$ is the excitation required to achieve it.

1.5 The Maximum SIR for an ESA with Discrete Phases

The solution given in the previous section is not valid when the values allowed for the components of $\underline{W}$ are restricted to a discrete set. In particular, in an ESA the only available values are

$$\left\{ e^{j2\pi(k-1)/L}, k=1, \dots, L=2^b \right\}$$

where b is the number of phase shifter bits.

Methods of continuous analysis cannot produce a result for this discrete problem. An exhaustive search is impractical in most circumstances. To demonstrate this, the total computational count will now be derived. Note that there will be L^N points to be considered where N is the number of array elements. The computational count for the

SIR is obtained as follows. From Equation 1.13

$$SIR = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^t \underline{M} \tilde{W}} \quad (1.13)$$

$N + 1$ multiplications are required to calculate the numerator, assuming that $\tilde{W}$ and $\tilde{W}_q$ are N -dimensional.

$N^2 + N$ multiplications are performed in calculating the denominator, assuming that $\underline{M}$ is $N \times N$. This results in the following number of complex multiplications required each time the SIR is calculated

$$N^2 + 2N + 1 \approx N^2 \quad (1.26)$$

This can be approximated by $4N^2$ real multiplications.

Searching for the optimum weight vector by exhaustion amounts to calculating the SIR for L^N cases with a total real operational count of

$$L^N (4N^2) = 4L^N N^2 \quad (1.27)$$

For example, consider a small array with 16 elements and 4-bit phase shifters. Then there are

$$4(16)^{16} (16)^2 = 1.89 \times 10^{22}$$

real multiplications. Assuming optimistically that one nanosecond is required for each multiplication, using the exhaustive search method would take

$$\frac{1.89 \times 10^{22} \times 10^{-9}}{3600 \times 24 \times 365} = 600,000 \text{ years}$$

to complete.

An efficient search algorithm suitable for this problem is known as branch and bound. This method reduces the number of points in the search space and still leads to a proven solution. This procedure has been used to obtain the results in this thesis.

1.6 Thesis Outline

The branch and bound algorithm is described in Chapter 2.

Chapter 3 provides a description of the computer program used to implement the branch and bound algorithm and generate numerical results. A functional block diagram is presented as well as flowcharts for the main processing and subroutines. Included in the descriptions are the branching rule and the Trench-Zohar Toeplitz matrix inversion method. In addition, a program listing is provided in Appendix C.

Chapter 4 contains the numerical results generated by the computer program. The branch and bound program has been used to evaluate four and eight element linear arrays with element spacing $\lambda/2$. Results have been obtained for arrays with 2 to 4 bit phase shifters. An untapered array illumination is assumed for the unadapted pattern since sidelobe control in the unadapted pattern is not essential to demonstrate the adaptation capacity of the ESA. One source of interference was stationed in different angular positions and the nulling capacities of the arrays were determined.

The thesis concludes with Chapter 5 in which the conclusions derived from this study are summarized and suggestions for future work are made.

Chapter 2

BRANCH AND BOUND

2.1 Introduction to Branch and Bound

The need for an efficient search algorithm suitable for obtaining the maximum signal to interference ratio (SIR) for a linear array with complex element weight vector $\underline{W}$ whose components are restricted to a discrete set was demonstrated in the previous chapter. It was shown that for a small 16 element array with 4 bit phase shifters, 600,000 years would be required to exhaustively search all cases on a digital computer.

The branch and bound algorithm is an efficient search technique which reduces the number of points in the search space and still leads to a proven solution in a realistic length of time. The version used to obtain the results of this thesis is described by Cohen [3]. Cohen considers clutter suppression in relation to the determination of transmitter coherent pulse train amplitudes. That problem is cast in identical mathematics as the problem at hand. In the array problem N dimensional complex formulation was chosen over $2N$ dimensional real formulation. Complex formulation preserves the Toeplitz form of the covariance matrices allowing the use of faster inversion algorithms (Trench-Zohar [4], [5]). For this reason complex formulation was chosen. This differs from Cohen's implementation and brought about a different branching rule as well as the need to use mathematical results and extensions thereof from Spafford [6]. Spafford gives the optimum subvector of weights over the continuum when the rest of the weight vector has been constrained to the allowed discrete values. This

result is essential in the use of the branch and bound algorithm.

2.2 SIR Maximization for the ESA with Discrete Phases

In Chapter 1 it was shown that the equation for the signal to interference ratio for a fully adaptive array is given by

$$SIR(\tilde{W}) = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^\dagger \underline{M} \tilde{W}} \quad (2.1)$$

For an ESA the elements of the weight vector $\tilde{W}$ are constrained to the discrete set

$$A = \{a_k : a_k = e^{j2\pi(k-1)/L}, k=1, 2, \dots, L=2^b\} \quad (2.2)$$

where b is the number of phase shifter bits.

We wish to find the maximum signal to interference ratio for an ESA using the branch and bound algorithm. Mathematically this can be formulated as follows.

Find:

$$\begin{aligned} \text{Max } SIR(\tilde{W}), \quad \tilde{W}^t = [w_1, w_2, \dots, w_N] \\ \tilde{W} \in A \end{aligned} \quad (2.3)$$

with $m = 1, 2, \dots, N$

For the fully adaptive array the weight vector $\tilde{W}$ has elements which are unconstrained. The ESA has weights that are constrained both in magnitude and phase. The intermediate cases in which some elements of the weight vector are constrained is used in the branch and bound algorithm. The specific notation for the partition of the N^{th} order weight vector $\tilde{W}$ into its constrained and unconstrained parts is given by

$$\tilde{W}^{\ell} = (\tilde{W}_c^{\ell}, \tilde{W}_u^{\ell}) \quad (2.4)$$

where $\tilde{W}_c^{\ell} = (w_1, w_2, \dots, w_{\ell})$ = constrained subvector of weights

$\tilde{W}_u^{\ell} = (w_{\ell+1}, \dots, w_N)$ = unconstrained subvector of weights

$\ell = 0, 1, 2, \dots, N$ = level indicator

N = number of linear array elements

Further explanation of this notation will follow in the description of tree structures used in the branch and bound method.

Equation 2.1 can now be restated as follows for the partially constrained weight vector

$$SIR(\tilde{W}) = \frac{|(\tilde{W}_c^{\ell t}, \tilde{W}_u^{\ell t}) \tilde{W}_q^*|^2}{(\tilde{W}_c^{\ell t}, \tilde{W}_u^{\ell t}) \underline{M} (\tilde{W}_c^{\ell}, \tilde{W}_u^{\ell})} \quad (2.5)$$

2.3 Tree Structure

Equation 2.5 is used with the branch and bound algorithm for a variety of possible values for ℓ and for each ℓ , different weight values are assigned from the set A . Since any combination of ℓ values and weight values may be needed, a complete enumeration of all possibilities is required. All such combinations can be visualized with the aid of a tree structure shown in Figure 2.1. It was mentioned in Equation 2.4 that the level indicator ℓ varies between 0 and N . Referring to Figure 2.1, $\ell = 0$ refers to the top node of the tree. At this node the SIR is given for the totally unconstrained N^{th} order

Figure 2.1 Tree Structure for Branch and Bound

weight vector by Equation 2.1. Continuing down the tree, the next level is denoted by $\ell = 1$. The SIR is now a function of the partially constrained weight vector $\tilde{W}^1$ and is given below.

$$SIR(\tilde{W}^1) = SIR(\tilde{W}_c^1 = (w_1), \tilde{W}_u^1 = (w_2, \dots, w_N))$$

In this case the SIR $(\tilde{W}_c^1, \tilde{W}_u^1)$ is given by Equation 2.5. At the k^{th} level the SIR is denoted by

$$SIR(\tilde{W}_c^k, \tilde{W}_u^k)$$

where $\tilde{W}_c^k = (w_1, w_2, \dots, w_k)$

$$\tilde{W}_u^k = (w_{k+1}, \dots, w_N)$$

The next level down is denoted by $k+1$ and the following method is used for constraining the $(k+1)^{st}$ element of $\tilde{W}_c^{k+1}$.

$$\tilde{W}_c^{k+1} = (\tilde{W}_c^k, w_{k+1} = a_1)$$

or

$$\tilde{W}_c^{k+1} = (\tilde{W}_c^k, w_{k+1} = a_2)$$

or

$$\tilde{W}_c^{k+1} = (\tilde{W}_c^k, w_{k+1} = a_L)$$

Note that each node branches to L values for the next level in the tree. The unconstrained subvector changes in the following manner. At the k^{th} level the vector is

$$\tilde{w}_u^k = (w_{k+1}, w_{k+2}, \dots, w_N) \text{ of order } N-k$$

Constraining the $(k+1)^{\text{st}}$ element to a_k , yields

$$\tilde{w}_u^{k+1} = (w_{k+2}, \dots, w_N) \text{ of order } N-k-1.$$

Level $\ell=N$ denotes the bottom of the tree. The SIR only depends upon the fully constrained vector of weights and is denoted by

$$\text{SIR } (\tilde{w}_c^N)$$

In summary, at the top of the tree all of the elements in the weight vector are unconstrained. At each level down in the tree another element of the weight vector is constrained to a member of the discrete set A of allowable weight values. At the bottom of the tree, all of the elements of the weight vector are constrained to elements in the allowable discrete set.

2.4 A Fundamental Mathematical Relation

The following relation is the central concept of the branch and bound method as presented here. It provides the justification for decreasing the number of points in the search space for the maximum SIR.

Note that for two sets $\Omega'' \subset \Omega'$

$$\max_{y \in \Omega'} f(y) \geq \max_{y \in \Omega''} f(y) \quad (2.6)$$

This is due to the fact that Ω'' is contained within Ω' and Ω'' has fewer elements than Ω' .

The inequality given in Equation 2.6 can be translated into tree notation as shown below.

$$\max_{\substack{W_{\sim u}^{\ell} \\ \in C^{N-\ell}}} \text{SIR}(\tilde{W}_{\sim c}^{\ell}, \tilde{W}_{\sim u}^{\ell}) \geq \max_{\substack{W_{\sim u}^{\ell+1} \\ \in C^{N-\ell-1}}} \text{SIR}(\tilde{W}_{\sim c}^{\ell+1}, \tilde{W}_{\sim u}^{\ell+1}) \quad (2.7)$$

where $C^{N-\ell}$ = $N-\ell$ dimensional complex space

$C^{N-\ell-1}$ = $N-\ell-1$ dimensional complex space

and $C^{N-\ell-1} \subset C^{N-\ell}$

2.5 Fathoming

The SIR is evaluated at the bottom of the tree when all elements of the weight vector have been constrained to values from the discrete set. The SIR, a ratio of quadratic forms, takes on real non-negative values. Let $\text{SIR}(\tilde{W}_1) = C_1$, where $\tilde{W}_1$ is a vector such that all of its components are in A and C_1 is a real constant. This constitutes a feasible solution to the problem stated in Equation 2.3. Note that C_1 is actually a current lower bound (CLB) to the solution.

Tracing the way down the tree, at a particular node the following inequality may hold.¹

$$\max_{\substack{W_{\sim u}^{\ell} \\ \in C^{N-\ell}}} \text{SIR}(\tilde{W}_{\sim c}^{\ell}, \tilde{W}_{\sim u}^{\ell}) < C_1$$

-
1. The specific equation used to calculate the maximum SIR for the partially constrained weight vector is derived analytically in Appendix B. Due to its complexity the equation and pertaining notation will not be presented here in order to avoid loss of continuity.

If this occurs, that node cannot lead to a feasible solution and it is declared fathomed. All succeeding nodes branching out below the fathomed node are automatically fathomed as well because of Equation 2.7.

2.6 Updating the Current Lower Bound

In the event that a node is not fathomed during the search through the tree and the bottom of the tree is reached, the SIR is calculated.
If

$$SIR = C_2 > C_1$$

then set $SIR = C_2$. This procedure is called updating the current lower bound for the maximum SIR. After this substitution occurs, the algorithm moves from the N^{th} level to level $N-1$. It will then find the SIR for the next node at the N^{th} level. After all nodes at the N^{th} level branching from the same node at level $N-1$ have been exhausted, the algorithm will move back to level $N-2$. It will then proceed to all nodes at level $N-1$ that branch from the same node at level $N-2$. It should be noted that when

$$1 \leq \ell \leq N-1$$

the maximum SIR is calculated at each node using the equation in Appendix B, and it is this value which is compared to the current lower bound for fathoming.

This process of tracing branches continues all the way back to the top of the tree. After all nodes have been exhausted in this manner, the current lower bound is the desired solution and the corresponding weight vector is $\tilde{W}_{opt}$.

Chapter 3

COMPUTER PROGRAM DESCRIPTION

3.1 Introduction to the Computer Program

This chapter describes the computer program implementation of the equations given in Chapter 1 for the fully adaptive array and the branch and bound algorithm described in the previous chapter for the ESA. A functional block diagram for the computer program is given in Figure 3.1.

The program first calculates the noise and interference covariance matrix $\underline{M}$, assuming that the jammer(s) power(s) and direction(s) are known and unchanging. It then calculates the quiescent or unadapted weight vector $\underline{W}_q$ which is determined when the mainbeam pointing direction is specified. With these two quantities the value for the maximum SIR for the fully adaptive array and the corresponding optimum weight vector are easily obtained from the equations given in Chapter 1.

$$SIR_{\max} = \frac{|\underline{W}_q^T \underline{W}_q^*|^2}{\underline{W}_q^T \underline{M} \underline{W}_q} \quad (3.1)$$

where

$$\underline{W} = \underline{W}_{\text{opt}} = \underline{M}^{-1} \underline{W}_q \quad (3.2)$$

The value for the maximum SIR is not used in the branch and bound algorithm since it represents the best performance for a fully adaptive array with weights having continuously varying magnitudes and phases. However, it is used to determine the relative performance of an ESA as

Figure 3.1 Block Diagram for the Computer Program

the number of phase shifters bits increases.

The ESA portion of the program begins with the third block in Figure 3.1. This part of the program uses a random number generator to select all of the discrete elements of the weight vector. Then the SIR is calculated using Equation 3.1 without using Equation 3.2. Many possible weight vectors are generated with the random number generator and the corresponding SIRs are compared to obtain the largest. This value is stored as the current lower bound. This process enables the branch and bound algorithm to converge more rapidly to the desired maximum SIR.

The next block represents the branch and bound algorithm. The theoretical concepts were introduced in Chapter 2. A more detailed description follows in Section 3.4.

The antenna patterns for the unadapted, fully adaptive and ESA are then plotted to determine jammer cancellation results. More accurate quantitative results are derived and tabulated from additional printed output.

The description of the high level structure of the computer program was given above; the next sections will give detailed descriptions of the computer algorithms.

3.2 Fully Adaptive Array Processing

The fully adaptive array processing includes two subroutines: MWQ which calculates the noise and interference covariance matrix $\underline{M}$ and the quiescent weight vector $\underline{W}_q$ and SIRCLB in which the maximum SIR for the fully adaptive array and the corresponding weight vector $\underline{W}_{opt}$ are determined. The following subsections will discuss each of these

routines in greater detail.

3.2.1 MWQ

It was shown in Chapter 1 that the noise plus interference covariance matrix is given by

$$\underline{M} = \underline{M}_i + \underline{M}_n$$

where

$\underline{M}_i$ represents the covariance matrix due to interference alone and

$\underline{M}_n$ denotes the covariance matrix for the noise alone,

where

$$\underline{M}_n = \begin{bmatrix} p_n & & 0 \\ & \ddots & \\ 0 & & p_n \end{bmatrix} = p_n \underline{I}_k$$

with

$\underline{I}_k$ = identity matrix of order k

and

p_n = thermal noise power in each array element.

The interference covariance matrix is given as

$$\underline{M}_i = \begin{bmatrix} p_i e^{j(k-l)\beta_i} \end{bmatrix} \quad (3.3)$$

where

p_i = interference power in each array element

k, ℓ are the k^{th} and ℓ^{th} channel, respectively

$$\beta_i = \frac{2\pi d}{\lambda} \sin \theta_i$$

θ_i = direction of the interference

For the case of multiple jammers the interference covariance matrix is given as a sum of terms similar to the right side of Equation 3.3.

For the interference covariance in Equation 3.3 whenever $(k-\ell) = m$, where m is a fixed integer, we obtain identical elements. In each successive row, one column over to the right, the element value is repeated as shown in Figure 3.2 for $N=4$ elements. Thus all diagonals parallel to the main diagonal have identical elements. This structure is called a Toeplitz matrix. Note also that the elements on opposite sides of the main diagonal are complex conjugates. Therefore, this matrix is Hermitian and Toeplitz. If $\ell = 1$, the following formulation is used to obtain the elements in the first row of Equation 3.3.

$$\begin{bmatrix} p_i & e^{j(k-1)\beta_i} \end{bmatrix} \quad (3.4)$$

Then the Toeplitz property is used to find the elements on the main diagonal and in the upper triangular portion of the matrix. Finally, the Hermitian property is used to fill the lower triangular portion.

The total noise plus interference covariance matrix is now obtained by adding the noise power p_n to the main diagonal elements of

$$\begin{bmatrix}
 P_i \cdot 1 & P_i e^{j\beta_i} & P_i e^{j2\beta_i} & P_i e^{j3\beta_i} \\
 P_i e^{-j\beta_i} & P_i \cdot 1 & P_i e^{j\beta_i} & P_i e^{j2\beta_i} \\
 P_i e^{-j2\beta_i} & P_i e^{-j\beta_i} & P_i \cdot 1 & P_i e^{j\beta_i} \\
 P_i e^{-j3\beta_i} & P_i e^{-j2\beta_i} & P_i e^{-j\beta_i} & P_i \cdot 1
 \end{bmatrix}$$

Figure 3.2 Interference Covariance Matrix

the interference covariance matrix.

The quiescent or unadapted weight vector which is the optimum compromise between resolution, gain and low sidelobes is calculated from the signal direction angle θ_s as shown in the following equations:

$$\tilde{W}_q = \begin{bmatrix} w_{q1} \\ w_{q2} \\ \vdots \\ w_{qN} \end{bmatrix} = \text{quiescent weight vector}$$

where the k^{th} element is denoted by

$$w_{qk} = a_k e^{-j(k-1)\beta_s}$$

and

$$\beta_s = \frac{2\pi d}{\lambda} \sin \theta_s$$

where

d = distance between elements

λ = wavelength

θ_s = desired signal direction angle

For this study an untapered array illumination is assumed for the unadapted pattern since sidelobe control in the unadapted pattern is not essential to demonstrate the adaptation capacity of the fully adaptive array or the ESA. Therefore,

$$a_k = 1$$

3.2.2 SIRCLB

The maximum SIR for the fully adaptive array and the corresponding optimum weight vector are calculated using the following equations from Chapter 1.

$$SIR_{\max} = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^t \underline{M} \tilde{W}} = \frac{\tilde{W}^+ \tilde{W}_q \tilde{W}_q^+ \tilde{W}}{\tilde{W}^+ \underline{M} \tilde{W}} \quad (3.5)$$

where

$$\tilde{W} = \tilde{W}_{\text{opt}} = \underline{M}^{-1} \tilde{W}_q \quad (3.6)$$

Substituting Equation 3.6 into Equation 3.5 we obtain the following formula for the maximum SIR.

$$\begin{aligned} SIR_{\max} &= \frac{\tilde{W}_q^+ \underline{M}^{-1} \tilde{W}_q \tilde{W}_q^+ \underline{M}^{-1} \tilde{W}_q}{\tilde{W}_q^+ \underline{M}^{-1} \underline{M} \underline{M}^{-1} \tilde{W}_q} \\ &= \tilde{W}_q^+ \underline{M}^{-1} \tilde{W}_q \\ &= \tilde{W}_q^+ \tilde{W}_{\text{opt}} \end{aligned}$$

3.3 ESA Processing -- Pre-Branch and Bound (SIRCLB)

In addition to obtaining the maximum SIR and the corresponding optimum weight vector for the fully adaptive array, the SIRCLB routine represented in Figure 3.3, performs another function, for the ESA which is described next.

The largest current lower bound for the maximum SIR for an ESA

Figure 3.3 Flowchart for SIRCLB ESA Processing

where all weights are constrained to discrete complex values is calculated in SIRCLB. This situation corresponds to a feasible solution for the maximum SIR at the bottom of the tree for an ESA as described in Chapter 2.

The discrete weight vector is calculated in a subroutine named WEIGHT which is called from SIRCLB.

The following processing occurs in the WEIGHT subroutine as shown in Figure 3.4. First a sequence of N random numbers is generated and written into a vector $\tilde{R}$ of order N where

$$\tilde{R} = \begin{bmatrix} r_1 \\ r_2 \\ . \\ . \\ r_N \end{bmatrix}$$

Then each element of the weight vector $\tilde{W}$ is assigned a value based on the entries in $\tilde{R}$ using the following inequality.

$$\text{If } \frac{(L_1-1)}{2^b} \leq r_k < \frac{L_1}{2^b} \quad (3.7)$$

where

$$L_1 = 1, 2, \dots, L = 2^b$$

b = number of phase shifter bits, and

$$r_k = k^{\text{th}} \text{ entry in } \tilde{R} \in [0,1]$$

Figure 3.4 Flowchart for Weight Subroutine, p. 1

Figure 3.4 Flowchart for Weight Subroutine, p. 2

Then the k^{th} element in the weight vector $\tilde{W}$ is given by

$$w_k = a_k e^{j\beta_k} \quad (3.8)$$

where

$$a_k = 1$$

$$\beta_k = \frac{-2\pi(L_1)}{2^b}$$

Equation 3.7 assigns a value to L_1 based on the value of r_k , which varies from 0 to 1. This value for L_1 is used in Equation 3.8 to determine the discrete phase for the weight vector. After the N elements of $\tilde{W}$ have been chosen in this manner, the subroutine returns to SIRCLB.

After the program has returned to the SIRCLB routine the SIR is calculated using the weight vector $\tilde{W}$ computed in the WEIGHT subroutine and the following equation:

$$SIR = \frac{|\tilde{W}^t \tilde{W}^*|^2}{\tilde{W}^\dagger \underline{M} \tilde{W}}$$

The SIR is always positive since the denominator is positive definite. This value is compared to the value for the current lower bound for the maximum SIR, ψ_{clb} , which is initialized to 0. If $SIR > \psi_{clb}$, the value for SIR is stored in ψ_{clb} and the weight vector $\tilde{W}$ is stored in $\tilde{W}_{clb}$.

The SIRCLB routine processes 2,000 weight vectors in this manner. The number 2,000 was selected because this should constitute a sufficient statistical sample. After this processing is concluded, control

returns to the MAIN program where the branch and bound logic is implemented.

3.4 Branch and Bound: Discrete Weights and Assignment Priority

The flowchart for the computer program implementing the branch and bound concepts described in Chapter 2 is shown in Figure 3.5. The largest current lower bound for the maximum SIR is initialized to ψ_{clb} which is an output from the SIRCLB subroutine. The branch and bound program next computes the allowed discrete set of weight values:

$$A_{mr} = \left\{ a_{mr_k} : a_{mr_k} = e^{j(k-1)2\pi/MDIM}, k = 1, 2, \dots, MDIM = 2^{NBITS} \right\}$$

where NBITS is the number of phase shifter bits.

The index I in the next block of the flowchart corresponds to the level indicator, L, plus 1. The reason for this convention is to avoid an illegal FORTRAN subscript of 0 (level 0, or the top of the tree). NPASS keeps track, at a given level L, of which tree branch of those emanating from one vertex, the program is processing. As shown in Figure 2.1, there are $L \triangleq MDIM = 2^{NBITS}$ branches originating from a node (vertex) at the previous level. Therefore, the values assigned to NPASS are 1, 2, ..., MDIM. Since the level indicator L is initialized to 0, the initial value for NML = N-L is N.

The program now calls two subroutines, ANGLE and R1R2 prior to constraining the elements in the weight vector to members of the allowable discrete set. This sequence is performed one level at a time. At the first level, L = 1, the first element in the weight vector is constrained; at the second level, the second weight is constrained, etc.

INITIALIZATION

Figure 3.5 Flowchart for Branch and Bound, p. 1

BRANCH AND BOUND
LOOP PROCESSING

Figure 3.5 Flowchart for Branch and Bound, p. 2

BOTTOM OF THE TREE

Figure 3.5 Flowchart for Branch and Bound, p. 3

FATHOM VERTEX

Figure 3.5 Flowchart for Branch and Bound, p. 4

In order to select the constrained discrete value, the program calls subroutine ANGLE which computes the angle, α , in the complex plane corresponding to the first element in the optimum weight vector $\tilde{w}_{opt}$ using the following formula:

$$\alpha = \tan^{-1} \left(\frac{\text{Im} \left(\tilde{w}_{opt_1} \right)}{\text{Re} \left(\tilde{w}_{opt_1} \right)} \right)$$

This value for a α is now used in subroutine R1R2 to select the allowable discrete value for the first element in the weight vector. Two vectors, $\tilde{R}_1$ and $\tilde{R}_2$, are defined in R1R2. Related vectors $\tilde{R}_3, \tilde{R}_4, \dots, \tilde{R}_{MDIM}$ where $MDIM = 2^{NBITS}$ are determined in another subroutine, RHIGH, which is described later. The method of selecting the discrete weight value is demonstrated with the aid of Figure 3.6, for $NBITS = 3$. Noting that the vector corresponding to angle α intersects the unit circle at a point denoted by p , which is closest to the discrete value a_2 , the program will select 2 as the first entry in $\tilde{R}_1$. The second closest value to p is a_1 . Therefore, the first entry in $\tilde{R}_2$ is 1. The point a_3 is the third closest value to p . Therefore, the first entry in $\tilde{R}_3$ is 3. The first entry in $\tilde{R}_4, \tilde{R}_5, \dots, \tilde{R}_{MDIM}$ is determined in the same manner.

In summary, R1R2 produces the vectors $\tilde{R}_1$ and $\tilde{R}_2$ which contain the first and second closest allowable discrete values to be assigned to a given weight, and they are indexed by level L in the branch and bound tree.

3.5 Branch and Bound: Loop Processing

As described in Chapter 2, the unconstrained portion of the weight

Figure 3.6 Complex Plane Representation of Discrete Set (NBITS = 3)

vector is of order $N-L$, while the constrained portion is of order L . L will take on values from 0 to N , where N is the number of elements in the weight vector. Conversely, $NML = N-L$ takes on values from N down to 0.

The vector $\tilde{B}_{\text{conc}}$ represents the constrained subvector of weights, which is indexed by L . The discrete value vector, $\tilde{A}_{\text{mr}}$, is indexed by either $\tilde{R}_1$ or $\tilde{R}_2$, depending on an indicator. $\tilde{R}_1$ and $\tilde{R}_2$ are indexed by level L .

I is compared to $N+1$ to determine if the processing has reached the bottom of the tree. If not, subroutine REORDR is called. This subroutine partitions all of the vectors and the noise and interference covariance matrix $\underline{M}$ which are used to calculate the maximum SIR for the partially constrained weight vector, as shown in Appendix B. The following vectors are partitioned and reordered in this context:

$$\tilde{A}_0 = \begin{bmatrix} a_{01} \\ \vdots \\ a_{0L} \end{bmatrix}_L = \begin{bmatrix} w_{q1} \\ \vdots \\ w_{qL} \end{bmatrix}_L = \text{"constrained" portion}$$

$$\tilde{A}_T = \begin{bmatrix} a_{t1} \\ \vdots \\ a_{t(N-L)} \end{bmatrix}_{N-L} = \begin{bmatrix} w_{q_{L+1}} \\ \vdots \\ w_{q_N} \end{bmatrix}_{N-L} = \text{"unconstrained" portion}$$

where

$$\begin{aligned} \tilde{W}_q &= \begin{bmatrix} w_{q_1} \\ \vdots \\ w_{q_N} \end{bmatrix}_N = \text{quiescent weight vector} \\ \tilde{B}_0 &= \begin{bmatrix} b_{0_1} \\ \vdots \\ b_{0_L} \end{bmatrix}_L = \begin{bmatrix} b_{\text{conc}_1} \\ \vdots \\ b_{\text{conc}_L} \end{bmatrix}_L = \text{constrained subvector of weights} \end{aligned}$$

The corresponding partition of the covariance matrix also occurs in REORDR, as follows:

$$\underline{M} = \begin{bmatrix} \begin{bmatrix} \underline{Y} \end{bmatrix}_{L \times L} & \begin{bmatrix} \underline{C}_0^+ \end{bmatrix}_{L \times (N-L)} \\ \begin{bmatrix} \underline{C}_0 \end{bmatrix}_{(N-L) \times L} & \begin{bmatrix} \underline{M}_c \end{bmatrix}_{(N-L) \times (N-L)} \end{bmatrix}_{N \times N}$$

where $\underline{M}_c$ is the "unconstrained" submatrix.

Note that only the constrained subvector of weights $\tilde{B}_0$ is actually being constrained to discrete values. The partitioning of $\tilde{W}_q$ into $\tilde{A}_0$ and $\tilde{A}_T$ is performed to be compatible in dimensionality with $\tilde{B}_0$. Similarly the partition of the $\underline{M}$ matrix is also done for this reason.

The maximum SIR for the partially constrained weight vector is calculated in subroutine SIMAX. The following intermediate quantities are used in the computer program to evaluate the equations derived in Appendix B.

$$\underline{\tilde{U}} = \underline{\tilde{M}}_c^{-1} \underline{\tilde{A}}_T$$

$$\underline{\tilde{V}} = \underline{\tilde{M}}_c^{-1} \underline{\tilde{C}}_0 \underline{\tilde{B}}_0$$

$$\text{NUMER} = \underline{\tilde{B}}_0^+ (\underline{\tilde{Y}} \underline{\tilde{B}}_0 - \underline{\tilde{C}}_0^+ \underline{\tilde{V}})$$

$$\text{DENOM} = \underline{\tilde{B}}_0^+ \underline{\tilde{A}}_0 - \underline{\tilde{V}}^+ \underline{\tilde{A}}_T$$

$$\alpha_1 = \frac{\text{NUMER}}{\text{DENOM}}$$

Finally, the maximum SIR is calculated using the next equation.

$$\psi_{\max} = \underline{\tilde{A}}_T^+ \underline{\tilde{U}} + (\text{DENOM})^*/\alpha_1$$

SIMAX then returns to the MAIN routine, where the value for $\psi_{\max}$ is compared to SIR_{clb} , the current lower bound for the SIR. If $\psi_{\max}$ is greater than or equal to SIR_{clb} , the node is not fathomed. As shown in Chapter 2, Section 2.3, only the first element in the unconstrained subvector of weights at level $L = k$ is constrained to the $(k+1)^{\text{st}}$ element in the constrained weight subvector at level $L = k+1$. Using the outputs from SIMAX, the first element in the optimum constrained weight vector can be obtained using the following formula:

$$b_{\text{opt}_1} = \alpha_1 u_1 - v_1$$

where u_1 is the first element in $\underline{\tilde{U}}$

v_1 is the first element in $\underline{\tilde{V}}$

The derivation of the equation for b_{opt_1} is given in Appendix B. The value for b_{opt_1} is used, in the branch and bound loop, as the input to the ANGLE subroutine. The output, α , from ANGLE constitutes the input to R1R2.

When the N^{th} level is reached (the bottom row of the tree) and all elements in the weight vector have been constrained to discrete values, the current lower bound for the SIR is calculated using a different formula from that in subroutine SIMAX. This is accomplished in a subroutine called NSIMAX, using the following familiar equation:

$$\psi_{max} = \frac{|\tilde{W}^+ \tilde{W}_q|^2}{\tilde{W}^+ \underline{M} \tilde{W}}$$

where

$\tilde{W} = B_{conc}$ = vector of constrained weights

$\tilde{W}_q$ = quiescent weight vector

$\underline{M}$ = noise and interference covariance matrix

After returning to the MAIN program, a comparison is made to determine if the current lower bound for the SIR should be updated to this value for ψ_{max} . If so, the corresponding weight vector $\tilde{W}_{clb}$ is set to B_{conc} .

A vertex is considered to be fathomed if it is at the bottom of the tree or when the maximum SIR is less than the current lower bound. The following processing occurs.

As described earlier, the values for NPASS are 1, 2, ..., MDIM = 2^{NBITS} . Therefore, at level L, where L is the index for NPASS, NPASS_L is incremented by 1. If some branches are yet to be processed (NPASS_L < MDIM+1) a check is made to determine if NPASS_L is equal to 2. If not, subroutine RHIGH is used to obtain the higher priority vectors $R_{\sim 3}, R_{\sim 4}, \dots, R_{\sim \text{MDIM}}$. It should be noted that the following implementation is used in RHIGH to save storage.

$$R_{\sim 3}, R_{\sim 5}, R_{\sim 7}, \dots, R_{\sim 2n+1} \rightarrow R_{\sim 1}$$

$$R_{\sim 2}, R_{\sim 4}, R_{\sim 6}, \dots, R_{\sim 2n} \rightarrow R_{\sim 2}$$

where n is an integer. The fathoming loop closes at the calculation of $B_{\sim \text{conc}}$ for the same level L.

If no branches are left to be processed (NPASS_L = MDIM+1), the logic will back up the tree to the previous level. This is manifested in the program by resetting NPASS at level L to 1, decrementing L by 1, and incrementing NML by 1. If L is not zero, the top of the tree has not yet been reached and NPASS is incremented at level L so that a search of the other branches at that level can continue. When L is zero, the top of the tree has been reached and the value for SIR_{clb} and $W_{\sim \text{clb}}$ are the solutions for the maximum SIR and the corresponding discrete optimum weight vector, respectively. These values are output at the printer and the antenna patterns for $W_{\sim \text{clb}}, W_{\sim \text{opt}}$ (the continuous optimum weight vector), and $W_{\sim \text{q}}$ (the quiescent weight vector) are plotted.

3.6 ARFACT and APLOT

The two routines ARFACT and APLOT compute the array factor and plot the resulting antenna pattern, respectively. Since they are used routinely for antenna applications, the programming details will not be presented here.

3.7 Trench-Zohar Complex Toeplitz Matrix Inversion Subroutine

A subroutine called ZORN* was used to perform all of the complex Toeplitz matrix inversions in this program. It takes $\approx N^2$ operations to invert an N-dimensional complex Toeplitz matrix as compared to the $\approx N^3$ operations required by the Amdahl SCILIB routine.

3.8 Program Listing

A complete listing of the program appears in Appendix C.

* ZORN was adapted from the real Toeplitz Matrix Inversion Routine written by Captain Alan Paige, U.S. Air Force.

Chapter 4

RESULTS

4.1 Introduction

The branch and bound algorithm was tested with four and eight element arrays, to determine correctness and speed of convergence. Numerical results were obtained to determine quantitative jammer cancellation performance. Some of the cases converged very slowly and the runs were deliberately interrupted. The intermediate results obtained were, in many cases, unsatisfactory. These problems are further discussed at the end of the chapter.

An untapered array illumination is assumed for the unadapted pattern since sidelobe control in the unadapted pattern is not essential to demonstrate the adaptation capacity of the ESA.

4.2 Results for Four Element Arrays

The branch and bound method has been tested for correctness by comparing its results with those obtained using an exhaustive search through all possible phase settings. A four element linear array with element spacing of $\lambda/2$ and three bit phase shifters has been used for this purpose. The results of the tests showed that the identical discrete optimal weight vectors were output from the branch and bound and the exhaustive search.

The increased speed of convergence due to fathoming did not shorten the runs for these smaller cases, since the overhead costs were significantly higher for the branch and bound program.

Results pertaining to the following special cases are shown.

Signal angle is always 40° from broadside. In each of the three configurations considered there is only one interfering signal 20 dB above array element thermal noise arriving at 20° (main beam) or -10° (first sidelobe) or -50° (second sidelobe) from broadside.

Results have been obtained for arrays with 2, 3, and 4 bit phase shifters. Antenna patterns are given in Figures 4.1, 4.2 and 4.3 for all jammer configurations, for the array with 3 bit phase shifters. One pattern for the array with 4 bit phase shifters is shown in Figure 4.4. In each of Figures 4.1, 4.2 and 4.3, there are three patterns: (A) unadapted array, (B) adapted discrete or ESA (discrete adjustable phases, fixed gains) and (C) adapted continuous (continuously varying phases and magnitudes). The positions of signal and jammer are indicated by arrows. We note invariably that both the ESA and the adaptive array are reducing the jamming severity by placing nulls in the vicinity of the jammer. As expected, the performance of (B) is worse than that of (C) as evidenced by less accurate placement of the jammer nulls, higher sidelobes elsewhere and more distortions in the main beam. Nevertheless, the performance of (B) is good enough to be of practical importance. In Figure 4.4, the same jamming case is covered as in Figure 4.1; the patterns displayed are type (C), type (B) 3 bit phase shifters and type (B) 4 bit phase shifters. Note the improved performance achieved with 4 bits over 3 bits. The array phase settings for the 4 bit phase shifters were obtained as an intermediate result in an incomplete branch and bound run. More quantitative results of performance improvement with an increase in the number of bits is given in Table 4.1. One would

FIGURE 4.1 ADAPTING PATTERNS WITH JAMMER IN MAIN BEAM, ESA WITH 3 BIT PHASE SHIFTERS

FIGURE 4.2 ADAPTING PATTERNS WITH JAMMER IN FIRST SIDELobe, ESA WITH 3 BIT

FIGURE 4.3 ANTENNA PATTERNS WITH JAMMER IN SECOND SIDELobe, ESA WITH 3 BIT PHASE SHIFTERS

FIGURE 4.4 COMPARISON OF ADAPTING PATTERNS WITH JAMMER IN MAIN BEAM,
ESA WITH 3 BIT PHASE SHIFTERS vs. 4 BIT PHASE SHIFTERS.

TABLE 4.1 PERFORMANCE IMPROVEMENT IN SIR (DB)
WITH INCREASING NUMBER OF PHASE SHIFTER BITS

θ_j	2 BITS ESA	3 BITS ESA	4 BITS ESA	ADAPTIVE ARRAY
20°	-5.71	-1.24	3.26	4.65
-10°	-5.78	1.83	3.81	5.77
-50°	4.63	4.63	4.63	5.89

expect the ESA performance (in the signal to interference plus noise sense) to improve as the number of bits increases without reaching the performance of the adaptive array. The behavior displayed by the data in Table 4.1 indicates that there is no definite number of bits where a performance plateau is reached. The ability of the ESA to combat interference is quantitatively described in Tables 4.2, 4.3 and 4.4. In Table 4.2, the signal to interference plus noise ratio is compared for a number of situations. It degrades by 10 to 20 dB for the fixed array as the environment changes from benign to the presence of one jammer. Allowing discrete phase-only adaptation recovers 8 to 17 of the formerly lost decibels. Finally, the additional margin required to meet the performance of the adaptive array is of the order of only 1.5 to 6 decibels. In Tables 4.3 and 4.4, information is given with regard to the ability of the ESA and the adaptive array to reject interference while still accepting the desired signal without appreciable degradation. The adaptive array reduces the jammer contribution to the point where operation is thermal noise limited with the degradation in SNR resulting primarily from the slightly reduced desirable signal reception concomitant with adaptation. The ESA helps reduce the interference by about 20 dB which constitutes a very valuable contribution, but as a net effect, the residual degradation in SNR comes about both from lesser reception of the desired signal as well as jammer noise.

4.3 Results for Eight Element Arrays

Many cases were tested using eight element arrays with four bit phase shifters. However, they could not reasonably be verified using

TABLE 4.2 IMPROVEMENT IN SIR (DB) DUE TO ADAPTATION
IN THE PRESENCE OF ONE JAMMER

θ_j	NO JAMMER	ONE JAMMER, SIGNAL AT 40°		
	UNADAPTING ARRAY	UNADAPTING ARRAY	ADAPTING PHASES ONLY, 3 BIT PHASE SHIFTERS	ADAPTING PHASES AND GAINS, CONTINUOUS
20°	6.02	-14.38	-1.24	4.65
-10°	6.02	- 7.74	1.83	5.77
-50°	6.02	- 4.93	4.63	5.89

INCOMPLETE SEARCH
FOR 4 BIT PHASE
SHIFTERS

20°	6.02	-14.38	3.26	4.65
------------	------	--------	------	------

TABLE 4.3 JAMMER REJECTION GIVEN AS A DIFFERENCE
BETWEEN ADAPTED AND QUIESCENT PATTERNS (DB)

ONE JAMMER, SIGNAL AT 40°		
θ_j	ADAPTING, PHASES ONLY, 3 BIT PHASE SHIFTERS	ADAPTING, PHASES AND GAINS, CONTINUOUS
20°	19.87	39.20
-10°	22.34	52.09
-50°	18.36	52.35

INCOMPLETE SEARCH FOR 4 BIT
PHASE SHIFTERS

20°	30.48	39.20
-----	-------	-------

TABLE 4.4 SIGNAL DEGRADATION GIVEN AS A DIFFERENCE
BETWEEN ADAPTED AND QUIESCENT PATTERNS (DB)

	← ONE JAMMER, SIGNAL AT 40° →	
θ_j	ADAPTING, PHASES ONLY, 3 BIT PHASE SHIFTERS	ADAPTING, PHASES AND GAINS, CONTINUOUS
20°	4.00	2.75
-10°	3.65	0.51
-50°	0.72	0.25

INCOMPLETE SEARCH FOR 4
BIT PHASE SHIFTERS

20°	2.36	2.75
-----	------	------

an exhaustive search. Noting that there are L^N points to be considered (see Section 1.5), where N is the number of array elements and $N=8$, $b=4$, the search space consists of 16^8 points and for $N^2=64$ multiplications and additions per point, the computation would last twenty-four days on an AMDAHL 470/V7. Therefore, program validation consisted of running many cases while varying the jammer position and power. The antenna patterns were used to obtain quantitative results on the jammer cancellation performance. Convergence time was also measured during this testing procedure. It is important to note that none of these cases completely finished executing. The results were obtained by using the program in an interactive mode in which the most recent value for the SIR current lower bound (CLB) was saved and used to initialize the input for the next run. In this manner, acceptable adaptation performance was achieved with the final value for the CLB very close to the maximum SIR for continuously varying weights (adapted continuous).

A representative 8 element, 4 bit phase shifter case is shown in Figure 4.5. The adapting ESA pattern was obtained in an incomplete branch and bound run. Note, however, that the CLB (PSICLB = 7.65) is very close to the fully adaptive array optimum SIR (SIRMAX = 7.87). Both the ESA and the adaptive array are reducing the jamming severity by placing nulls in the vicinity of the jammer.

4.4 Problems with Algorithm

When running many four element cases with more than four bits in the phase shifters, convergence problems occurred. For five bits the program converged when used in the interactive fashion described in

FIGURE 4.5 PATTERNS FOR EIGHT
ELEMENT LINEAR ARRAY

the previous section. Results for the six element cases with three bits were also obtained in this manner. However, when 8 element arrays with 4 bit phase shifters were used, none of the cases completely finished executing. Using the program interactively led to results that were very close to the adapted continuous maximum SIR.

In order to enable the program to start converging, the pre-ESA processing used for the smaller cases to select the current lower bound (CLB) for the SIR and the corresponding optimum weight vector was suspended. This was done because no advantage was perceived. Final judgment on this point, however, is reserved for later. Instead, the CLB for the SIR was initialized externally to a value smaller than the adapted continuous maximum SIR. These runs had varying degrees of success. Sometimes the program would update the CLB several times before reaching a plateau where the same value for the CLB would repeat. This type of behavior resulted in the antenna pattern shown in Figure 4.5. On the other hand, there were many cases where the input value for the CLB was not updated after several minutes. In these cases, another guess was made for the initial value, and the program was rerun. The algorithm appears to be extremely sensitive to the initial value. The explanation for this behavior is that the SIR surface in $\mathbb{W} \in \mathbb{C}^N$ is rather flat; therefore a long computer run is required for convergence if the initial value is not close enough, from below, to the discrete solution. This amounts, practically, to guessing the maximum SIR for the discrete weights. The program is still valuable since it produces the required weights and provides performance verification. Results for larger cases ($N=16$, $b=4$ and $N=27$, $b=5$)

were unsatisfactory; the SIR_{clb} at the end of incomplete runs was small compared to the adapted continuous array performance.

More work will be done to improve the program so that these larger cases can be studied. Some specific ideas will be discussed in the next chapter.

Chapter 5

CONCLUSIONS AND FUTURE WORK

A method has been described for obtaining optimal phase settings for jammer cancellation for an electronically scanning array whose only variable parameters are element phases which vary in discrete steps. The method has been tested by exploring a few examples. The jammer cancellation results in these cases have been encouraging. Results have been obtained for $N=8$ and $b=4$ which could not reasonably be obtained with an exhaustive search. Even though the amount of computation is reduced by orders of magnitude, running the program is still costly.

As a minimum, the approach described provides upper bounds for adaptation performance for electronically scanning arrays that are controlled with digital phase shifters, for small cases.

As was mentioned in the previous chapter, the results of the branch and bound were not always successful for larger cases. Therefore, the following programming improvements will be made to increase program efficiency and thereby reduce cost and generate more results:

- 1) Removing double precision on all computations, after verifying that the accuracy of the results will not be impaired.
- 2) Changing the output from the Toeplitz matrix inversion routine so that the scaling will no longer be done in the matrix formation where N^2 multiplications must occur. Instead, the matrix

will be multiplied by a vector and the resulting vector will be scaled. This will result in only N multiplications.

- 3) Removing debugging subroutine calls and executing the program on the optimized H-extended compiler, which will greatly increase the program's run-time efficiency.
- 4) Eventually running the program on a dedicated processor with low overhead cost.

Additional conceptual improvements can be made to the branch and bound method itself. By constraining several weights at one time, the branch and bound tree will shrink vertically while expanding laterally. Constraining more than one weight as the algorithm proceeds from one level to another down the tree should result in more fathoming. This would provide computational savings. At one extreme, constraining all of the weights in one step is equivalent to solving the problem with the exhaustive search method, which is not practical for even moderately large N and b . Therefore, it is argued that the number of weights that are constrained at each step offers a computational effort tradeoff. Hopefully, the minimum execution time, as a function of the number of weights constrained at each step, is not a flat function.

The method described previously whereby more than one weight is constrained at each level is a rigorous branch and bound technique. It is possible to approximate the branch and bound method in a way that

would lead, with high probability, to valid solutions with shorter execution times. One such example is described with the aid of Figure 5.1. The branch and bound is rigorously applied to subtree I. The vertex where the $\text{MAX}_{\substack{\tilde{w}_u^l \in C^{N-l}}} \text{SIR}(\tilde{w}_c^l, \tilde{w}_u^l)$ occurs is chosen as the top vertex of tree II. This procedure is repeated until the bottom of the tree is reached. Clearly, the method is imprecise since the trajectory leading to the true solution at the bottom of the tree may be outside the search path thus outlined. Yet, intuition indicates that much of the time a reasonable solution would be obtained with a lesser computational effort.

Another method to test the branch and bound algorithm versus an exhaustive search will be implemented for larger cases. Both programs will be allowed to run for a fixed amount of time. At the end of this period, the resulting current lower bound for the SIR from the two methods will be compared. If the branch and bound value is larger for several cases, the algorithm will be shown to be superior to the exhaustive search, in addition to verifying the branch and bound algorithm without having the program run to completion.

It is expected that using the improvements discussed in this chapter will enable the program to complete much larger cases and to verify the results. Additional development and testing is required to verify the method in terms of correctness and practicality for use in real time systems.

Figure 5.1 Approximate Branch and Bound Search

References

1. B. Widrow, P. Mantey, L. Griffith and B. Goode, "Adaptive Antenna Systems," Proc. IEEE, Vol. 55, pp. 2143-2159, December 1967.
2. S. P. Applebaum, "Adaptive Arrays," IEEE Trans. Antennas Propagat. Vol. AP-24, pp. 583-598, September 1976.
3. A. I. Cohen, "An Algorithm for Designing Burst Waveforms with Quantized Transmitter Weights," IEEE Trans. Aerospace and Electronic Systems, Vol. AES-11, pp. 56-75, January 1975.
4. W. F. Trench, "An Algorithm for the Inversion of Finite Toeplitz Matrices," J. Soc. Indust. Appl. Math., Vol. 12, No. 3, pp. 515-522, September 1964.
5. S. Zohar, "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench," J. Ass. Comput. Mach., Vol. 16, pp. 592-601, October 1969.
6. L. J. Spafford, "Optimum Radar Signal Processing in Clutter," IEEE Trans. Information Theory, Vol. IT-14, pp. 734-743, September 1968.
7. D. K. Cheng and F. I. Tseng, "Maximization of Directive Gain for Circular and Elliptical Arrays," Proc. IEEE, Vol. 114, No. 5, pp. 589-594, May 1967.
8. F. R. Gantmacher, Matrix Theory, Chelsea Publishing Co., New York, N.Y., Vol. 1, p. 322, 1960.
9. R. F. Harrington, Field Computation by Moment Methods, The Macmillan Co., New York, N.Y., pp. 191-193, 1968.

Appendix A

MAXIMIZATION OF THE SIGNAL TO INTERFERENCE RATIO

The results of Cheng and Tseng [7] in relation to the maximization of the ratio of two quadratic forms are presented below. Their proof establishes that

$$\underline{\tilde{W}} = \underline{M}^{-1} \underline{\tilde{W}}_q \quad (A1)$$

maximizes the signal to interference ratio equation A2.

$$SIR = \frac{|\underline{\tilde{W}}^t \underline{\tilde{T}}|^2}{\underline{\tilde{W}}^t \underline{M} \underline{\tilde{W}}} \quad (A2)$$

Cheng and Tseng's results are applicable subject to $\underline{M}$ being Hermitian and positive definite and the numerator of Equation A2 being a quadratic form of an Hermitian matrix. These features are established next.

Expanding the numerator, Equation A3 is obtained.

$$|\underline{\tilde{W}}^t \underline{\tilde{T}}|^2 = \left(\underline{\tilde{W}}^t \underline{\tilde{T}} \right)^* \left(\underline{\tilde{W}}^t \underline{\tilde{T}} \right)^t = \underline{\tilde{W}}^t \begin{bmatrix} \underline{\tilde{T}}^* & \underline{\tilde{T}}^t \end{bmatrix} \underline{\tilde{W}} \quad (A3)$$

We will now use the following substitution

$$\underline{\tilde{T}}^* = \underline{\tilde{W}}_q \quad (A4)$$

where $\underline{\tilde{W}}_q$ is the vector of unadapted weights representing the optimum compromise between such performance factors as resolution, gain, and

low sidelobes. Another substitution is used in Equation A5.

$$\underline{A} = \begin{bmatrix} W_{\sim q} & W_{\sim q}^{\dagger} \end{bmatrix} \quad (\text{A5})$$

It should be noted that this is an outer product, or dyadic form which is an Hermitian matrix as shown in Equation A6.

$$\underline{A}^{\dagger} = \begin{bmatrix} W_{\sim q} & W_{\sim q}^{\dagger} \end{bmatrix}^{\dagger} = \begin{bmatrix} W_{\sim q} & W_{\sim q}^{\dagger} \end{bmatrix} = \underline{A} \quad (\text{A6})$$

If the noise components are uncorrelated, the noise and interference covariance matrix $\underline{M}$ will be diagonal. In general, $\underline{M}$ may have nonzero entries in any position. It is denoted by

$$\underline{M} = \begin{bmatrix} m_{ij} \end{bmatrix}, \quad i=1,k, \quad j=1,k \quad (\text{A7})$$

where k is the number of receiver channels. The covariance of the noise in the i^{th} and j^{th} channel is given by

$$m_{ij} = E\{n_i^* n_j\} \quad (\text{A8})$$

$$m_{ji} = E\{n_j^* n_i\} = E\{n_i n_j^*\} = m_{ij}^* \quad (\text{A9})$$

This results in Equations A10 and A11.

$$\underline{M}^t = \underline{M}^* \quad (\text{A10})$$

$$\underline{M}^{\dagger} = \underline{M} \quad (\text{A11})$$

Therefore, matrix $\underline{M}$ is Hermitian. It is also positive definite since

$\underline{M}$ is the covariance matrix for a stationary random process. P_n in Equation A12 is greater than 0 whenever $\underline{W} \neq 0$.

$$P_n = \underline{W}^\dagger \underline{M} \underline{W} \quad (\text{A12})$$

Therefore, Equation A2 can be written as

$$\text{SIR} = \frac{\underline{W}^\dagger \underline{A} \underline{W}}{\underline{W}^\dagger \underline{M} \underline{W}} \quad (\text{A13})$$

The proof of Cheng and Tseng makes use of the following theorem from Gantmacher [8].

THEOREM:

Given $n \times n$ Hermitian matrices $\underline{C}$ and $\underline{D}$ with $\underline{D}$ positive definite the equation

$$|\underline{C} - \lambda \underline{D}| = 0$$

has in general n distinct values for λ (eigenvalues).

Then:

- 1) The largest characteristic value λ_M is the maximum of the ratio of the quadratic forms

$$\frac{\underline{W}^\dagger \underline{C} \underline{W}}{\underline{W}^\dagger \underline{D} \underline{W}};$$

$$\lambda_M = \max \frac{\underline{W}^\dagger \underline{C} \underline{W}}{\underline{W}^\dagger \underline{D} \underline{W}}$$

and this maximum is assumed only for the eigenvector $\underline{W}$ associated with

the characteristic value λ_M , which satisfies the following homogeneous equation:

$$\underline{C} \underline{W} = \lambda_M \underline{D} \underline{W} \quad (\text{A14})$$

The following properties enable the maximum obtainable SIR for a receiving array pointed in a specified direction to be determined, as well as the required set of weights to achieve it.

- a) The largest eigenvalue λ_M of the linear combination $\underline{A} - \lambda \underline{M}$ is its only non-zero value and is given by

$$\lambda_M = \underline{W}_q \underline{M}^{-1} \underline{W}_q \quad (\text{A15})$$

- b) The eigenvector corresponding to λ_M is

$$\underline{W}_M = \underline{M}^{-1} \underline{W}_q \quad (\text{A16})$$

The following proof is used to establish the above properties.

Proof that $\underline{C} - \lambda \underline{D}$ has only one non-zero eigenvalue λ_M

In order to show that the linear combination of matrices $\underline{C} - \lambda \underline{D}$ has only one non-zero eigenvalue λ_M , the following approach is used.

THEOREM:

If $\underline{C}$ and $\underline{D}$ are, respectively, $m \times n$ and $n \times m$ matrices, then

$$(-1)^m \lambda^n \det (\underline{C} \underline{D} - \lambda \underline{I}_m) = (-1)^n \lambda^m \det (\underline{D} \underline{C} - \lambda \underline{I}_n) \quad (\text{A17})$$

where $\underline{I}_m$ denotes the $m \times m$ identity matrix.

Proof:

$$\begin{bmatrix} \underline{C} \underline{D} - \lambda \underline{I}_m & \underline{C} \\ 0 & \lambda \underline{I}_n \end{bmatrix} \begin{bmatrix} -\underline{I}_m & 0 \\ \underline{D} & \underline{I}_n \end{bmatrix} = \begin{bmatrix} \lambda \underline{I}_m & \underline{C} \\ \lambda \underline{D} & \lambda \underline{I}_n \end{bmatrix} \quad (\text{A18})$$

$$\begin{bmatrix} \underline{I}_m & 0 \\ \underline{D} & -\underline{I}_n \end{bmatrix} \begin{bmatrix} \lambda \underline{I}_m & \underline{C} \\ 0 & \underline{D} \underline{C} - \lambda \underline{I}_n \end{bmatrix} = \begin{bmatrix} \lambda \underline{I}_m & \underline{C} \\ \lambda \underline{D} & \lambda \underline{I}_n \end{bmatrix} \quad (\text{A19})$$

Since the right sides of Equations A18 and A19 are the same, the determinants of the left sides are the same.

Therefore, Equation A20 follows:

$$\det \begin{bmatrix} \underline{C} \underline{D} - \lambda \underline{I}_m & \underline{C} \\ 0 & \lambda \underline{I}_n \end{bmatrix} \det \begin{bmatrix} -\underline{I}_m & 0 \\ \underline{D} & \underline{I}_n \end{bmatrix} = \det \begin{bmatrix} \underline{I}_m & 0 \\ \underline{D} & -\underline{I}_n \end{bmatrix} \det \begin{bmatrix} \lambda \underline{I}_m & \underline{C} \\ 0 & \underline{D} \underline{C} - \lambda \underline{I}_n \end{bmatrix} \quad (\text{A20})$$

Noting that the determinants of the matrices in Equation A20 have the following properties

$$\det \begin{bmatrix} -\underline{I}_m & 0 \\ \underline{D} & \underline{I}_n \end{bmatrix} = (-1)^m \quad (\text{A21})$$

$$\det \begin{bmatrix} \underline{I}_m & 0 \\ \underline{D} & -\underline{I}_n \end{bmatrix} = (-1)^n \quad (\text{A22})$$

since both matrices are lower triangular and the determinants are

equal to the product of the elements on the diagonal and

$$\det \begin{bmatrix} \underline{C} \underline{D} & - & \lambda \underline{I}_m & \underline{C} \\ & 0 & & \lambda \underline{I}_n \end{bmatrix} = \lambda^n \det (\underline{C} \underline{D} - \lambda \underline{I}_m) \quad (\text{A23})$$

$$\det \begin{bmatrix} \lambda \underline{I}_m & \underline{C} \\ 0 & \underline{D} \underline{C} - \lambda \underline{I}_n \end{bmatrix} = \lambda^m \det (\underline{D} \underline{C} - \lambda \underline{I}_n) \quad (\text{A24})$$

since both matrices are upper triangular and their determinants are equal to the product of the determinants of the (matrix) elements on the diagonal, we obtain Equation A17:

$$(-1)^m \lambda^n \det (\underline{C} \underline{D} - \lambda \underline{I}_m) = (-1)^n \lambda^m \det (\underline{D} \underline{C} - \lambda \underline{I}_n)$$

Q.E.D.

For the purpose of establishing property a) and Equation A15 the eigenvalues of the linear combination $\underline{A} - \lambda \underline{M}$ are needed. These are the roots of the characteristic equation

$$\det (\underline{A} - \lambda \underline{M}) = 0 \quad (\text{A25})$$

Since $\underline{M}$ is Hermitian and positive definite it has an inverse and Equation A25 can be written as

$$\det (\underline{M}^{-1}) \det (\underline{A} - \lambda \underline{M}) = \det (\underline{M}^{-1} [\underline{A} - \lambda \underline{M}]) = 0 \quad (\text{A26})$$

leading to Equation A27.

$$\det (\underline{M}^{-1} \underline{A} - \lambda \underline{I}_n) = 0 \quad (\text{A27})$$

Substituting $\underline{C} = \underline{W}_q^\dagger$ and $\underline{D} = \underline{M}^{-1} \underline{W}_q$ into Equation A17 results in

$$(-1)^m \lambda^n \det (\underline{W}_q^\dagger \underline{M}^{-1} \underline{W}_q - \lambda \underline{I}_m) = (-1)^n \lambda^m \det (\underline{M}^{-1} \underline{W}_q \underline{W}_q^\dagger - \lambda \underline{I}_n) \quad (\text{A28})$$

With $m = 1$ and $n = N$ we obtain

$$(-1)^1 \lambda^N \det (\underline{W}_q^\dagger \underline{M}^{-1} \underline{W}_q - \lambda \underline{I}_1) = (-1)^N \lambda^1 \det (\underline{M}^{-1} \underline{W}_q \underline{W}_q^\dagger - \lambda \underline{I}_N) \quad (\text{A29})$$

$$\text{or } (-1)^{(1-N)} \lambda^{(N-1)} \det (\underline{W}_q^\dagger \underline{M}^{-1} \underline{W}_q - \lambda \underline{I}_1) = \det (\underline{M}^{-1} \underline{W}_q \underline{W}_q^\dagger - \lambda \underline{I}_N) \quad (\text{A30})$$

Letting $\underline{A} = \underline{W}_q \underline{W}_q^\dagger$ and using Equation A27 gives the result

$$\lambda^{(N-1)} \det (\underline{W}_q^\dagger \underline{M}^{-1} \underline{W}_q - \lambda \underline{I}_1) = 0 \quad (\text{A31})$$

Hence, $N-1$ of the N eigenvalues of $\underline{A} - \lambda \underline{M}$ are zero and the only non-zero eigenvalue is

$$\lambda_M = \underline{W}_q^\dagger \underline{M}^{-1} \underline{W}_q$$

thus obtaining Equation A15.

Let an $N \times 1$ column vector be equal to

$$\underline{\tilde{W}} = \underline{M}^{-1} \underline{W}_q \quad (\text{A32})$$

which can be written as

$$\underline{\tilde{W}}_q = \underline{M} \underline{\tilde{W}} \quad (\text{A33})$$

It follows that

$$\tilde{W}_q^\dagger = (\underline{M} \tilde{W})^\dagger = \tilde{W}^\dagger \underline{M}^\dagger = \tilde{W}^\dagger \underline{M} \quad (\text{A34})$$

since $\underline{M}$ is Hermitian.

Substituting Equations A33 and A34 into Equation A15, the only non-zero eigenvalue is

$$\lambda_M = (\tilde{W}_q^\dagger \underline{M}) \underline{M}^{-1} (\underline{M} \tilde{W}) = \tilde{W}_q^\dagger \underline{M} (\underline{M}^{-1} \underline{M}) \tilde{W} = \tilde{W}_q^\dagger \underline{M} \tilde{W} > 0 \quad (\text{A35})$$

since $\underline{M}$ is Hermitian and positive definite. Thus, λ_M in Equation A15 is the largest eigenvalue.

We proceed next to establish property b) by proving that the eigenvector of $\underline{A} - \lambda_M \underline{M}$ corresponding to its largest eigenvalue is equal to $\underline{M}^{-1} \tilde{W}_q$. Using $\underline{C} = \underline{A}$ and $\underline{D} = \underline{M}$ in Equation A14 we obtain

$$\underline{A} \tilde{W} = \lambda_M \underline{M} \tilde{W} \quad (\text{A36})$$

which can be rewritten as

$$(\tilde{W}_q \tilde{W}_q^\dagger) (\underline{M}^{-1} \tilde{W}_q) = \lambda_M \underline{M} (\underline{M}^{-1} \tilde{W}_q) \quad (\text{A37})$$

using Equations A5 and A32.

From Equation A15

$$\lambda_M = \tilde{W}_q^\dagger \underline{M}^{-1} \tilde{W}_q$$

Therefore, Equation A37 can be written as

$$\underline{W}_{\sim q} \lambda_M = \lambda_M \underline{W}_{\sim q} \quad (\text{A38})$$

Hence, Equation A32 is the optimum eigenvector $\underline{W}_{\sim \text{opt}}$. Therefore,

$$\underline{W}_{\sim \text{opt}} = k \underline{M}^{-1} \underline{W}_{\sim q}$$

where k is an arbitrary scalar which we set equal to unity without affecting the results.

Appendix B

MAXIMUM SIR FOR A PARTIALLY CONSTRAINED WEIGHT VECTOR

The SIR for the fully adaptive array is given by

$$SIR = \frac{|\tilde{W}^t \tilde{W}_q^*|^2}{\tilde{W}^t \underline{M} \tilde{W}} = \frac{\tilde{W}^t [\tilde{W} \tilde{W}_q^*] \tilde{W}}{\tilde{W}^t \underline{M} \tilde{W}} \quad (B1)$$

where the optimum weight vector which maximizes the SIR is

$$\tilde{W} = \underline{M}^{-1} \tilde{W}_q$$

Consider a partially constrained weight vector $\tilde{W}$, which is denoted by $\tilde{B}$. The $\tilde{B}$ vector is partitioned into an unconstrained and constrained subvector as follows:

$$\tilde{B} = \begin{bmatrix} \tilde{b} \\ \tilde{B}_0 \end{bmatrix} \quad (B2)$$

where $\tilde{b}$ = unconstrained subvector

$\tilde{B}_0$ = constrained subvector

The unconstrained subvector $\tilde{b}$ that maximizes the SIR is derived next. For this purpose, the vector $\tilde{W}_q$ and matrix $\underline{M}$ are partitioned to be compatible with Equation B2 as follows:

$$\tilde{W}_q = \begin{bmatrix} \tilde{a} \\ \tilde{A}_0 \end{bmatrix} \quad (B3)$$

$$\underline{M} = \begin{bmatrix} \underline{M}_c & \underline{C}_0 \\ \underline{C}_0^\dagger & \underline{Y} \end{bmatrix} \quad (B4)$$

Substituting Equations B2, B3 and B4 into B1 yields

$$\begin{aligned} \text{SIR} &= \frac{[b^\dagger \quad B_0^\dagger] \begin{bmatrix} \underline{a} & \underline{a}^\dagger & \underline{aA}_0^\dagger \\ \underline{A}_0 \underline{a}^\dagger & \underline{A}_0 \underline{A}_0^\dagger \end{bmatrix} \begin{bmatrix} \underline{b} \\ \underline{B}_0 \end{bmatrix}}{[b^\dagger \quad B_0^\dagger] \begin{bmatrix} \underline{M}_c & \underline{C}_0 \\ \underline{C}_0^\dagger & \underline{Y} \end{bmatrix} \begin{bmatrix} \underline{b} \\ \underline{B}_0 \end{bmatrix}} \\ &= \frac{[b^\dagger \quad B_0^\dagger] \begin{bmatrix} \underline{aa}^\dagger \underline{b} + \underline{aA}_0^\dagger \underline{B}_0 \\ \underline{A}_0 \underline{a}^\dagger \underline{b} + \underline{A}_0 \underline{A}_0^\dagger \underline{B}_0 \end{bmatrix}}{[b^\dagger \quad B_0^\dagger] \begin{bmatrix} \underline{M}_c \underline{b} + \underline{C}_0 \underline{B}_0 \\ \underline{C}_0^\dagger \underline{b} + \underline{Y} \underline{B}_0 \end{bmatrix}} \\ &= \frac{\underline{b}^\dagger [\underline{aa}^\dagger \underline{b} + \underline{aA}_0^\dagger \underline{B}_0] + [\underline{B}_0^\dagger \underline{A}_0 \underline{a}^\dagger \underline{b} + \underline{B}_0^\dagger \underline{A}_0 \underline{A}_0^\dagger \underline{B}_0]}{\underline{b}^\dagger [\underline{M}_c \underline{b} + \underline{C}_0 \underline{B}_0] + [\underline{B}_0^\dagger \underline{C}_0^\dagger \underline{b} + \underline{B}_0^\dagger \underline{Y} \underline{B}_0]} \\ &= \frac{\underline{b}^\dagger \underline{\eta} + \Theta}{\underline{b}^\dagger \underline{\rho} + \gamma} \end{aligned} \quad (B5)$$

where $\underline{\eta}$, Θ , $\underline{\rho}$ and γ are shorthand notations for the lengthy expressions in the previous step. The equation can be rewritten as

$$\text{SIR} = \frac{\sum_i \underline{b}_i^* \underline{\eta}_i + \Theta}{\sum_j \underline{b}_j^* \underline{\rho}_j + \gamma} = \frac{N}{D} \quad (B6)$$

η , θ , ρ , and γ are all functions of $\underline{b}$. SIR is a real function of $\underline{b}$. Let the dimension of $\underline{b}$ be k ; then SIR is a function of the $2k$ real variables that constitute the real and imaginary parts of the components of $\underline{b}$. To maximize SIR, it is necessary to take $2k$ partial derivatives with respect to the real and imaginary parts of the components of $\underline{b}$ and set them equal to 0. Consider

$$\frac{\partial(\text{SIR})}{\partial[\text{Real}(\underline{b}_r)]} = 0 \quad \text{and} \quad \frac{\partial(\text{SIR})}{\partial[\text{Imag}(\underline{b}_r)]} = 0 \quad (\text{B7})$$

Equation B7 can be shown in [9] to be equivalent to

$$\frac{\partial(\text{SIR})}{\partial \underline{b}_r} = 0 \quad \text{and} \quad \frac{\partial(\text{SIR})}{\partial \underline{b}_r^*} = 0 \quad (\text{B8})$$

where the r^{th} component of $\underline{b}$, $\underline{b}_r$, and its complex conjugate, $\underline{b}_r^*$, are treated, for the purpose of taking the partial derivatives, as if they were real and independent variables. Taking the partial derivatives of Equation B6 yields

$$\frac{\partial(\text{SIR})}{\partial \underline{b}_r^*} = \frac{\eta_r D - \rho_r N}{D} = 0$$

which gives $\eta_r = \rho_r \frac{N}{D} = \rho_r (\text{SIR})$

or, in vector notation,

$$\underline{\eta} = (\text{SIR}) \underline{\rho} \quad (\text{B9})$$

Substituting for $\underline{\eta}$ and $\underline{\rho}$:

$$[\underline{a}^\dagger \underline{b} + \underline{A}_0^\dagger \underline{B}_0] \underline{a} = (\text{SIR}) [\underline{M}_c \underline{b} + \underline{C}_0 \underline{B}_0] \quad (\text{B10})$$

The unknown in Equation B10 is the vector $\tilde{b}$. Rearranging:

$$\frac{\underline{M}}{\underline{c}} \tilde{b} = \frac{[\tilde{a}^\dagger \tilde{b} + \underline{A}_0^\dagger \underline{B}_0]}{(\text{SIR})} \tilde{a} - \frac{\underline{C}_0}{\underline{c}} \underline{B}_0 \quad (\text{B11})$$

The expression

$$\alpha = \frac{[\tilde{a}^\dagger \tilde{b} + \underline{A}_0^\dagger \underline{B}_0]}{(\text{SIR})} \quad (\text{B12})$$

is an unknown constant. Substituting Equation B12 into Equation B11 we obtain:

$$\tilde{b} = \frac{\underline{M}}{\underline{c}}^{-1} (\alpha \tilde{a} - \frac{\underline{C}_0}{\underline{c}} \underline{B}_0) = \alpha \underline{M}^{-1} \tilde{a} - \underline{g} \quad (\text{B13})$$

where $\underline{g} = \frac{\underline{M}}{\underline{c}}^{-1} \underline{C}_0 \underline{B}_0$.

Substituting Equation B13 into Equation B5 yields

$$\text{SIR} = \frac{[\alpha^* \tilde{a}^\dagger \underline{M}^{-1} - \underline{g}^\dagger, \underline{B}_0^\dagger] \begin{bmatrix} \tilde{a} & \tilde{a}^\dagger & \tilde{a} & \underline{A}_0^\dagger \\ \underline{A}_0 & \tilde{a}^\dagger & \underline{A}_0 & \underline{A}_0^\dagger \end{bmatrix} \begin{bmatrix} \alpha \underline{M}^{-1} \tilde{a} - \underline{g} \\ \underline{B}_0 \end{bmatrix}}{[\alpha^* \tilde{a}^\dagger \underline{M}^{-1} - \underline{g}^\dagger, \underline{B}_0^\dagger] \begin{bmatrix} \underline{M} & \underline{C}_0 \\ \underline{C}_0^\dagger & \underline{Y} \end{bmatrix} \begin{bmatrix} \alpha \underline{M}^{-1} \tilde{a} - \underline{g} \\ \underline{B}_0 \end{bmatrix}} \quad (\text{B14})$$

Thus, the SIR is a function of only one complex scalar, α . Maximizing SIR with respect to α yields the value of α which, when substituted in Equation B13, gives the optimum $\tilde{b}$, denoted in Chapter 3 by $\tilde{b}_{\text{opt}}$. When the same value for α is substituted in Equation B14, the maximum SIR is obtained.

Carrying out the multiplications in B14 yields

$$SIR = \frac{[\alpha^* \underline{a}^\dagger \underline{M}_c^{-1} - \underline{g}^\dagger, \underline{B}_0^\dagger] \begin{bmatrix} \alpha \underline{a} \underline{a}^\dagger \underline{M}_c^{-1} \underline{a} - \underline{a} \underline{a}^\dagger \underline{g} + \underline{a} \underline{A}_0^\dagger \underline{B}_0 \\ \alpha \underline{A}_0 \underline{a}^\dagger \underline{M}_c^{-1} \underline{a} - \underline{A}_0 \underline{a}^\dagger \underline{g} + \underline{A}_0 \underline{A}_0^\dagger \underline{B}_0 \end{bmatrix}}{[\alpha^* \underline{a}^\dagger \underline{M}_c^{-1} - \underline{g}^\dagger, \underline{B}_0^\dagger] \begin{bmatrix} \alpha \underline{a} - \underline{M}_c \underline{g} + \underline{C}_0 \underline{B}_0 \\ \alpha \underline{C}_0^\dagger \underline{M}_c \underline{a} - \underline{C}_0^\dagger \underline{g} + \underline{Y} \underline{B}_0 \end{bmatrix}} \quad (B15)$$

Making the following substitutions,

$$q = \underline{a}^\dagger \underline{M}_c^{-1} \underline{a}, \quad p = \underline{A}_0^\dagger \underline{B}_0$$

Equation B15 now reduces to

$$SIR = \frac{\alpha^* \alpha q^2 - \alpha^* q \underline{a}^\dagger \underline{g} + \alpha^* q p + \alpha p^* q - \alpha \underline{g}^\dagger \underline{a} q + \underline{g}^\dagger \underline{a} \underline{a}^\dagger \underline{g} - \underline{g}^\dagger \underline{a} p - p^* \underline{a}^\dagger \underline{g} + p^* p}{\alpha^* \alpha q + \underline{g}^\dagger \underline{M}_c \underline{g} - \underline{g}^\dagger \underline{C}_0 \underline{B}_0 - \underline{B}_0^\dagger \underline{C}_0^\dagger \underline{g} + \underline{B}_0^\dagger \underline{Y} \underline{B}_0} \quad (B16)$$

To find the value of α required to maximize the SIR, the partial derivatives of Equation B16 are taken with respect to α^* and α as if they were independent real variables, obtaining

$$\frac{\partial \psi}{\partial \alpha^*} = D[\alpha q^2 - q \underline{a}^\dagger \underline{g} + qp] - N[\alpha q] = 0 \quad (B17)$$

$$\frac{\partial \psi}{\partial \alpha} = D[\alpha^* q^2 - q \underline{g}^\dagger \underline{a} + p^* q] - N[\alpha^* q] = 0$$

The two above equations are complex conjugates and yield the same information. Equation B17 simplifies to

$$D[\alpha q - \underline{a}^\dagger \underline{g} + p] - N\alpha = 0 \quad (B18)$$

From Equation B18 we obtain

$$SIR = \frac{N}{D} = q - [a^+g - p]/\alpha \quad (B19)$$

When the α corresponding to the minimum or maximum SIR is substituted in Equation B19, the corresponding extremum value for the SIR results. Substituting the values for N and D from Equation B6 into Equation B18, following some complicated algebraic manipulation, the following quadratic equation in α is obtained:

$$\alpha^2 qu^* + \alpha[\theta q - \mu u^*] - \mu\theta = 0 \quad (B20)$$

where $\mu = (a^+g - p)$

$$\theta = (B_0^+ Y B_0 - g^+ M_c g)$$

The solutions for α are

$$\alpha = \begin{cases} \mu/q & = \frac{(C_0^+ M_c^{-1} a - A_0)^+ B_0}{a^+ M_c^{-1} a} \end{cases} \quad (B21)$$

$$-(\theta/\mu^*) = \frac{B_0^+ [Y - C_0^+ M_c^{-1} C_0] B_0}{B_0^+ [A_0 - C_0^+ M_c^{-1} a]} \quad (B22)$$

Substituting Equation B21 in Equation B19

$$SIR_{\min} = q - \frac{\mu}{(\mu/q)} = 0$$

Substituting Equation B22 in Equation B19 gives the maximum SIR

$$SIR_{\max} = q - \frac{\mu}{\left(\frac{-\theta}{\mu^*}\right)} = q + \frac{\mu\mu^*}{\theta}$$

or, in terms of the original quantities,

$$SIR_{\max} = \tilde{a}^T \tilde{M}_c^{-1} \tilde{a} + \frac{\tilde{B}_0^T \{ [\tilde{C}_0^T \tilde{M}_c^{-1} \tilde{a} - \tilde{A}_0] [\tilde{C}_0^T \tilde{M}_c^{-1} \tilde{a} - \tilde{A}_0]^T \} \tilde{B}_0}{\tilde{B}_0^T [\tilde{Y} - \tilde{C}_0^T \tilde{M}_c^{-1} \tilde{C}_0] \tilde{B}_0} \quad (B23)$$

This quantity is denoted in Chapter 3 by $\psi_{\max}$.

Appendix C
COMPUTER PROGRAM LISTING

**** MAXSIR (N,L ARE USED FOR VARIABLE DIMENSIONS)

THIS PROGRAM CALCULATES THE LARGEST LOWER BOUND FOR ALL WEIGHTS
CONSTRAINED, THEN IT USES A BRANCH AND BOUND PROCEDURE TO
SELECT THE OPTIMUM PHASES IN ORDER TO MAXIMIZE THE SIGNAL TO
INTERFERENCE RATIO. IT WILL THEN PLOT PATTERNS FOR ARRAYS WITH
A VARIETY OF JAMMER CONFIGURATIONS.

```

      DIMENSION A(64),BETAJ(10),EXP(64),
+      M(64,64),MINVRS(64,64),
+      MJ(64,64),PJ(10),W(64),
+      WOPT(64),WQ(64),RN(64),
+      WTCONM(64),WCLB(64),THETAJ(10),PJDB(10),PROD(64,64)
      DIMENSION A(64),BETAJ(10),EXP(64),
+      M(64,64),MINVRS(64,64),MJ(64,64),
+      PJ(10),W(64),WOPT(64),WQ(64),RN(64),
+      WTCONM(64),WCLB(64),THETAJ(10),PJDB(10),PROD(64,64)
      BRANCH AND BOUND DATA

```

```

      DIMENSION CON(64),UNCON(64),BCONC(64)
      DIMENSION MC(64,64),CO(64,64),COCT(64,64),Y(64,64),NPASS(64)
      DIMENSION AT(64),AO(64),RO(64)
      DIMENSION R1(64),R2(64),ROPT(64)
      DIMENSION MCINV(64,64),MCINCO(64,64),
+      VOUT1(64),VOUT2(64),V3(64),U(64),V(64)

```

ASSUME THE LARGEST NUMBER OF BITS FOR PHASE SHIFTERS IS 6

```

      DIMENSION AMR(64)

```

```

      DIMENSION CON(64),UNCON(64),BCONC(64)
      DIMENSION MC(64,64),CO(64,64),COCT(64,64),Y(64,64)
      DIMENSION AT(64),AO(64),RO(64),NPASS(64)
      DIMENSION R1(64),R2(64),ROPT(64)
      DIMENSION MCINV(64,64),MCINCO(64,64),
+      VOUT1(64),VOUT2(64),V3(64),U(64),V(64)

```

```

      COMPLEX*16 EXP,M,MINVRS,WOPT,WQ,W,WCLB,WTCONM,BN1(64,64)
      COMPLEX*8 WSOL

```

```

      DOUBLE PRECISION A,BETAJ,BETAS,PJ,PQ,SIRNUM,SIRDEN,THETAS,RAD,
+      THETAJ,PJDB,PSICLB,SIP
      REAL*8 PSIMAX,SICLB
      COMPLEX*16 MC,CO,COCT,Y,AT,AO,RO,AMR,ROPT,BCONC
      COMPLEX*16 MCINV,MCINCO,VOUT1,VOUT2,V3,U,V,A1
      COMPLEX*16 WOPT,WQ,WCLB,M,WTCONM
      INTEGER R1,R2,CON,UNCON

```

*N,L ARE USED FOR VARIABLE DIMENSIONING
COMMON N,L

** INPUT DATA DEFINITIONS
N,L DETERMINE DIMENSION FOR SEVERAL VECTORS AND ARRAYS

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000405
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540

N=NUMBER OF ELEMENTS IN THE ADAPTIVE ARRAY
 L=NUMBER OF JAMMERS
 A=APERATURE ILLUMINATION VECTOR (IF A IS ALL 1'S, THERE IS UNIFORM
 WEIGHT AND NO SIDELOBE ATTENUATION)
 THETAJ=JAMMING SIGNAL DIRECTION ANGLE FROM MECH. BORESIGHT
 IN DEGREES (L INDICATES LTH JAMMER)
 THETAS=DESIRED SIGNAL DIRECTION ANGLE FROM MECH. BORESIGHT
 IN DEGREES
 PJDB=NOISE POWER OF EACH JAMMER IN DECIBELS (0 TO 100)
 (INDEX L INDICATES LTH JAMMER)
 PQ=NOISE POWER OUTPUT OF EACH ELEMENT (QUIESCENT NOISE POWER)
 AS A RATIO OF SIGNAL TO NOISE
 NBITS = NUMBER OF PHASE SHIFTER BITS (USED IN WEIGHT SUBROUTINE)
 RANGE 2 TO 6 BITS

NAMELIST/INPUT1/N,L,A,THETAJ,THETAS,PJDB,PQ,NBITS,LIMNB,TEMCLB,
 + KCMAX,KCMAXN

DATA DLAM/0.5/,PI/3.141592654/,DEG/57.296/,RAD/0.017453/

INITIALIZATION

READ (5,INPUT1,END=2168)

BETAS =2.0 * PI*DLAM*DSIN(THETAS*RAD)

DO 5 I=1,L

BETAJ(I) = 2.0*PI*DLAM*DSIN(THETAJ(I)*RAD)

CONVERT PJDB FROM DB TO A RATIO OF SIGNAL TO JAMMER

PJ(I) =10.0** (PJDB(I)*0.1)

5 CONTINUE

CALCULATE THE QUIESCENT WEIGHT VECTOR AND THE JAMMER COVARIANCE MATRIX
 AND ITS INVERSE

CALL MWQ(A,BETAJ,BETAS,EXP,M,MINVRS,MJ,PJ,PQ,WQ,
 + PROD)

CALCULATE THE LARGEST CURRENT LOWER BOUND WHERE ALL WEIGHTS
 ARE CONSTRAINED TO A DISCRETE SET OF VALUES (PSICLB)

THE ASSOCIATED WEIGHT VECTOR IS ALSO OUTPUT

10 CALL SIRCLB(W,WQ,M,MINVRS,WTCONM,WOPT,WCLB,NBITS,RN,A,
 + PSICLB)

*** REVISED AND SIMPLIFIED BRANCH AND BOUND PROGRAM

N IS THE NUMBER OF ARRAY ELEMENTS

NBITS IS THE NUMBER OF PHASE SHIFTER BITS

WOPT IS THE OPTIMUM WEIGHT VECTOR OVER THE CONTINUUM

00000550
 00000560
 00000570
 00000580
 00000590
 00000600
 00000610
 00000620
 00000630
 00000640
 00000650
 00000660
 00000670
 00000680
 00000690
 00000701
 00000705
 00000710
 00000720
 00000730
 00000740
 00000750
 00000760
 00000770
 00000780
 00000790
 00000800
 00000810
 00000820
 00000830
 00000840
 00000850
 00000860
 00000870
 00000880
 00000890
 00000900
 00000910
 00000920
 00000930
 00000940
 00000950
 00000960
 00000970
 00000980
 00000990
 00001000
 00001010
 00001020
 00001030
 00001040
 00001050
 00001060
 00001070
 00001080

C	WQ IS THE QUIESCENT (UNJAMMED) WEIGHT VECTOR	00001090
C	WQ IS PARTITIONED INTO WQ(UNCONSTRAINED)	00001100
C	AND WQ(CONSTRAINED)	00001110
C	BCONC IS THE CONSTRAINED OPTIMAL WEIGHT VECTOR. ONE ELEMENT IS	00001120
C	DEFINED EACH TIME CHSW IS EXECUTED.	00001130
C		00001140
C		00001150
C	BOPT IS THE UNCONSTRAINED OPTIMAL WEIGHT VECTOR. IT IS CALCULATED	00001160
C	FROM PARAMETERS DEFINED IN THE PSIMAX SUBROUTINE	00001170
C	PSICLB IS THE LARGEST CURRENT LOWER BOUND FOR THE SIR	00001180
C	WCLB IS THE WEIGHT VECTOR CORRESPONDING TO PSICLB	00001190
C	M IS THE JAMMER COVARIANCE MATRIX	00001200
C	SIRMAX IS THE OPTIMUM SIR OVER THE CONTINUUM--NOT USED	00001210
C		00001220
C		00001230
C		00001240
C	** THIS IS THE BRANCH AND BOUND PORTION OF THE PROGRAM	00001250
C		00001260
C	KCNTN=0	00001270
C	KCNT=0	00001280
C	KBOT = 0	00001290
C	* THE INDEX I WILL RUN FROM 1 TO N+1	00001300
C	* BUT I-1 WILL BE USED FOR DIMENSIONING THE ARRAYS	00001305
C		00001310
C	WRITE (6,*) KCMAX	00001312
C	I = 1	00001320
C	K22CNT = 0	00001330
C		00001340
C	NPASS(I) = 1	00001350
C	IF(NBITS.NE.2) GO TO 20	00001360
C	SICLB=PSICLB	00001370
C	GO TO 30	00001380
C	20 IF(SICLB.GT.PSICLB) GO TO 30	00001385
C	20 SICLB=TEACLB	00001390
C	WRITE (6,*) SICLB	00001400
C	30 CONTINUE	00001410
C	CALCULATE THE VECTOR FOR THE WEIGHT VALUES BASED ON THE NUMBER OF	00001420
C	PHASE SHIFTER BITS	00001430
C		00001440
C	MDIM = 2 ** NBITS	00001450
C		00001460
C	RRAD = 2.0 * PI/FLOAT(MDIM)	00001470
C		00001480
C	DO 50 J=1,MDIM	00001490
C	AMR(J) = CEXP((0.0,1.0)*(J-1)*RRAD)	00001500
C	50 CONTINUE	00001510
C		00001520
C	FOR FIRST TIME THROUGH THE BRANCH RULE, M IS NOT PARTITIONED AND	00001530
C	BOPT = WOPT AND PSIMAX IS PSICLB	00001540
C	WCLB, PSICLB IS THE SOLUTION FOR THE UNCONSTRAINED CASE.	00001550
C		00001560
C	NML = N	00001570
C	CALL ANGLE(WOPT,NML,ANGL,PI)	00001580
C		00001590

0000016000
0000016100
0000016200
0000016300
0000016400
0000016500
0000016600
0000016700
0000016800
0000016900
0000017000
0000017100
0000017200
0000017300
0000017400
0000017500
0000017600
0000017700
0000017800
0000017900
0000018000
0000018100
0000018200
0000018300
0000018400
0000018500
0000018600
0000018700
0000018800
0000018900
0000019000
0000019100
0000019200
0000019300
0000019400
0000019500
0000019600
0000019700
0000019800
0000019900
0000020000
0000020100
0000020200
0000020300
0000020350
0000020400
0000020500
0000020600
0000020700
0000020800
0000020810
0000020820
0000020830
0000020840

```
100 I = I+1  
    NPASS(I) = 1  
    WRITE (6,*) I, NPASS(I)
```

SELECT VALUE FOR NEWLY CONSTRAINED COMPONENT OF WEIGHT VECTOR

```
155 RCONC(L)=AMR(R2(L))
```

160 CONTINUE

```
IF(I,GE,(N+1)) GO TO 200
```

REORDER AND PARTITION THE MATRICES USED IN PSIMAX

```
CALL REORDR(NHL,N,BCONC,M,MC,CO,COCT,Y,AT,AO,BO,L,WQ,CON,UNCON)
CALL SIMAX(NHL,L,MC,CO,COCT,Y,AT,AO,BO,MCINV,PROD,MCINCO,VOUT1,
VOUT2,V3,U,V,AI,PSIMAX,RN1)
```

```

      KCNT = KCNT + 1
      IF (KCNT .EQ. KCNMAX) WRITE (6,*) KCNT, KROT, SICLB

```

```
IF (KCNT.EQ.(KCMAX+1)) KCMAX=KCMAX+1000
```

```
WRITE (6,*) PSIMAX,STCLB
```

```
IF (PSIMAX.LT.SICLB) GO TO 400
```

$$ROPT(i) = A1 * U(i) - V(i)$$

CALL ANGLE (ADPT,NML,ANGL,PI)

CALL R1R2(I,MDIM,ANGL,R1,R2,PI,SEQ,IND,N)

GO TO 100

```
200 CALL NSIMAX(BCONC,WQ,M,WTCONM,N,PSIMAX)
```

KCN TN = KCN TN + 1

KCNT = KCNT+1

K80Y K80Y 1

```
IF (KCNT.EQ.25000) WRITE (6,*) KCNT,KBOT,SIC,B,WCLB
```

```
WRITE (6,*) 'I,PSIMAX,SICLR
```

TF (PSIMAX,GE,SICLH) GO TO 210

```
IF (KCNTN.EQ.KCMAXN) WRITE (6,*) KCNTN,KBOT,SICLR,KCMAXN
```

```
IF (KCNTN.EQ.(KCMAXN+1)) KCMAXN=KCMAXN+1000
```

GO TO 400

```
210 IF (PSIMAX.EQ.SICLB) K22CNT=K22CNT+1
```

```

      WRITE (6,*) PSIMAX,SICLB,K22CNT
      SICLB = PSIMAX
      DO 250 J=1,N
      WCLB(J) = BCONC(J)
250  CONTINUE
      IF(K22CNT.EQ.10) GO TO 1999
C
C
C
      ** FATHOM VERTEX
C
C
C
400  NPASS(L) = NPASS(L) + 1
      IF (NPASS(L).GE.(MDIM+1)) GO TO 500
      IF (NPASS(L).NE.2) GO TO 450
      IND = 1
      GO TO 150
C
C
450  I = I-1
      L = L-1
      NML = NML+1
      CALL RHIGH(IND,SEQ,R1,R2,MDIM,I,N)
      I = I+1
      L = L+1
      NML = NML-1
      GO TO 150
C
C
500  NPASS(L) = 1
      I = I-1
      L = L-1
      NML = NML+1
      IND = 0
      IF (I.GT.1) GO TO 400
1999 CONTINUE
      DO 2000 KKK=1,MDIM
      WSOL = AMR(KKK)
      WRITE (6,*) WSOL
2000 CONTINUE
      DO 2100 KKK=1,N
      WSOL = WCLB(KKK)
      WRITE (6,*) WSOL
2100 CONTINUE
      WRITE (6,*) SICLB,NBITS,KCNT,KBOT
C
C
C
      CALC. ARRAY FACTOR AND PLOT GAIN FOR ALL THREE WEIGHT VECTORS
C
C
C
      CALL ARFACT(N,WCLB,WOPT,WQ,THETAJ,THETAS)
      IF(NBITS.EQ.LIMNB) GO TO 2168
      NBITS = NBITS+1
      GO TO 10
2168 CONTINUE
      STOP
      DEBUG SUBCHK
      END
C
C
      SUBROUTINE MWQ(A,BETAJ,BETAS,EXP,M,MINVRS,MJ,PJ,

```

```

00002085
00002090
00002100
00002110
00002120
00002125
00002130
00002140
00002150
00002160
00002170
00002180
00002190
00002200
00002210
00002220
00002230
00002240
00002250
00002260
00002270
00002280
00002290
00002300
00002310
00002320
00002330
00002340
00002350
00002360
00002370
00002373
00002374
00002375
00002376
00002377
00002382
00002383
00002386
00002388
00002390
00002392
00002393
00002395
00002396
00002400
00002410
00002420
00002430
00002440
00002450
00002460
00002470
00002480
00002490

```

```

      +
      PQ,WQ,PROD)
C
C THIS ROUTINE CALCULATES THE QUIESCENT WEIGHT VECTOR
C AND THE JAMMER CONFIGURATION (COVARIANCE MATRIX)
C PLUS ITS INVERSE
C
      DIMENSION A(N),BETAJ(L),EXP(N),
      + M(N,N),MINVRS(N,N),MJ(N,N),PJ(L),WQ(N),
      + PROD(N,N)
C
      COMPLEX*16 EXP,M,MINVRS,MJ,WQ,SUM,PROD,TH(100),R(100)
C
      DOUBLE PRECISION A,BETAJ,BETAS,PJ,PQ
C
      COMMON N,L
C
      CALCULATE THE QUIESCENT WEIGHT VECTOR (WQ=A(TRANPOSE) * EXP)
C
      DO 100 K=1,N
      EXP(K) = CDEXP((0.0,-1.0)*(K-1)*BETAS)
      WQ(K) = A(K) * EXP(K)
100 CONTINUE
C
      THIS LOGIC FILLS IN THE FIRST ROW OF MJ
C
      DO 200 J=1,N
      SUM = (0.0,0.0)
      DO 150 I=1,L
      SUM = SUM + PJ(I)*CDEXP((0.0,1.0)*(J-1)*BETAJ(I))
150 CONTINUE
      MJ(1,J) = SUM
200 CONTINUE
C
      FILL IN THE MAIN DIAGONAL ELEMENTS
C
      DO 300 K=2,N
      MJ(K,K) = MJ(1,1)
300 CONTINUE
C
      FILL IN UPPER TRIANGULAR PORTION OF THE MJ MATRIX (TOEPLITZ)
      HERMITIAN IMPLIES M(J,I)=M(I,J)*
C
      NM1 = N-1
      NM2 = N-2
      DO 700 K=2,NM1
      DO 500 J=1,NM2
      JKT = J+K-N
      IF (JKT) 1,1,500
      1 MJ(J+1,J+K) = MJ(1,K)
      MJ(J+K,J+1) = DCONJG(MJ(J+1,J+K))
      IF ((J+K).EQ.N) GO TO 600
500 CONTINUE
600 MJ(K,1) = DCONJG(MJ(1,K))
700 CONTINUE
      MJ(N,1) = DCONJG(MJ(1,N))

```

```

00002500
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700
00002710
00002720
00002730
00002740
00002750
00002760
00002770
00002780
00002790
00002800
00002810
00002820
00002830
00002840
00002850
00002860
00002870
00002880
00002890
00002900
00002910
00002920
00002930
00002940
00002950
00002960
00002970
00002980
00002990
00003000
00003010
00003020
00003030
00003040

```


C	INITIALIZE SEED FOR RN GENERATOR IN WEIGHT SUBROUTINE	00003600
C	IS = 5	00003610
C	KR = 2	00003620
C	PSICLB = 0.0	00003630
C	DO 700 I=1,KR	00003640
C	GENERATE A WEIGHT VECTOR FOR ALL WEIGHTS CONSTRAINED	00003650
C	THIS ROUTINE WILL GENERATE ONE WEIGHT VECTOR W	00003660
C	CALL WEIGHT(A,N,NBITS,RN,IS,W)	00003670
C	CALCULATE THE CURRENT LOWER BOUND FOR THE SIR * * *	00003680
C	CALC. W(TRANPOSE-CONJUGATE)*WQ AND	00003690
C	WQ(TRANPOSE-CONJUGATE)*W	00003700
C	W1 = (0.0,0.0)	00003710
C	DO 100 I=1,N	00003720
C	W1 = W1 + DCONJG(W(I1))*WQ(I1)	00003730
C	100 CONTINUE	00003740
C	NUMERATOR OF SIR = W1 * W2 (REAL) SINCE M IS HERMITIAN	00003750
C	SIRNUM = W1 * DCONJG(W1)	00003760
C	CALCULATE THE DENOMINATOR FOR SIRCLB = W(TRANPOSE-CONJUGATE) * M * W	00003770
C	CALCULATE W(TRANPOSE-CONJUGATE) * M	00003780
C	DO 200 K=1,N	00003790
C	WTCONM(K) = (0.0,0.0)	00003800
C	200 CONTINUE	00003810
C	DO 400 K=1,N	00003820
C	DO 300 I2=1,N	00003830
C	WTCONM(K) = WTCONM(K) + DCONJG(W(I2))*M(I2,K)	00003840
C	300 CONTINUE	00003850
C	400 CONTINUE	00003860
C	CALCULATE W(TRANPOSE-CONJUGATE) * M * W = SIR DENOMINATOR (REAL)	00003870
C	SINCE M IS HERMITIAN	00003880
C	SIRDEN = 0.0	00003890
C	DO 500 K=1,N	00003900
C	SIRDEN = SIRDEN + WTCONM(K)*W(K)	00003910
C	500 CONTINUE	00003920
C	CALCULATE SIR(CURRENT LOWER BOUND) (REAL) SINCE M HERMITIAN	00003930
		00003940
		00003950
		00003960
		00003970
		00003980
		00003990
		00004000
		00004010
		00004020
		00004030
		00004040
		00004050
		00004060
		00004070
		00004080
		00004090
		00004100
		00004110
		00004120
		00004130
		00004140

```

C      SIR = SIRNUM/SIRDEN
C
C      FIND LARGEST CURRENT LOWER BOUND FOR SIR AND THE CORR. WCLB
C      IF (SIR.LE.PSICLB) GO TO 700
C      PSICLB = SIR
C
C      DO 600 K=1,N
C      WCLB(K) = W(K)
C      600 CONTINUE
C      700 CONTINUE
C
C      CALCULATE THE OPTIMUM WEIGHT VECTOR , WOPT = MINVRS * WQ
C
C      DO 800 K=1,N
C      WOPT(K) = (0.0,0.0)
C      800 CONTINUE
C
C      DO 950 K=1,N
C      DO 900 I=1,N
C      WOPT(K) = WOPT(K) + MINVRS(K,I)*WQ(I)
C      900 CONTINUE
C      950 CONTINUE
C
C      CALCULATE THE MAXIMUM SIGNAL TO INTERFERENCE RATIO OVER THE CONTINUUM
C      UPPER BOUND (NOT ACTUALLY USED)
C
C      SIRMAX = 0.0
C
C      DO 1000 K=1,N
C      SIRMAX = SIRMAX + DCONJG(WQ(K))*WOPT(K)
C      1000 CONTINUE
C
C      PRINT OUT WCLB,PSICLB, AND WOPT AND SIRMAX
C
C      WRITE (6,1) PSICLB,SIRMAX
C      1 FORMAT (1H, 'OUTPUT FROM SIRCLB ROUTINE****',/,
C      + 'PSICLB=',F16.10,5X,'SIRMAX=',F16.10,/,/, 'WCLB',10X,'WOPT',
C      + '10X',10X,'WQ',10X,'M',10X,'N',10X,'NBITS',10X,'PSICLB',10X,
C      + 'SIRMAX',/)
C
C      WRITE (6,*) WCLB,WOPT,WQ,N,NBITS,PSICLB,SIRMAX
C      WRITE (1,*) WCLB,WOPT,WQ,N,NBITS,PSICLB,SIRMAX
C
C      RETURN
C      DEBUG SUBCHK
C      END
C
C      SUBROUTINE WEIGHT(A,N,NBITS,RN,IS,W)

```

```

00004150
00004160
00004170
00004180
00004190
00004200
00004210
00004220
00004230
00004240
00004250
00004260
00004270
00004280
00004290
00004300
00004310
00004320
00004330
00004340
00004350
00004360
00004370
00004380
00004390
00004400
00004410
00004420
00004430
00004440
00004450
00004460
00004470
00004480
00004490
00004500
00004510
00004520
00004530
00004540
00004550
00004560
00004570
00004580
00004590
00004600
00004610
00004620
00004630
00004640
00004650
00004660
00004670
00004680
00004690

```

THIS ROUTINE CALCULATES ONE SOLUTION VECTOR, W, FROM THE BOTTOM
OF THE TREE. EACH ELEMENT IN W TAKES ON A DISCRETE, COMPLEX VALUE
NBITS = NO. OF PHASE SHIFTER BITS

DIMENSION A(N), RN(N), W(N)

COMPLEX W*16
DOUBLE PRECISION BETAK, A
DATA PI/3.141592654/
NVAL = 2 ** NBITS
VAL = NVAL

GENERATE A SEQUENCE OF N RANDOM NUMBERS

CALL RNDUN(15, RN, N)

CALCULATE N ELEMENTS OF THE VECTOR W

DO 200 K=1, N

CHOOSE W(K)=AM (AM TAKES ON VALUES BETWEEN 1 AND 2 ** NBITS)

DO 100 L1=1, NVAL
IF(RN(K).LE.(FLOAT(L1)/VAL)) GO TO 150
100 CONTINUE

150 BETAK = -(FLOAT(L1) * 2.0*PI)/FLOAT(NVAL)
W(K) = A(K) * CDEXP((0.0,1.0)*BETAK)
200 CONTINUE

300 CONTINUE
RETURN
DEBUG SUBCHK
END

SUBROUTINE ANGLE(BOPT,NML,ANGL,PI)

COMPLEX*16 BOPT(NML)

ONLY ONE WEIGHT IS USED AT A TIME

IF(REAL(BOPT(1)).NE.0.0) GO TO 50
IF(AIMAG(BOPT(1)).LT.0.0) ANGL = 1.5*PI
IF(AIMAG(BOPT(1)).GT.0.0) ANGL = 0.5*PI
IF(AIMAG(BOPT(1)).EQ.0.0) WRITE (6,1)
GO TO 100
50 ANGL = ATAN(AIMAG(BOPT(1))/REAL(BOPT(1)))
IF((AIMAG(BOPT(1)).LT.0.0).AND.(REAL(BOPT(1)).LT.0.0)) ANGL =
+ ANGL+PI
IF((AIMAG(BOPT(1)).GT.0.0).AND.(REAL(BOPT(1)).LT.0.0)) ANGL =
+ ANGL+PI

00004700
00004710
00004720
00004730
00004740
00004750
00004760
00004770
00004780
00004790
00004800
00004810
00004820
00004830
00004840
00004850
00004860
00004870
00004880
00004890
00004900
00004910
00004920
00004930
00004940
00004950
00004960
00004970
00004980
00004990
00005000
00005100
00005110
00005120
00005130
00005140
00005150
00005160
00005170
00005180
00005190
00005200
00005210
00005220
00005230
00005240
00005250
00005260
00005270
00005280
00005290
00005300
00005310
00005320

```

100 IF (ANGL.LT.0.0) ANGL=ANGL+2*PI
CONTINUE

1 FORMAT(1H , 'REAL AND IMAG. PARTS OF WOPT BOTH ZERO--ANGLE RTN. ')
RETURN
DEBUG SUBCHK
END

SUBROUTINE R1R2(I,MDIM,ANGL,R1,R2,PI,SEQ,IND,N)

THIS ROUTINE CALCULATES THE R'S FOR A(R) (R=1,...,MDIM)
R IS THE INDEX INTO THE ALLOWED VALUES FOR THE WEIGHTS DUE TO PHASE
SHIFTER BITS R IS ROUNDED UP TO INTEGER VALUE

SEQ DENOTES THE DIRECTION (-1=CLOCKWISE AND +1= COUNTER CLOCKWISE)
FOR THE ASSIGNMENT OF VALUES FOR HIGHER ORDER R'S
THESE VALUES ALTERNATE CW AND CCW AROUND THE CIRCLE FORMED BY
THE VALUES FOR A(R)

INTEGER R1(N),R2(N),SEQ
C = FLOAT(MDIM)/(2.0*PI)

R1(I) GIVES THE CLOSEST ALLOWED VALUE A(R) TO THE WEIGHT W(I)
R2(I) GIVES THE SECOND CLOSEST VALUE A(R) TO THE WEIGHT W(I)

R1(I)=AINT(C*ANGL + 0.5) + 1
IF (R1(I).GE.(MDIM+1)) R1(I)=R1(I)-MDIM

IF((C*ANGL)-(R1(I)-1)) 210,220,220

THE ALLOWED VALUE FOR WEIGHT W IS LESS THAN THE CLOSEST INTEGER
MULTIPLE OF THE FUNDAMENTAL ANGLE DETERMINED BY THE PHASE SHIFTERS

210 R2(I) = R1(I)-1
IF (R2(I).LE.0) R2(I)=R2(I)+MDIM
SEQ = -1
GO TO 300

220 R2(I) = R1(I)+1
IF (R2(I).GE.(MDIM+1)) R2(I)=R2(I)-MDIM
SEQ = 1

300 IND = 0

RETURN
DEBUG SUBCHK
END
SUBROUTINE REORDR(NML,N,BCONC,M,MC,CO,COCT,Y,AT,AO,BO,L,WQ,CON,UNC
+ ON)

```

```

00005330
00005340
00005350
00005360
00005370
00005380
00005390
00005400
00005410
00005420
00005430
00005440
00005450
00005460
00005470
00005480
00005490
00005500
00005510
00005520
00005530
00005540
00005550
00005560
00005570
00005580
00005590
00005600
00005610
00005620
00005630
00005640
00005650
00005660
00005670
00005680
00005690
00005700
00005710
00005720
00005730
00005740
00005750
00005760
00005770
00005780
00005790
00005800
00005810
00005820
00005830
00005840
00005850
00005860
00005870

```


C	PARTITION SUBROUTINE ****	00005880
C	THIS ROUTINE PARTITIONS AND REORDERS THE COVARIANCE MATRIX AND THE A	00005890
C	VECTOR BASED ON B (CONCATENATED)	00005900
C	OUTPUTS FROM THIS ROUTINE ARE MC, CO, COCT, Y, AT, AO, BO	00005910
C		00005920
C	COMPLEX*16 M(N,N), MC(NML,NML), CO(NML,L), COCT(L,NML), Y(L,L)	00005930
C	COMPLEX*16 AT(NML), AO(L), BO(L), BCONC(N)	00005940
C	COMPLEX*16 WQ(N)	00005950
C	INTEGER CON(N), UNCON(N)	00005960
C		00005970
C	DO 100 J=1,L	00005980
C	AO(J)=WQ(J)	00005990
C	BO(J)=BCONC(J)	00006000
C	100 CONTINUE	00006010
C		00006020
C	K1=1	00006030
C	KP=L+1	00006040
C	DO 200 K=KP,N	00006050
C	AT(K1)=WQ(K)	00006060
C	UNCON(K1)=K	00006070
C	K1=K1+1	00006080
C	200 CONTINUE	00006090
C		00006100
C	300 DO 500 K1=1,L	00006110
C	DO 400 K2=1,L	00006120
C	Y(K1,K2)=H(K1,K2)	00006130
C	400 CONTINUE	00006140
C	500 CONTINUE	00006150
C	DO 700 K1=1,NML	00006160
C	DO 600 K2=1,NML	00006170
C	MC(K1,K2)=M(UNCON(K1),UNCON(K2))	00006180
C	600 CONTINUE	00006190
C	700 CONTINUE	00006200
C	DO 900 K1=1,NML	00006210
C	DO 800 K2=1,L	00006220
C	CO(K1,K2)=H(UNCON(K1),K2)	00006230
C	800 CONTINUE	00006240
C	900 CONTINUE	00006250
C	DO 1100 K1=1,L	00006260
C	DO 1000 K2=1,NML	00006270
C	COCT(K1,K2)=CONJG(CO(K2,K1))	00006280
C	1000 CONTINUE	00006290
C	1100 CONTINUE	00006300
C		00006310
C	RETURN	00006320
C	DEBUG SUBCHK, INIT(MC,COCT,CO,Y,AT,AO,BO,CON,UNCON)	00006330
C	DEBUG SUBCHK	00006340
C	END	00006350
C	SUBROUTINE SIMAX(NML,L,MC,CO,COCT,Y,AT,AO,BO,MCINV,PROD,	00006360
C	+ MCINCO,VOUT1,VOUT2,V3,U,V,A1,PSIMAX,BN1)	00006370
C		00006380
C		00006390
C		00006400
C		00006410
C		00006420

C ** PSIMAX CALCULATION SUBROUTINE

THIS ROUTINE CALCULATES THE MAXIMUM SIGNAL TO INTERFERENCE RATIO.
IT ALSO OUTPUTS U,V, AND A1 TO BE USED IN BOPT CALC.(SUBROUTINE).

REAL*8 PSIMAX
COMPLEX*16 MC(NML,NML),CO(NML,L),COCT(L,NML),Y(L,L),
+ AT(NML),AO(L),BO(L),U(NML),V(NML),DENOM,A1,NUMER,
+ ATCU,TM(100),RNI(NML,NML),R(100)
COMPLEX*16 MCINV(NML,NML),PROD(NML,NML),MCINCO(NML,L),
+ VOUT1(L),VOUT2(L),V3(L),BOCTAO,VCTAT

C * INVERT MC AND VERIFY MC * MC(INVERSE) = I

IF(L.NE.(NML+L-1)) GO TO 1015
MCINV(1,1) = 1./MC(1,1)
GO TO 99
1015 IF (L.NE.(NML+L-2)) GO TO 1000
DENIM=MC(1,1)*MC(2,2)-MC(1,2)*MC(2,1)
MCINV(1,1)=MC(2,2)/DENIM
MCINV(1,2)=-MC(1,2)/DENIM
MCINV(2,1)=-MC(2,1)/DENIM
MCINV(2,2)=MC(1,1)/DENIM
GO TO 99
1000 DO 1010 INDEX=1,NML
TM(INDEX)=MC(INDEX,1)/MC(1,1)
1010 CONTINUE
NMIE=NML-1
DO 1020 NIND=1,NMIE
NINDP1=NIND+1
R(NIND)=TM(NINDP1)
1020 CONTINUE
CALL ZORN(R,MCINV,NML)
DO 1011 IZR=1,NML
DO 1012 IZC=1,NML
MCINV(IZR,IZC)=MCINV(IZR,IZC)/MC(1,1)
1012 CONTINUE
1011 CONTINUE
CALL COMXMP(MC,NML,MCINV,NML,NML,NML,NML,PROD,NML)

C ** CALCULATE U=MC(INVERSE) * AT (ORDER NMLX1)

99 CALL COMXMP(MCINV,NML,AT,NML,NML,NML,1,U,NML)

C ** CALCULATE V=MC(INVERSE)*CO*BO

CALL COMXMP(MCINV,NML,CO,NML,NML,NML,L,MCINCO,NML)

CALL COMXMP(MCINCO,NML,BO,L,NML,L,1,V,NML)

C ** CALCULATE NUMER=BO(CONJ. TRANSPOSE)* (Y*BO - COCT*V)

CALL COMXMP(COCT,L,V,NML,L,NML,1,VOUT1,L)
CALL COMXMP(Y,L,BO,L,L,L,1,VOUT2,L)

00006430
00006440
00006450
00006460
00006470
00006480
00006490
00006500
00006510
00006520
00006530
00006540
00006550
00006560
00006561
00006562
00006563
00006564
00006566
00006567
00006568
00006569
00006571
00006572
00006574
00006580
00006590
00006600
00006610
00006620
00006630
00006640
00006650
00006660
00006670
00006680
00006690
00006700
00006710
00006720
00006730
00006740
00006750
00006760
00006770
00006780
00006790
00006800
00006810
00006820
00006830
00006840
00006850
00006860
00006870

C	DO 100 K1=1,L	00006880
	V3(K1) = VOUT2(K1) - VOUT1(K1)	00006890
C	100 CONTINUE	00006900
	NUMER = (0.0,0.0)	00006910
C		00006920
	DO 200 K1=1,L	00006930
	NUMER = NUMER + DCONJG(B0(K1)) * V3(K1)	00006940
C	200 CONTINUE	00006950
		00006960
C		00006970
C	** CALCULATE DENOM=B0(CONJ. TRANSPOSE)*A0 -V(CONJ. TRANSPOSE)*AT	00006980
C		00006990
	B0CTA0 =(0.0,0.0)	00007000
	VCTAT =(0.0,0.0)	00007010
C		00007020
	DO 300 K1=1,L	00007030
	B0CTA0 = B0CTA0 + DCONJG(B0(K1))*A0(K1)	00007040
C	300 CONTINUE	00007050
		00007060
C		00007070
	DO 400 K1=1,NML	00007080
	VCTAT = VCTAT + DCONJG(V(K1))*AT(K1)	00007090
C	400 CONTINUE	00007100
		00007110
C	DENOM = B0CTA0 - VCTAT	00007120
		00007130
C	** CALCULATE ALPHA1	00007140
C		00007150
	A1 = NUMER/DENOM	00007160
C		00007170
C	** CALCULATE PSIMAX	00007180
C		00007190
	ATCU = (0.0,0.0)	00007200
C		00007210
	DO 500 K1=1,NML	00007220
	ATCU = ATCU + DCONJG(AT(K1))*U(K1)	00007230
C	500 CONTINUE	00007240
		00007250
C		00007260
	PSIMAX = ATCU + DCONJG(DENOM)/A1	00007270
C		00007280
	CONTINUE	00007290
C		00007300
	RETURN	00007310
	DEBUG SUBCHK	00007320
	END	00007330
C		00007340
	SUBROUTINE RHIGH(IND,SEQ,R1,R2,MDIM,I,N)	00007350
C		00007360
C	** THIS ROUTINE CALCULATES THE HIGHER ORDER R VECTORS--R3,R4,...,RMDIM	00007370
C	OUTPUT GOES TO R1 IF IND=0 AND R2 IS FILLED IF IND=1.	00007380
C		00007390
		00007400
		00007410
	INTEGER SEQ,R1(N),R2(N)	00007420

```

C      IF(IND.EQ.1) GO TO 300
C
C      ** IND=0--FILL R2 VECTOR
C
C      IND = 1
C      IF(SFQ.EQ.1) GO TO 100
C      R2(I) = R2(I) + 1
C      IF(R2(I).GT.MDIM) R2(I)=R2(I)-MDIM
C      GO TO 500
C
C      100 R2(I) = R2(I)-1
C      IF(R2(I).LE.0) R2(I)=R2(I)+MDIM
C
C      GO TO 500
C
C      ** IND = 1--FILL R1 VECTOR
C
C      300 IND = 0
C      IF(SFQ.EQ.1) GO TO 400
C      R1(I) = R1(I) + 1
C      IF(R1(I).GT.MDIM) R1(I)=R1(I)-MDIM
C      GO TO 500
C
C      400 R1(I) = R1(I)-1
C      IF(R1(I).LE.0) R1(I)=R1(I)+MDIM
C
C      500 CONTINUE
C
C      RETURN
C      DEBUG SUBCHK
C      END
C
C      SUBROUTINE NSIMAX(BCONC,WQ,M,WTCONM,N,PSIMAX)
C
C      THIS ROUTINE WILL CALC. THE LARGEST CURRENT LOWER BOUND FOR
C      THE CASE WHERE ALL WEIGHTS ARE CONSTRAINED TO DISCRETE VALUES.
C
C
C
C
C      REAL*8 PSIMAX,SIRNUM,SIRDEN
C      COMPLEX*16 M(N,N),BCONC(N),WQ(N),WTCONM(N),W1,W2
C
C      CALCULATE THE CURRENT LOWER BOUND FOR THE SIR * * *
C
C      CALC. W(TRANSPOSE=CONJUGATE)*WQ AND
C      WQ(TRANSPOSE=CONJUGATE)*W
C
C      W1 = (0.0,0.0)
C
C      DO 100 I=1,N
C      W1 = W1 + DCONJG(BCONC(I1))*WQ(I1)
C      100 CONTINUE

```

```

00007030
00007040
00007050
00007060
00007070
00007080
00007090
00007100
00007110
00007120
00007130
00007140
00007150
00007160
00007170
00007180
00007190
00007200
00007210
00007220
00007230
00007240
00007250
00007260
00007270
00007280
00007290
00007300
00007310
00007320
00007330
00007340
00007350
00007360
00007370
00007380
00007390
00007400
00007410
00007420
00007430
00007440
00007450
00007460
00007470
00007480
00007490
00007500
00007510
00007520
00007530
00007540
00007550
00007560
00007570
00007580
00007590
00007600
00007610
00007620
00007630
00007640
00007650
00007660
00007670
00007680
00007690
00007700
00007710
00007720
00007730
00007740
00007750
00007760
00007770
00007780
00007790
00007800
00007810
00007820
00007830
00007840
00007850
00007860
00007870
00007880
00007890
00007900
00007910
00007920
00007930
00007940
00007950
00007960
00007970

```

C	NUMERATOR OF SIR = W1 * W2 (REAL) SINCE M IS HERMITIAN	00007980
C	SIRNUM = W1 * DCONJG(W1)	00007990
C	CALCULATE THE DENOMINATOR FOR SIRCLB = W(TRANSPPOSE=CONJUGATE) * M * W	00008000
C		00008010
C		00008020
C		00008030
C		00008040
C		00008050
C		00008060
C	CALCULATE W(TRANSPPOSE=CONJUGATE) * M	00008070
C		00008080
C		00008090
C	DO 200 K=1,N	00008100
C	WTCONM(K) = (0.0,0.0)	00008110
C	200 CONTINUE	00008120
C		00008130
C	DO 400 K=1,N	00008140
C	DO 300 I2=1,N	00008150
C	WTCONM(K) = WTCONM(K) + DCONJG(BCONC(I2))*M(I2,K)	00008160
C	300 CONTINUE	00008170
C	400 CONTINUE	00008180
C		00008190
C	CALCULATE W(TRANSPPOSE=CONJUGATE) * M * W = SIR DENOMINATOR (REAL)	00008200
C	SINCE M IS HERMITIAN	00008210
C		00008220
C	SIRDEN = 0.0	00008230
C		00008240
C	DO 500 K=1,N	00008250
C	SIRDEN = SIRDEN + WTCONM(K)*BCONC(K)	00008260
C	500 CONTINUE	00008270
C		00008280
C	CALCULATE SIR(CURRENT LOWER BOUND) (REAL) SINCE M HERMITIAN	00008290
C		00008300
C	PSIMAX = SIRNUM/SIRDEN	00008310
C		00008320
C	RETURN	00008330
C	DEBUG SUBCHK	00008340
C	END	00008350
C		00008360
C	SUBROUTINE ZORN(R,BN1,NP1)	00008370
C		00008380
C	HERMITIAN MATRIX TODEPLITZ INVERSION (COMPLEX)	00008390
C		00008400
C	COMPLEX*16 X(201),BN1(NP1,NP1),R(100),RG,GAI,UGIJ	00008410
C	N=NP1-1	00008420
C	N2P1=2*N+1	00008430
C	XL1=1.-CDABS(R(1))*2	00008440
C	X(N)=R(1)	00008450
C	X(N+2)=DCONJG(X(N))	00008460
C	NM1=N-1	00008470
C	DO 3 I=1,NM1	00008480
C	RG=0.	00008490
C	DO 4 K=1,I	
C	L=I-K+1	
C	4 RG=R(K)*X(NP1-L)+RG	

```

IP1=I+1
GAI=-(R(IP1)+RG)
X(NP1-IP1)=GAI/XLI
DO 5 J=1,I
L=I-J+1
5 X(NP1-J)=X(NP1-J)+X(NP1+L)*GAI/XLI
DO 6 J=1,IP1
6 X(NP1+J)=DCONJG(X(NP1-J))
3 XLI=XLI-(COABS(GAI)*2)/XLI
X(NP1)=XLI
X(NP1)=DCONJG(X(1))
BN1(1,1)=1/X(N+1)
DO 20 I=1,N
J=NP1-I
BN1(I+1,1)=X(J)/X(N+1)
20 BN1(I,1)=DCONJG(BN1(I+1,1))
DO 22 I=1,NM1
IM1=I-1
DO 22 J=1,I
UGIJ=X(N+1-I)*X(N+1+J)-X(2*N+2-I)*X(J)
BN1(I+1,J+1)=BN1(I,J)+UGIJ/X(N+1)
22 BN1(J+1,I+1)=DCONJG(BN1(I+1,J+1))
DO 24 I=2,NP1
J=NP1-I
BN1(J,I)=BN1(J+1-I,1)
24 BN1(I,J)=DCONJG(BN1(J,I))
RETURN
DEBUC SUBCHK
END

```

```

C
SUBROUTINE ARFACT(N,WCLR,WOPT,WQ,THETAJ,THETAS)
REAL FTP8(181),FTPT(181),FTPQ(181),R,LAMBDA
REAL*8 GDRB,GDRJ,GDRQ
COMPLEX*16 SUMB,SUMT,SUMQ,T1B,T1T,T1Q
COMPLEX*16 WCLR(N),WOPT(N),WQ(N)
LAMBDA=1.18029
PI=3.141592654
RD=180.0/PI
DR=1.0/RD
K=2.0*PI/LAMBDA

```

```

C
IPH=0
PHD=0.0
PH=0.0
FMAXB=0.0
FMAXT=0.0
FMAXQ=0.0
DO 10 J=1,181
JJ=J-1
THD=FLOAT(JJ)
TH=THD*DR

```

```

C
SUMB=(0.0,0.0,0.0)
SUMT=(0.0,0.0,0.0)
SUMQ=(0.0,0.0,0.0)

```

```

00008500
00008510
00008520
00008530
00008540
00008550
00008560
00008570
00008580
00008590
00008600
00008610
00008620
00008630
00008640
00008650
00008660
00008670
00008680
00008690
00008700
00008710
00008720
00008730
00008740
00008750
00008760
00008765
00008770
00008775
00008778
00008780
00008785
00008790
00008795
00008800
00008810
00008820
00008830
00008840
00008860
00008880
00008890
00008895
00008900
00008903
00008907
00008910
00008920
00008930
00008940
00008950
00008960
00008965
00008970

```

```

CT=COS(TH)
DO 15 N1=1,N
NN=NN1+1
T1B=NC(LB(N1))*(COS(PI*NN*CT)+(0.,1.)*SIN(PI*NN*CT))
T1T=WC(T(N1))*(COS(PI*NN*CT)+(0.,1.)*SIN(PI*NN*CT))
T1Q=WQ(N1)*(COS(PI*NN*CT)+(0.,1.)*SIN(PI*NN*CT))
SUMT = SUMT + T1T
SUMQ = SUMQ + T1Q
15 SUMB=SUMB+T1B
IF(THD.NE.(90.0-THETAJ)) GO TO 34
GOBR=20.0*DLOG10(CDARS(SUMB))
GDBT=20.0*DLOG10(CDARS(SUMT))
GDBQ=20.0*DLOG10(CDARS(SUMQ))
WRITE (6,*) THETAJ,GOBR,GDBT,GDBQ
34 IF(THD.NE.(90.0-THETAS)) GO TO 35
GOBR=20.0*DLOG10(CDARS(SUMB))
GDBT=20.0*DLOG10(CDARS(SUMT))
GDBQ=20.0*DLOG10(CDARS(SUMQ))
WRITE (6,*) THETAS,GOBR,GDBT,GDBQ
35 CONTINUE

C
FTP(B(J))=CDARS(SUMB)
FTP(T(J))=CDARS(SUMT)
FTP(Q(J))=CDARS(SUMQ)
FMAXB=AMAX1(FMAXB,FTP(B(J)))
FMAXT=AMAX1(FMAXT,FTP(T(J)))
FMAXQ=AMAX1(FMAXQ,FTP(Q(J)))
10 CONTINUE
FMXN = AMAX1(FMAXB,FMAXT)
FMXN = AMAX1(FMXN,FMAXQ)
WRITE (6,2) FMAXB,FMAXT,FMAXQ,FMXN
2 FORMAT (1H, 'FMAXB=',F12.4,1X, 'FMAXT=',F12.4,1X, 'FMAXQ=',F12.4,
+ 1X, 'FMXN=',F12.4)
CALL APL0T(FTP(B),FMXN,FTP(T),FMXN,FTP(Q),FMXN,PHD)
5 CONTINUE
RETURN
DEBUG SUBCHK
END

C
SUBROUTINE APL0T(YB,YBMAX,YT,YTMAX,YQ,YQMAX,PHD)
REAL YB(361),YT(361),YQ(361),THD(181),FB(181),FT(181),FQ(181)
CALL PLOT1
DO 10 J=1,181
THD(J)=FL0AT(J-91)
IF(YB(J)) 8,12,8
8 FB(182-J)=20.0*ALOG10(YB(J)/YBMAX)
IF(FB(182-J)+70.0) 12,12,10
12 FB(182-J)=-70.0
10 CONTINUE
DO 20 J=1,181
IF(YT(J)) 16,24,16
16 FT(182-J)=20.0*ALOG10(YT(J)/YTMAX)
IF(FT(182-J)+70.0) 24,24,20
24 FT(182-J)=-70.0
20 CONTINUE

```

```

00008980
00008990
00009000
00009010
00009011
00009012
00009015
00009017
00009020
00009043
00009044
00009045
00009046
00009047
00009048
00009050
00009051
00009052
00009053
00009058
00009060
00009070
00009072
00009074
00009080
00009082
00009084
00009090
00009095
00009098
00009100
00009110
00009111
00009115
00009120
00009130
00009135
00009140
00009150
00009160
00009170
00009175
00009190
00009200
00009210
00009220
00009230
00009240
00009240
00009241
00009243
00009244
00009245
00009246
00009287

```

```

DO 100 J=1,181
  IF(YQ(J)) 80,120,80
80  FQ(182-J)=20.0+ALOG10(YQ(J)/YQMAX)
  IF(FQ(182-J)+70.0) 120,120,100
120  FQ(182-J)=-70.0
100  CONTINUE
  CALL CPLOT2(3,THD,10.0,'THETA (DEGREES)',15,7.0,'GAIN (DB)',
+  9,181,FB,181,FT,181,FQ)
  CALL CPLOT2(1,THD,10.0,'THETA (DEGREES)',15,7.0,'GAIN (DB)',
+  9,181,FB)
  CALL CPLOT2(1,THD,10.0,'THETA (DEGREES)',15,7.0,'GAIN (DB)',
+  9,181,FT)
  CALL CPLOT2(1,THD,10.0,'THETA (DEGREES)',15,7.0,'GAIN (DB)',
+  9,181,FQ)
1000 RETURN
      DEBUG SUBCHK
      END

```

```

00009291
00009293
00009294
00009295
00009296
00009297
00009320
00009321
00009324
00009325
00009327
00009328
00009330
00009331
00009336
00009400
00009401

```