AD-A240 667

SSD-TR-91-27

AEROSPACE REPORT NO.
TR-0091(6925-05)-3

DTIC
ELECTE
SEP 1 9 1991
S D

A Conjugate Gradient Based Algorithm to Minimize the Sidelobe Level of Planar Arrays with Element Failures

Prepared by

T. J. PETERS
Communications Systems Subdivision

31 August 1991

Prepared for

SPACE SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960

Engineering and Technology Group

91-11030

THE AEROSPACE CORPORATION
El Segundo, California

91 9 18 085

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Systems Division, P. O. Box 92960, Los Angeles, CA 90009-2960. It was reviewed and approved for The Aerospace Corporation by J. M. Straus, Principal Director, Communications Systems Subdivision.

Seth Parkoff, Capt, USAF, was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

SETH PARKOFF, Capt, USAF
MOIE Project Officer
SSD/MHE

JONATHAN M. EMMES, Maj, USAF
MOIE Program Manager
PL/WCO OL-AH

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-0091(6925-05)-3	5. MONITORING ORGANIZATION REPORT NUMBER(S) SSD-TR-91-27
6a. NAME OF PERFORMING ORGANIZATION The Aerospace Corporation Communications Systems Subdivision 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Space Systems Division
6c. ADDRESS (City, State, and ZIP Code) P. O. Box 92957 Los Angeles, CA 90009-2957	7b. ADDRESS (City, State, and ZIP Code) Los Angeles Air Force Base P. O. Box 92960 Los Angeles, CA 90009-2960
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F04701-88-C-0089
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Conjugate Gradient Based Algorithm to Minimize the Sidelobe Level of Planar Arrays with Element Failures

12. PERSONAL AUTHOR(S)
Peters, T. J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM _____ TO _____	31 August 1991	83

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Antenna array synthesis Conjugate gradient method

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

Element failures increase the sidelobe power level of an array. Reconfiguring the amplitude and phase of the remaining elements can partially compensate for the failed elements and thus reduce the sidelobe level. The result of this investigation is an algorithm that yields the reconfigured distribution by minimizing the ratio of the average peak sidelobe power level to the power in the mainbeam, taking into account the defective elements. The minimization of this nonlinear function is carried out via a conjugate gradient method. The algorithm is applied to the synthesis of sum and difference patterns of planar arrays.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

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

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Preface

The author would like to thank Dr. Keith M. Soo Hoo, Dr. Robert B. Dybdal and Don J. Hinshilwood for their helpful comments.

A-1

Contents

Preface 1
I. Introduction 7
II. Power Pattern With Element Failures 9
III. Minimizing Nonlinear Functions 13
IV. Sum Pattern Synthesis 19
V. Difference Pattern Synthesis 25
VI. Results and Discussion 27
VII. Conclusion 37
References 39
Appendix: Fortran 77 Programs 41

Figures

1. Independently Controlled Elements for (a) Sum Pattern and (b) Difference Pattern 10
2. One-Dimensional Minimization 14
3. One-Dimensional Average Sidelobe Accumulation 21
4. Sum Pattern Sidelobe Sample Region 22
5. Difference Pattern Sidelobe Sample Region 26
6. Hexagonal 91 Element Array with 3 Element Failures 28
7. Sum Power Pattern with No Failures 29
8. Sum Power Pattern with 3 Element Failures 30
9. Synthesized Sum Power Pattern with 3 Element Failures 31
10. Difference Power Pattern with No Failures 32
11. Difference Power Pattern with 3 Element Failures 33
12. Synthesized Difference Power Pattern with 3 Element Failures 34

I. Introduction

An antenna array is composed of many elements whose excitation amplitude and phase can be individually adjusted to yield a desired pattern. If all array elements operate properly, then well-known analytic techniques can be used to find the optimum amplitude and phase of each element, to yield a given beamwidth and sidelobe level. However, if any elements fail, then no analytic means exists to find the aperture distribution that compensates for the degradation of the pattern. Element failures destroy symmetry and cause sharp variations in the field intensity across the array aperture, increasing the sidelobe level of the power pattern. The simplest solution to this problem is to increase the taper of the array distribution in order to lower the sidelobes back to the design level. Unfortunately, this solution has several disadvantages. First, as the design sidelobe level for an array decreases, the sensitivity of the sidelobe level to a perturbation in an excitation value increases. Second, changing the design taper may broaden the main beam to an unacceptable level. Finally, merely increasing the taper does not compensate for lack of symmetry or smooth out sharp field variations. Compensation for the defective elements can be achieved, by numerically finding the excitation of each nondefective element that optimizes some function. This function must take into account the location of the failed elements.

The approach used in this study is to reformulate the optimization problem to take into account the element failures. An algorithm is developed that synthesizes the amplitude and phase of each nondefective element in order to produce the lowest sidelobe level for a given beamwidth. This algorithm can also be used to find the element failure limits of any array. More specifically, the algorithm can find the maximum number of elements beyond which it is impossible to recover lost performance. Also, the loss in performance can be determined by the geometric arrangement of the failed elements. The numerical implementation of the synthesis algorithm involves the minimiza-

tion of a nonlinear function, which is the ratio of the sum of a set of peak sidelobe power points to the power in the main beam. This minimization will be carried out via a conjugate gradient method.

II. Power Pattern With Element Failures

This analysis neglects the element factor and considers only the array factor. No mutual coupling effects are modeled. Failed elements are deleted from the array factor. The $e^{j\omega t}$ time convention is understood, where ω is the angular frequency of the excitation with a freespace wavelength of λ_0. Letting N denote the number of nonfailed elements, the array factor (AF) is given by

$$\mathrm{AF} = \sum_{n=1}^{N} I_n e^{j(k_x x_n + k_y y_n)} \tag{1}$$

where

$$k_x = k_0 \sin(\theta)\cos(\phi) \tag{2}$$

$$k_y = k_0 \sin(\theta)\sin(\phi) \tag{3}$$

and $k_0 = 2\pi/\lambda_0$. The excitation of the nth element, denoted by I_n, can be expressed by the complex quantity $I_n = u_n + jv_n$, where $\sqrt{u_n^2 + v_n^2}$ yields the amplitude and $\tan^{-1}(v_n/u_n)$ yields the phase angle. The power pattern is proportional to the square of the magnitude of AF.

The array is assumed to operate in either sum or difference mode. The sum mode produces a peak in the pattern in the broadside direction and the difference mode produces a null in the broadside direction. A possible element configuration to generate both types of patterns is shown in Fig. 1. Sum mode operation allows all 37 elements to be independently controlled, as shown in Fig. 1(a). The same array operating in difference mode has only 17 independently controlled elements, as shown in Fig. 1(b). This is because half of the array elements are excited with their phase opposite from that of the corresponding elements symmetric with respect to the center of the array. If an element fails, then the symmetric counterpart is deactivated. This ensures a stationary beam null and a symmetric beam slope, independent of the synthesis technique used.

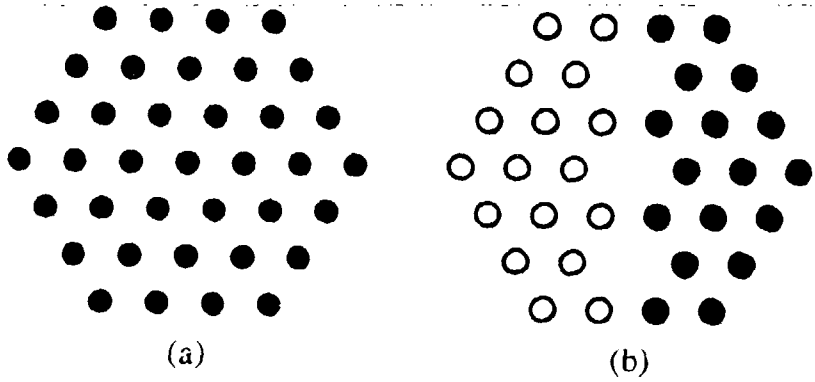

Fig. 1. Independently Controlled Elements for (a) Sum Pattern and (b) Difference Pattern

The synthesis of the excitation values has three possible combinations: synthesizing with amplitude only, synthesizing with phase only, and synthesizing with both amplitude and phase. It is well known that synthesizing the lowest possible sidelobe level for a given beamwidth over a symmetric aperture yields a uniform phase and a nonuniform amplitude distribution. However, if the failed elements yield a nonsymmetric array distribution, then the optimal phase may not be constant. Phase synthesis with a constant amplitude was found by Deford and Gandhi (Ref. 1) to require an extremely large number of elements to yield low sidelobes. A similar analysis by Peters (Ref. 2) found that phase-only synthesis was not very effective in reducing sidelobes for fixed-amplitude arrays. Therefore, both amplitude and phase synthesis would seem to be required for the generation of a sum pattern. Since the difference pattern preserves the symmetry, amplitude-only synthesis is required. The sum power pattern is given by

$$P(\theta, \phi, u, v) = \left[\sum_{n=1}^{N} [u_n \cos(\psi_n) - v_n \sin(\psi_n)]\right]^2 + \left[\sum_{n=1}^{N} [u_n \sin(\psi_n) + v_n \cos(\psi_n)]\right]^2 \quad (4)$$

where $\psi_n = k_x x_n + k_y y_n$ and the difference power pattern may be written as

$$P(\theta, \phi, u) = \left[\sum_{n=1}^{N} u_n \sin(\psi_n)\right]^2 . \quad (5)$$

Note that the amplitude is allowed to be negative on the difference pattern, since that involves only a simple phase shift.

III. Minimizing Nonlinear Functions

Equations (4) and (5) indicate that the power pattern of the array is a nonlinear function of the amplitude and phase of the excitation. Minimizing the peak sidelobe level will require a mathematical technique that can handle an nonlinear function. A multivariable quadratic function can be minimized by the conjugate gradient algorithm developed by Hestenes and Stiefel (Ref. 3). An arbitrary nonlinear function can also be minimized by the same algorithm if a suitable quadratic approximation can be found. Although several variations are given in the literature, all have the same two major steps discussed by Hestenes (Ref. 4). First, the function must be approximated by a quadratic truncation of a Taylor series expanded around an initial estimate of the solution. Second, this quadratic is minimized by the conjugate gradient algorithm to obtain a new estimate. This new estimate is in turn used as the expansion point for a new quadratic approximation. Since the quadratic approximation is valid only near the point of expansion, finding a minimum of this quadratic is the same as finding a minimum of the original function only if both have the same minimum. Therefore, the algorithm must be restarted so that the quadratic approximation can ultimately be expanded around a point that approaches the minimum of the original function. The one-dimensional case is illustrated in Fig. 2. Given the function $f(s)$ and the initial guess s_0, the quadratic $q_0(s)$ is formed and given by

$$q_0(s) = f(s_0) + f'(s_0)(s - s_0) + \frac{1}{2} f''(s_0)(s - s_0)^2. \tag{6}$$

The minimum of this quadratic is denoted by s_1 and is given by the Newton estimate

$$s_1 = s_0 - \frac{f'(s_0)}{f''(s_0)}. \tag{7}$$

The estimate s_1 is then used to compute $q_1(s)$. The iteration then continues until the minimum of the quadratic approximation coincides with the minimum of the original function.

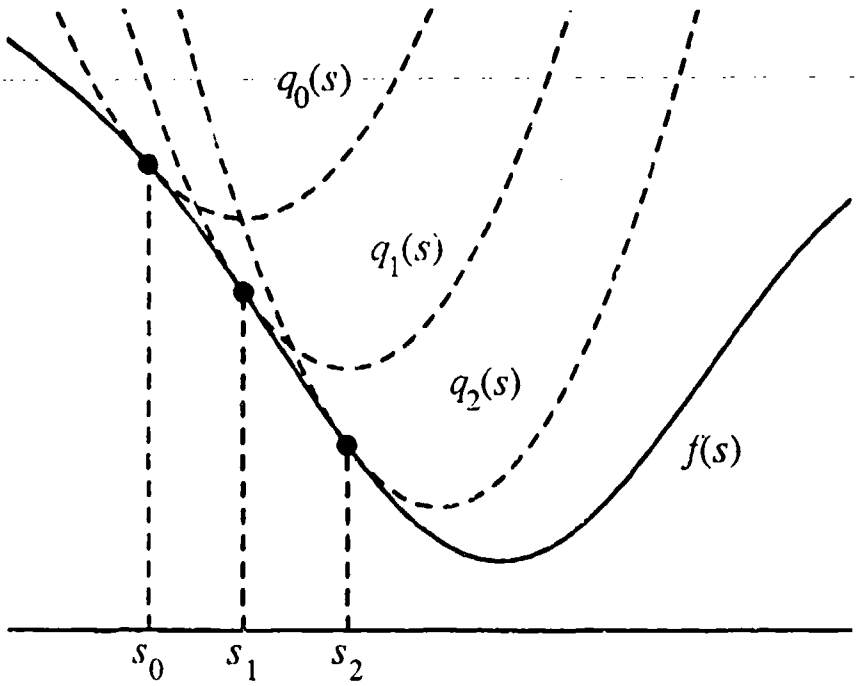

Fig. 2. One-Dimensional Minimization

The principles of the one-dimensional minimization apply also to multidimensional functions. For example, consider the function $f(u, v)$, where u and v are $N \times 1$ column vector variables. The quadratic approximation may be written in matrix form as

$$f(u, v) = f(u^1, v^1) + [\gamma - \gamma^1]^T G + \frac{1}{2}[\gamma - \gamma^1]^T H [\gamma - \gamma^1] \tag{8}$$

where T denotes the transpose and the components are defined as follows. The vector γ is a $2N \times 1$ column vector defined by $\gamma = [u \quad v]^T$. The gradient, denoted by G, is a $2N \times 1$ column vector evaluated at (u^1, v^1) and is given by $G = [G_u \quad G_v]^T$, where each subvector has the form

$$G_u = [\, \frac{\partial f}{\partial u_1} \quad \frac{\partial f}{\partial u_2} \quad \cdots \quad \frac{\partial f}{\partial u_N} \,]. \tag{9}$$

The Hessian, denoted by H, is a $2N \times 2N$ matrix whose elements are evaluated at (u^1, v^1) and expressed as

$$H = \begin{bmatrix} H_{uu} & H_{uv} \\ H_{vu} & H_{vv} \end{bmatrix} \tag{10}$$

where each submatrix has the form

$$H_{uv} = H_{vu}^T = \begin{bmatrix} \frac{\partial^2 f}{\partial u_1 \partial v_1} & \frac{\partial^2 f}{\partial u_1 \partial v_2} & \cdots & \frac{\partial^2 f}{\partial u_1 \partial v_N} \\ \frac{\partial^2 f}{\partial u_2 \partial v_1} & \frac{\partial^2 f}{\partial u_2 \partial v_2} & \cdots & \frac{\partial^2 f}{\partial u_2 \partial v_N} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial u_N \partial v_1} & \frac{\partial^2 f}{\partial u_N \partial v_2} & \cdots & \frac{\partial^2 f}{\partial u_N \partial v_N} \end{bmatrix}. \tag{11}$$

Using the one-dimensional analogy, the minimum of Eqn. (8) may be computed from the matrix form of Newton's formula, as given by

$$\gamma_1 = \gamma_0 + H^{-1}(\gamma_0) G(\gamma_0). \tag{12}$$

However, this requires that the inverse of H be computed. In many cases of practical interest, only an estimate of the minimum of the quadratic is required; therefore, it is more efficient to use the conjugate gradient method to obtain an estimate of this minimum. Making the substitution $z_u = u - u^1$,

$z_v = v - v^1$, and $D = -G$ into Eqn. (8) and excluding the constant yields a standard quadratic function, denoted by c and written as

$$c = \frac{1}{2}\begin{bmatrix} z_u & z_v \end{bmatrix}\begin{bmatrix} H_{uu} & H_{uv} \\ H_{vu} & H_{vv} \end{bmatrix}\begin{bmatrix} z_u \\ z_v \end{bmatrix} - \begin{bmatrix} z_u & z_v \end{bmatrix}\begin{bmatrix} D_u \\ D_v \end{bmatrix}. \tag{13}$$

Minimizing this expression is the same as solving the linear system

$$\begin{bmatrix} H_{uu} & H_{uv} \\ H_{vu} & H_{vv} \end{bmatrix}\begin{bmatrix} z_u \\ z_v \end{bmatrix} = \begin{bmatrix} D_u \\ D_v \end{bmatrix} \tag{14}$$

where the Hessian matrix H is always symmetric and nonsingular but not always positive definite. Therefore, the conjugate gradient algorithm must be general enough to handle this situation. A suitable algorithm with L restarts and $K << N$ conjugate gradient iterations per restart is given as follows (Ref. 5):

begin loop for $l=1,\ldots,L$

compute D^l, H^l *and set* $z_u^1 = 0, z_v^1 = 0$

$$r_{u,v}^1 = D_{u,v}^l \tag{15}$$

$$q_u = H_{uu}r_u^1 + H_{uv}r_v^1 \tag{16}$$

$$q_v = H_{vu}r_u^1 + H_{vv}r_v^1 \tag{17}$$

$$\beta_0 = \frac{1}{\langle q_u, q_u\rangle + \langle q_v, q_v\rangle} \tag{18}$$

$$p_{u,v}^1 = \beta_0 q_{u,v} \tag{19}$$

begin loop for $k=1,\ldots,K$

$$q_u = H_{uu}p_u^k + H_{uv}p_v^k \tag{20}$$

$$q_v = H_{vu}p_u^k + H_{vv}p_v^k \tag{21}$$

$$\alpha_k = \frac{1}{\langle q_u, q_u\rangle + \langle q_v, q_v\rangle} \tag{22}$$

$$z_{u,v}^{k+1} = z_{u,v}^k + \alpha_k p_{u,v}^k \tag{23}$$

$$r_{u,v}^{k+1} = r_{u,v}^k - \alpha_k q_{u,v} \tag{24}$$

$$err = \frac{\langle r_u^{k+1}, r_u^{k+1}\rangle + \langle r_v^{k+1}, r_v^{k+1}\rangle}{\langle r_u^k, r_u^k\rangle + \langle r_v^k, r_v^k\rangle} \tag{25}$$

if err < tolerance, terminate k loop, else

$$q_u = H_{uu} r_u^{k+1} + H_{uv} r_v^{k+1} \tag{26}$$

$$q_v = H_{vu} r_u^{k+1} + H_{vv} r_v^{k+1} \tag{27}$$

$$\beta_k = \frac{1}{\langle q_u, q_u \rangle + \langle q_v, q_v \rangle} \tag{28}$$

$$p_{u,v}^{k+1} = p_{u,v}^{k} + \beta_k q_{u,v} \tag{29}$$

continue k loop

$$u, v^{l+1} = u, v^{l} + z_{u,v}^{k+1} \tag{30}$$

continue l loop

The l loop determines the quadratic approximation to the function $f(u, v)$. The gradient and Hessian are computed at the current estimate and the appropriate conjugate gradient parameters are initialized. The k loop obtains an estimate of the minimum of c by computing an estimate of the solution to the linear system described by Eqn. (14). The dominant computational effort for each k iteration is caused by the matrix vector products, which require $\mathcal{O}(8N^2)$ multiplications. The computation of the Hessian requires $\mathcal{O}(\frac{1}{2}TN^2)$ multiplications, where T is the number of multiplications per matrix element. Since T depends on the function, the computational efficiency of the overall algorithm is function-dependent. An alternate algorithm used by Fong and Birgenheier (Ref. 6) and Press *et al.* (Ref. 7) does not require the computation of the Hessian, but implicitly assumes that it is positive definite. This can lead to an unpredictable breakdown in the algorithm.

IV. Sum Pattern Synthesis

The goal of the sum pattern synthesis is to maximize the power in the main beam direction while simultaneously minimizing the power in the peak sidelobe. Unfortunately, forcing the sidelobe to a lower level at a particular angular location without any constraint on the other sidelobes will merely shift the peak sidelobe to a new location. Because the peak sidelobe location may change, there is no mathematically elegant way to achieve the goal directly. Rather, the same objective may be achieved indirectly, by minimizing the average of the accumulated peak sidelobe power points. The average is defined to be the average of these peak sidelobes, computed at each restart of the minimization algorithm. If M_l denotes the number of distinct peak sidelobe points at the lth restart, then the average power, denoted by $P_a(u,v)$, is defined by

$$P_a(u,v) = \frac{1}{M_l}\sum_{i=1}^{M_l} P_i. \tag{31}$$

The power at the ith angular point is defined as $P_i = P(u,v,\theta_i,\phi_i)$, where $i = 0$ denotes the main beam location and $i = 1...M$ denotes peak sidelobe points. In order to minimize the sidelobe power while simultaneously maximizing the main beam power, a function $f(u,v)$ is defined as the ratio of the average peak power to the main beam power and is expressed as

$$f(u,v) = \frac{P_a(u,v)}{P_0(u,v)}. \tag{32}$$

Minimizing this function will achieve the desired goal. A possible one-dimensional scenario is illustrated in Figure 3. The peak sidelobe level is found at the initial start. This sidelobe level is then reduced by a partial minimization of $f(u,v)$. At the first restart the peak sidelobe point is still at the same location, so that peak is reduced again. At the second restart a new peak sidelobe location is found, so this peak and the previous peak are averaged and reduced. At the third restart the peak location has changed

again, so the average is updated and $f(u,v)$ is minimized. At the fourth restart another point is added to the average and $f(u,v)$ is minimized. The pattern has reached the steady-state power distribution.

An inherent characteristic of this method is that all the sidelobes will tend to be brought to the same level. It should be noted that all the sidelobe sample points could have been used a priori at each restart without searching for the peak sidelobe point and the results would be the same. However, using the accumulated average has several advantages. The function $f(u,v)$ changes rapidly when few sidelobe points are present, so that the peak may be brought down fast. As more peak points are added to the average, a self-damping effect takes place and reduces the sensitivity of the pattern to changes in u and v, allowing a smooth transition to an optimum. Accumulating the peak points is numerically much faster than starting with all points, since the computation time for evaluating $f(u,v)$ is dependent on M_l.

The sidelobe region is defined as the hemisphere formed by the limits $0 \leq \phi \leq 2\pi$ and $0 \leq \theta \leq \pi$ minus the circular aperture of the main beam up to the first null. The density of the sample points should be chosen in such a way that the peak sidelobe may be accurately estimated. A good rule of thumb is that the sample separation distance should be no greater than half the main beamwidth of the design pattern. A uniformly spaced square lattice is sufficient and is shown in Fig. 4. The parameter k_m is the radius of the main beam null of the design pattern, increased slightly to allow the beam to broaden as the sidelobe level is reduced.

The components of D and H required for the conjugate gradient method are given as follows:

$$D_u^m = -\frac{\partial f}{\partial u_m} = \frac{1}{P_0}\left[\frac{P_a}{P_0}\frac{\partial P_0}{\partial u_m} - \frac{\partial P_a}{\partial u_m}\right] \tag{33}$$

$$D_v^m = -\frac{\partial f}{\partial v_m} = \frac{1}{P_0}\left[\frac{P_a}{P_0}\frac{\partial P_0}{\partial v_m} - \frac{\partial P_a}{\partial v_m}\right] \tag{34}$$

$$H_{uu}^{mn} = \frac{\partial^2 f}{\partial u_m \partial u_n} = \frac{1}{P_0}\left[D_u^n \frac{\partial P_0}{\partial u_m} + D_u^m \frac{\partial P_0}{\partial u_n} + \frac{\partial^2 P_a}{\partial u_m \partial u_n} - \frac{P_a}{P_0}\frac{\partial^2 P_0}{\partial u_m \partial u_n}\right] \tag{35}$$

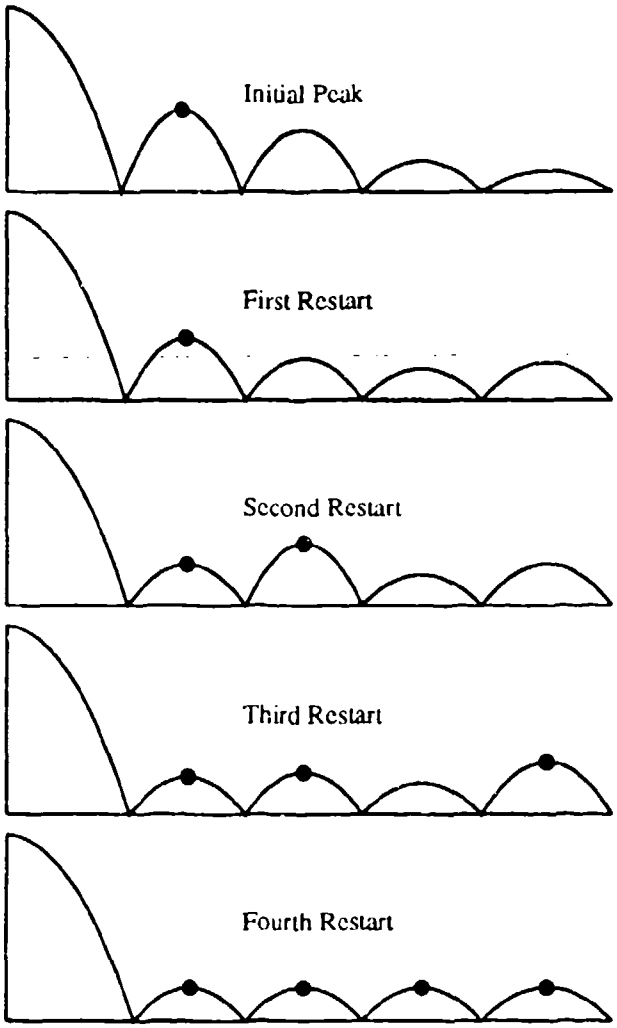

Fig. 3. One-Dimensional Average Sidelobe Accumulation

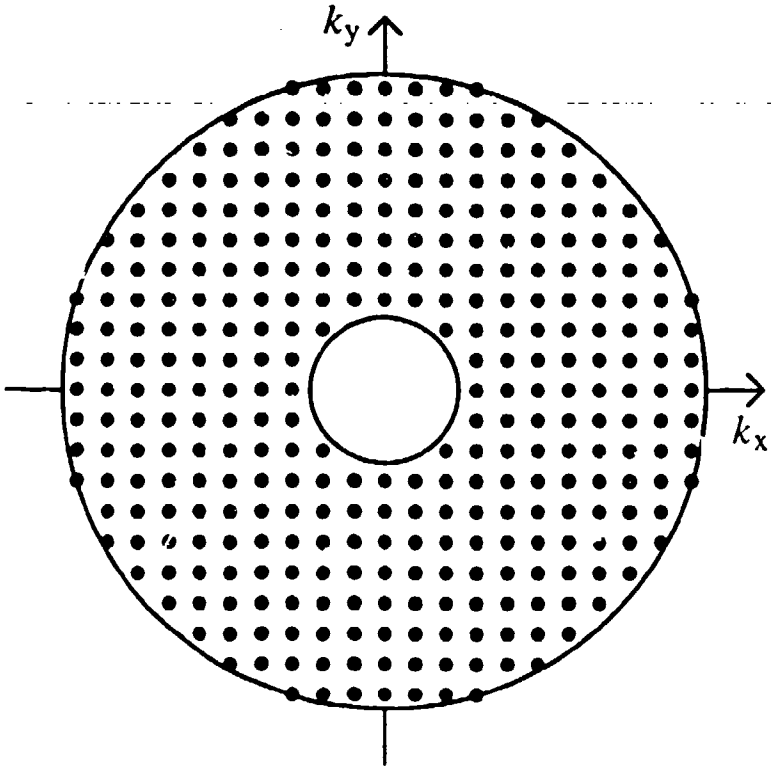

Fig. 4. Sum Pattern Sidelobe Sample Region

$$H_{uv}^{mn} = \frac{\partial^2 f}{\partial u_m \partial v_n} = \frac{1}{P_0}\left[D_v^n \frac{\partial P_0}{\partial u_m} + D_u^m \frac{\partial P_0}{\partial v_n} + \frac{\partial^2 P_a}{\partial u_m \partial v_n} - \frac{P_a}{P_0}\frac{\partial^2 P_0}{\partial u_m \partial v_n}\right] \tag{36}$$

$$H_{vu}^{mn} = H_{uv}^{mn} \tag{37}$$

$$H_{vv}^{mn} = \frac{\partial^2 f}{\partial v_m \partial v_n} = \frac{1}{P_0}\left[D_v^n \frac{\partial P_0}{\partial v_m} + D_v^m \frac{\partial P_0}{\partial v_n} + \frac{\partial^2 P_a}{\partial v_m \partial v_n} - \frac{P_a}{P_0}\frac{\partial^2 P_0}{\partial v_m \partial v_n}\right] \tag{38}$$

The power at the ith angle can be computed by introducing the auxiliary functions ξ_i and χ_i and defining $P_i = \xi_i^2 + \chi_i^2$ such that

$$\xi_i = \sum_{w=1}^{N}[u_w \cos(\psi_{wi}) - v_w \sin(\psi_{wi})] \tag{39}$$

$$\chi_i = \sum_{w=1}^{N}[u_w \sin(\psi_{wi}) + v_w \cos(\psi_{wi})] \tag{40}$$

where $\psi_{wi} = k_{xi}x_w + k_{yi}y_w$. The first-derivatives of the power used in the gradient are given by

$$\frac{\partial P_i}{\partial u_m} = 2\left[\xi_i \cos(\psi_{mi}) + \chi_i \sin(\psi_{mi})\right] \tag{41}$$

$$\frac{\partial P_i}{\partial v_m} = 2\left[\chi_i \cos(\psi_{mi}) - \xi_i \sin(\psi_{mi})\right] \tag{42}$$

and the second derivative terms used in the Hessian are expressed as

$$\frac{\partial^2 P_i}{\partial u_m \partial u_n} = 2\cos(\psi_{mi} - \psi_{ni}) \tag{43}$$

$$\frac{\partial^2 P_i}{\partial u_m \partial v_n} = 2\sin(\psi_{mi} - \psi_{ni}) \tag{44}$$

$$\frac{\partial^2 P_i}{\partial v_m \partial v_n} = \frac{\partial^2 P_i}{\partial u_m \partial u_n}. \tag{45}$$

Note that the computation of the Hessian requires the evaluation of the power at the main beam location and at each peak sidelobe point. Since the power computation requires $\mathcal{O}(N)$ multiplications, the multiplications per matrix element is $T = \mathcal{O}(M_l N)$. Therefore, the Hessian computation dominates the computation of a single l iteration.

V. Difference Pattern Synthesis

The synthesis procedure for the difference pattern is similar to that used for the sum pattern, except that now only amplitude synthesis is necessary. The appropriate function defined in Eqn. (32) is reduced to one dimension as

$$f(u) = \frac{P_a(u)}{P_0(u)}. \tag{46}$$

The power at the ith location is defined as $P_i = \xi_i^2$, where

$$\xi_i = \sum_{w=1}^{N} u_w \sin(\psi_{wi}). \tag{47}$$

The first derivative of the power is computed from

$$\frac{\partial P_i}{\partial u_m} = 2\xi_i \sin(\psi_{mi}) \tag{48}$$

with the second derivative given by

$$\frac{\partial^2 P_i}{\partial u_m \partial u_n} = 2\sin(\psi_{mi})\sin(\psi_{ni}). \tag{49}$$

For the configuration shown in Fig. 1, the power pattern is symmetric across the $y - z$ plane. Thus, the k-space sample region need only contain points where $k_z > 0$, as shown in Fig. 5.

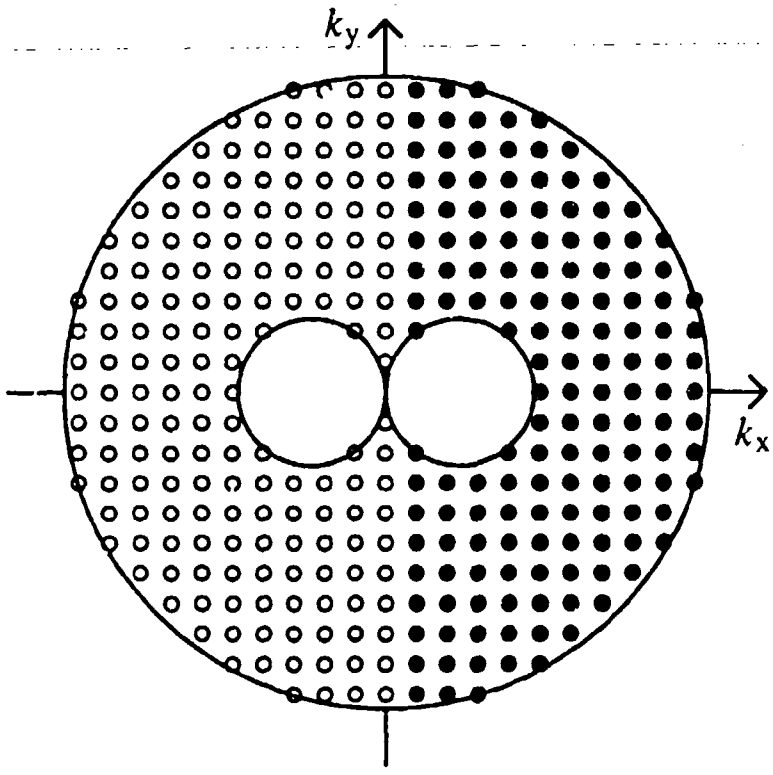

Fig. 5. Difference Pattern Sidelobe Sample Region

VI. Results and Discussion

A possible element failure scenario is shown in Fig. 6. The hexagonal array has a total of 91 elements spaced $0.6\lambda_0$ apart with 3 element failures. The desired aperture function for the sum pattern is a parabolic with power $N = 2$ and an edge taper of -20 dB. This yields a -35 db sidelobe level. The aperture function for the difference pattern is a Bayliss distribution with $\bar{n} = 10$ and a -40 dB sidelobe level. All the power patterns are plotted in dB using cylindrical coordinates, with θ as the radial coordinate in the range $0 \leq \theta \leq 90°$. A reference -32 dB floor is placed on each plot.

The sum power pattern with no element failures is shown in Fig. 7. The peak sidelobe level is -30.9 dB with a peak gain of 20.8 dB. The main beam has azimuthal symmetry with a half-power beamwidth of 12.86°. The synthesized sum pattern of Fig. 8 required 300 restarts, with 5 conjugate gradient iterations per restart, to solve for the 176 required unknowns. The effect of the failures is to increase the sidelobe level to -27.4 dB and reduce the gain to 20.5 dB. The main beam becomes slightly elliptical with a minimum half-power beamwidth of 12.57° and a maximum of 13.00°. The synthesized pattern, shown in Fig. 9, has a peak sidelobe level of -31.3 dB and a gain of 19.5 dB. The main beam also has an elliptical cross section with a minimum half-power beamwidth of 13.71° and a maximum of 16.29°. As expected, the reduction in the sidelobe level produces a corresponding increase in the beamwidth and a drop in gain.

The difference power pattern with no element failures is shown in Fig. 10. The peak sidelobe level is -33.0 dB with a peak gain of 18.4 dB. The angular half-power null width between the peaks is 8.57°. Fig. 11 shows that the effect of the element failures is to increase the sidelobe level to -21.7 dB and drop the gain to 18.0 dB, with the half-power null width remaining about the same. The synthesized difference pattern of Fig. 12 required 100 restarts with 5 conjugate gradient iterations per restart to solve for the 80 unknowns.

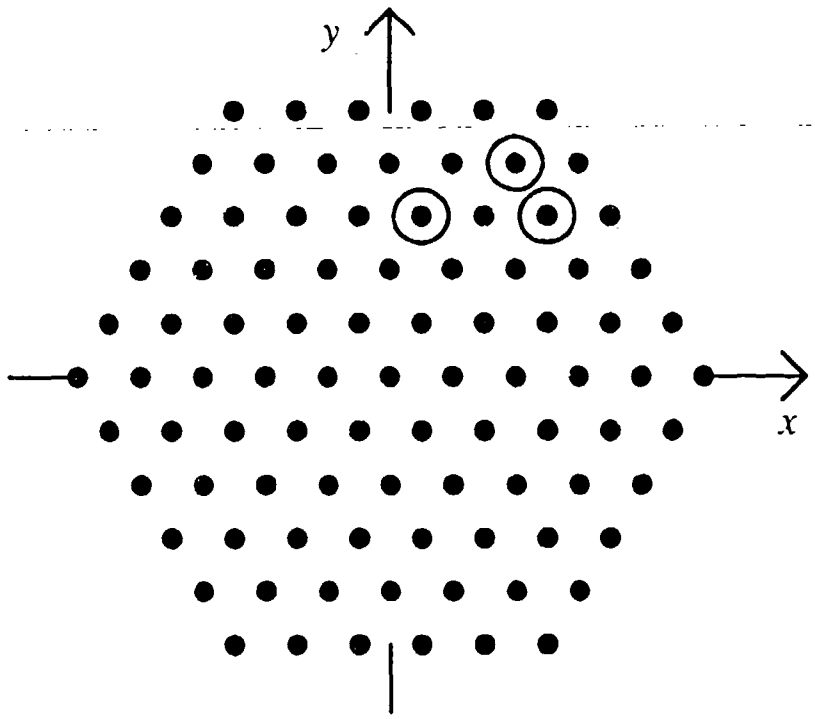

Fig. 6. Hexagonal 91 Element Array with 3 Element Failures

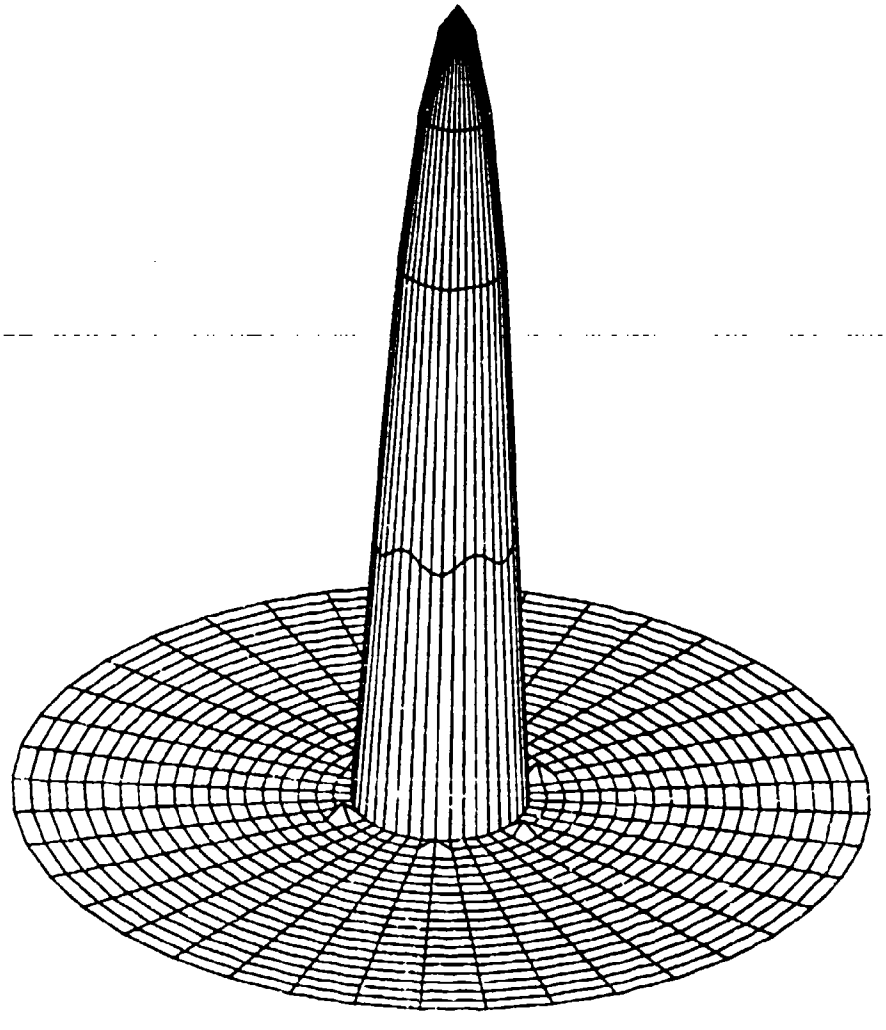

Fig. 7. Sum Power Pattern with No Failures

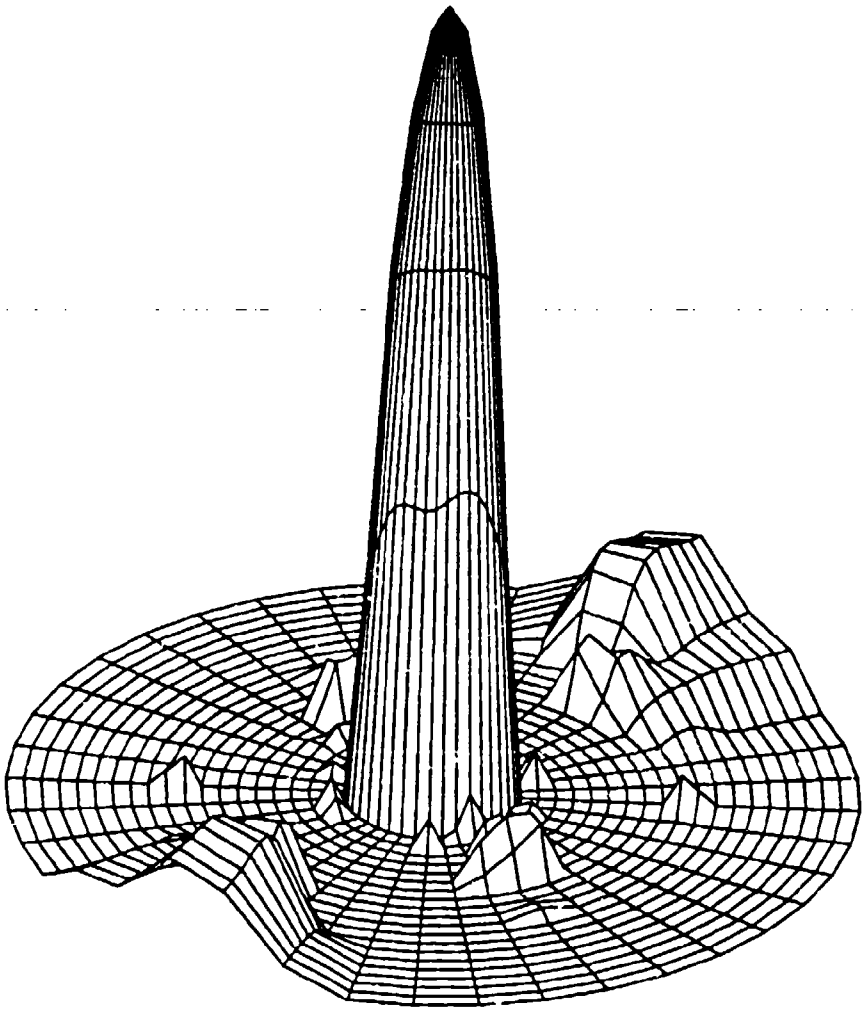

Fig. 8. Sum Power Pattern with 3 Element Failures

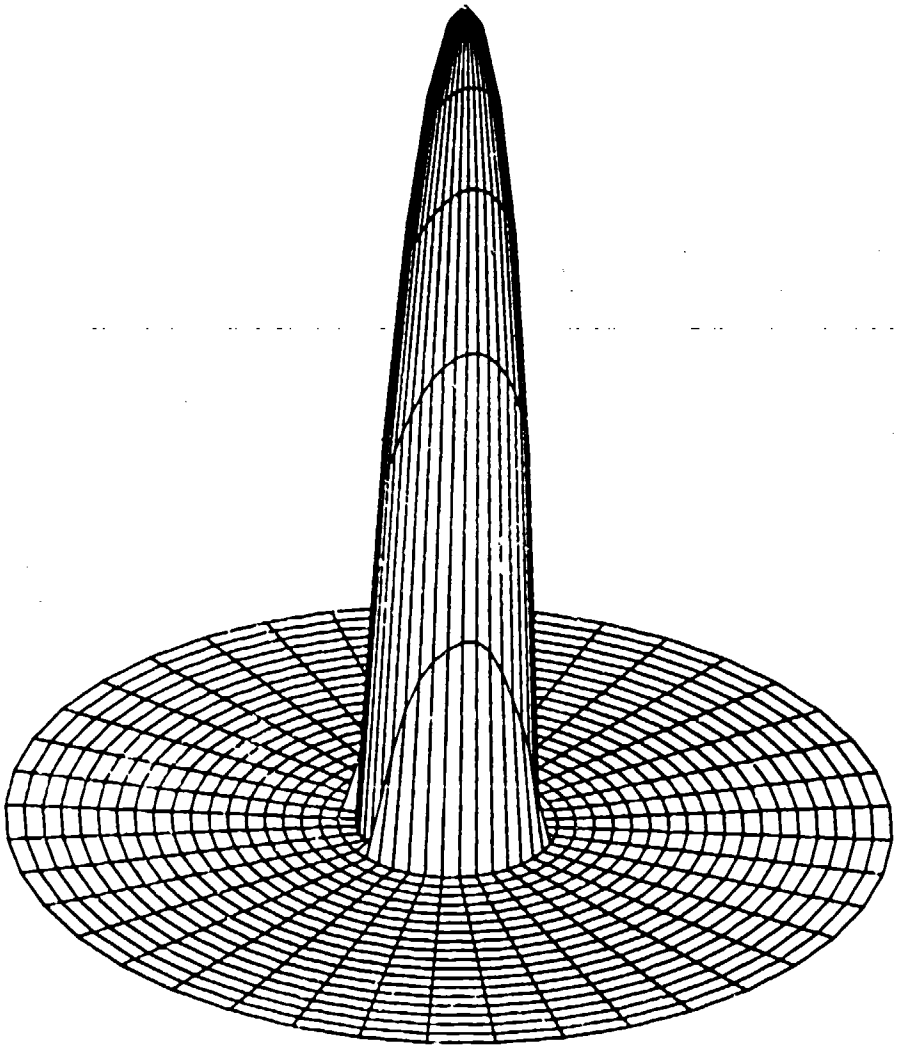

Fig. 9. Synthesized Sum Power Pattern with 3 Element Failures

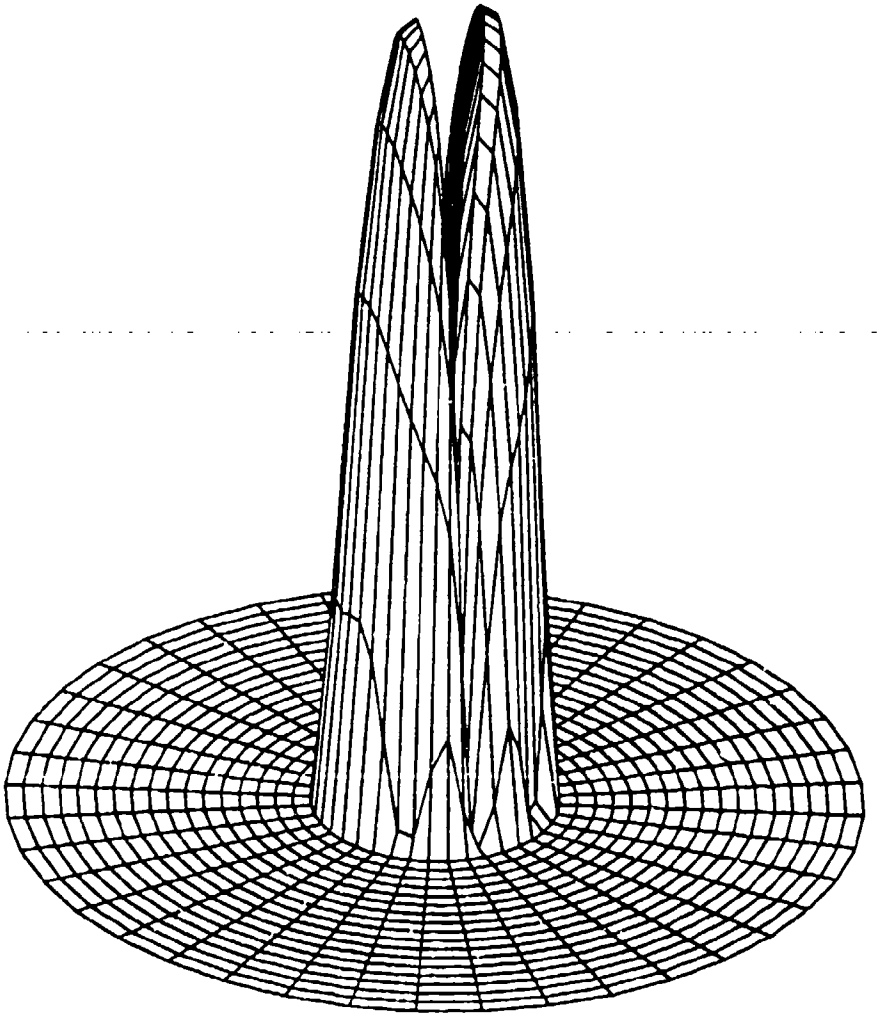

Fig. 10. Difference Power Pattern with No Failures

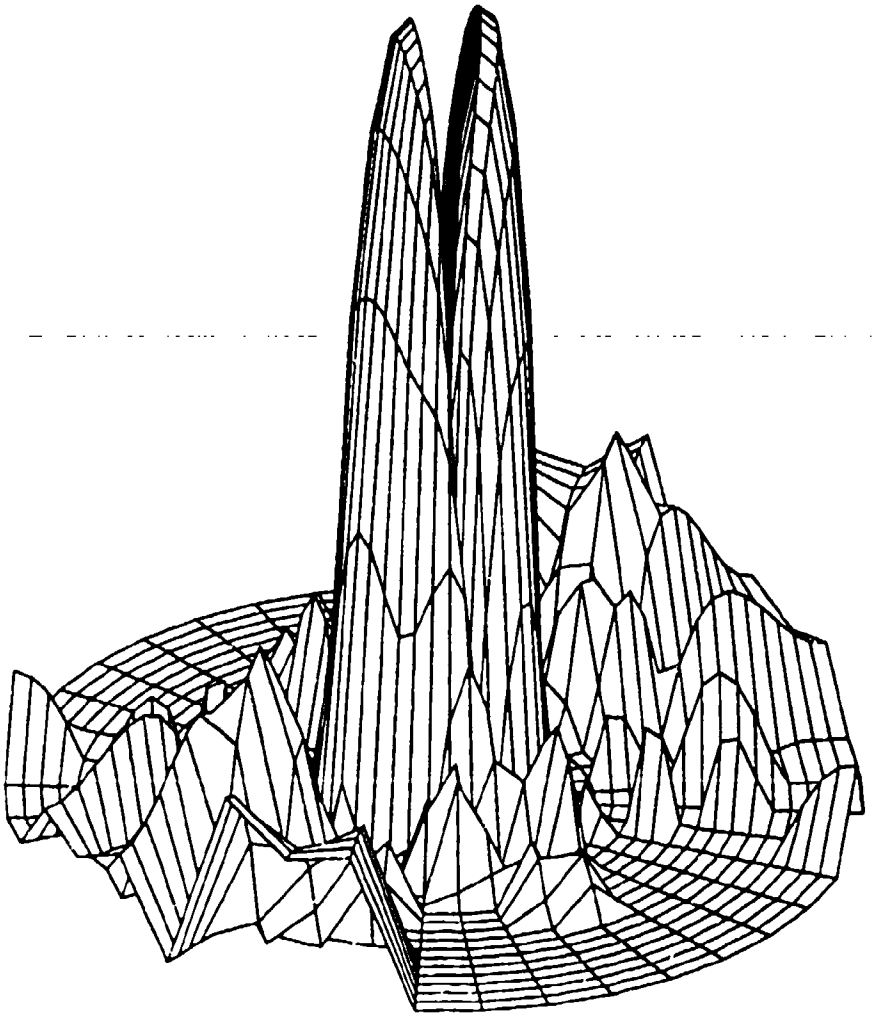

Fig. 11. Difference Power Pattern with 3 Element Failures

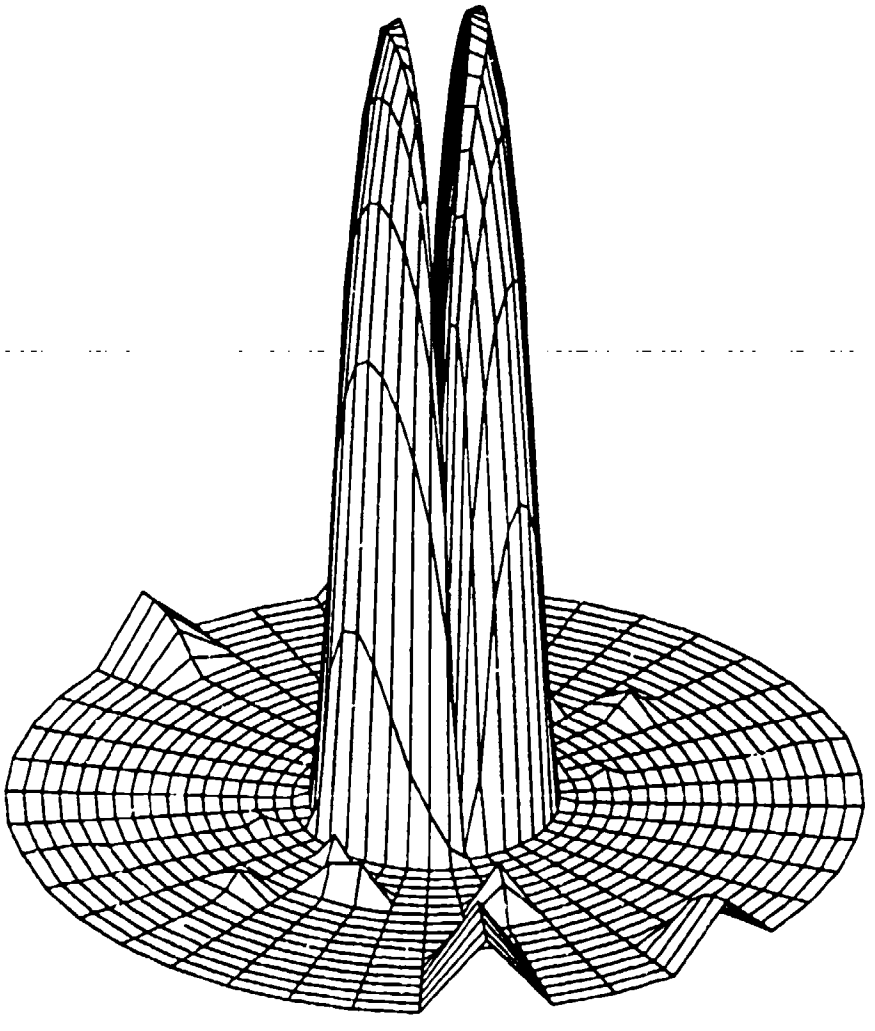

Fig. 12. Synthesized Difference Power Pattern with 3 Element Failures

The peak sidelobe level is -28.7 dB with a gain of 17.1 dB. The sidelobe level of the synthesized pattern cannot be brought down to the level of the design pattern as was the case with the sum pattern, because the degrees of freedom are now about halved. The half-power null width between peaks increased to 9.09°.

The results of this test case and several others, indicate that it is possible to reduce the sidelobe level, of an array with failed elements, down to near the design level. However, the success depends on the nature of the failure. For example, if the central element fails in a sum pattern, the synthesis yields no improvement. This is to be expected since this failure is quite similar to blockage. As expected, elements which fail and have a large relative weight are much more difficult to recover from than elements with a small weight. Also, clustered failures are easier to recover from than the same number of random failures. This is because a clustered failure yields a relatively local increase in the sidelobe region as opposed to the same number of random failures which yields a uniform increase in the sidelobe level. It was interesting to observe that is was easier to recover from an entire row failure than from the same number of random failures. In general, the decrease in the sidelobe level comes with a drop in gain and a slightly broader main beam.

VII. Conclusion

The performance degradation of arrays with element failures may be partially compensated for by a redistribution of the amplitude and phase over the remaining elements. The extent of this compensation is determined by the number and location of the failed elements. The accumulated averaging scheme, combined with the conjugate gradient algorithm, provides a very stable means to synthesize the new array aperture distribution.

References

1. J. F. Deford and O. P. Gandhi, "Phase-Only Synthesis of Minimum Peak Sidelobe Patterns for Linear and Planar Arrays," IEEE Trans. Antennas Propagat., vol, 36, no. 2, pp. 191-201, Feb. 1988.

2. T. J. Peters, "Application of a Conjugate Gradient Method to the Synthesis of Phase-Only Arrays," Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0090(5925-05)-1, Feb. 1991.

3. M. R. Hestenes and E. Steifel, "Method of Conjugate Gradients for Solving Linear Systems," J. Res. Nat. Bur. Standard., vol. 49, no. 6, pp. 409-436, Dec. 1952.

4. M. R. Hestenes, " Conjugate Direction Methods in Optimization," New York: Springer-Verlag, 1980, pp.135-140.

5. T. J. Peters, " Computation of the Scattering by Planar and Non-Planar Plates Using a Conjugate Gradient FFT Method," Ph.D. dissertation, Radiation Laboratory, University of Michigan, 1988.

6. T. S. Fong and R. A. Birgenheier, "Method of Conjugate Gradients for Antenna Pattern Synthesis," Radio Sci., vol. 6, no. 12, pp. 1123-1130, Dec. 1971.

7. W. A. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Conjugate Gradient Methods in Multidimensions," in Numerical Recipes: Cambridge Cambridge University Press, 1988, pp. 301-307.

Appendix: Fortran 77 Programs

This appendix contains the Fortran 77 programs SUMSYN and DIFSYN as well as all associated subroutines. The program SUMSYN performs the sum pattern synthesis with element failures and the program DIFSYN performs difference pattern synthesis with element failures. The required inputs to each program are listed in the comments found in the source code. These programs were used to generate all the results presented in this report.

```
      PROGRAM SUMSYN
C     ******************************************************************
C     * THIS PROGRAM SYNTHESIZES THE AMPLITUDE AND PHASE DISTRIBUTION  *
C     * NECESSARY TO GENERATE A SUM PATTERN OVER A HEXAGONAL ARRAY OF  *
C     * POINT SOURCES WITH SOME ELEMENT FAILURES.                      *
C     ******************************************************************
C     * TIMOTHY J. PETERS                              LAST UPDATED    *
C     * THE AEROSPACE CORPORATION                      3/1/91          *
C     * 2350 EAST EL SEGUNDO BOULEVARD.                                *
C     * EL SEGUNDO, CA 90245                                           *
C     ******************************************************************
C     * INPUTS:                                                        *
C     *                                                                *
C     * NL  - NUMBER OF LAYERS FORMING THE HEXAGONAL LATTICE           *
C     * NTE - NUMBER OF TOTAL ELEMENTS = 1+3*NL*(NL+1)                 *
C     * NBE - NUMBER OF BAD ELEMENTS                                   *
C     * BAD(1..NBE) - INTEGER ARRAY HOLDING THE NUMBER POSITION OF EACH *
C     *       BAD ELEMENT IN THE ARRAY.                                *
C     * DS  - THE ELEMENT SPACING IN WAVELENGTHS.                      *
C     * NE  - NUMBER OF ACTUAL ELEMENTS = NTE-NBE.                     *
C     * ET  - THE EDGE TAPER OF THE PARABOLIC DISTRIBUTION.            *
C     * N   - THE ORDER OF THE PARABOLIC DISTRIBUTION.                 *
C     * BNF - THE THEORETICAL BEAM WIDTH NULL FACTOR FOR THE CHOSEN    *
C     *       DISTRIBUTION.                                            *
C     * BBF - THE BEAM WIDTH BROADENING FACTOR.  THIS ALLOWS THE BEAM  *
C     *       TO BROADEN BEYOND THE DESIGN VALUE IN ORDER TO REDUCE THE *
C     *       SIDELOBES WHEN ELEMENTS FAIL.                            *
C     * NSP - NUMBER OF SAMPLE POINTS IN THE SIDELOBE REGION.          *
C     * NS  - NUMBER OF RESTARTS.                                      *
C     * NI  - NUMBER OF CONJUGATE GRADIENT ITERATIONS PER RESTART.     *
C     *                                                                *
C     * OUTPUTS:                                                       *
C     *                                                                *
C     * U(NE) - REAL PART OF THE EXCITATION OF EACH ELEMENT.           *
C     * V(NE) - IMAGINARY PART OF THE EXCITATION OF EACH ELEMENT.      *
C     *                                                                *
C     ******************************************************************
      PARAMETER (NL=5,NTE=91,NBE=3,NE=88,NSP=1550)
      REAL*4 U(NE),V(NE),X(NE),Y(NE),SU(NE),SV(NE),GU(NE),GV(NE)
      REAL*4 HUU(NE,NE),HUV(NE,NE),HVU(NE,NE),HVV(NE,NE)
      REAL*4 PU(NE),PV(NE),QU(NE),QV(NE),RU(NE),RV(NE),ZU(NE),ZV(NE)
      REAL*4 KX(NSP),KY(NSP)
      INTEGER IP(NSP),BAD(NE)
      OPEN(UNIT=2,FILE='OUTPUT',STATUS='UNKNOWN')
      OPEN(UNIT=4,FILE='SUM_FIXED_EST',STATUS='UNKNOWN')
C     ******************************************************************
C     * GENERATE THE GEOMETRY.                                         *
C     ******************************************************************
      DS=0.6
```

```
      BAD(1)=35
      BAD(2)=37
      BAD(3)=44
      CALL GEOMET(NL,DS,NE,NBE,BAD,X,Y)
C     ******************************************************************
C     * INITIALIZE THE TAPER DISTRIBUTION.                             *
C     ******************************************************************
      ET=0.1
      N=2
      BNF=1.5
      IFLAG1=0
      IF (IFLAG1 .EQ.-1) THEN
        CALL TAPER(NE,U,V,X,Y,ET,N,NL,DS)
        REWIND 4
        WRITE(4,101) (I,U(I),V(I),I=1,NE)
      ELSE
      END IF
      REWIND 4
      READ(4,101) (I,U(I),V(I),I=1,NE)
C     ******************************************************************
C     * COMPUTE THE GAIN (DIRECTIVITY) OF THE ARRAY.                   *
C     ******************************************************************
      CALL GAIN(NE,U,V,X,Y,GDB)
C     ******************************************************************
C     * GET THE BEAM WIDTH BETWEEN FIRST NULLS.                        *
C     ******************************************************************
      CALL BEAM(NE,U,V,X,Y,BWFN)
C     ******************************************************************
C     * GENERATE THE K-SPACE SAMPLE POINTS.                            *
C     ******************************************************************
      BBF=7.0
      CALL KSPACE(NMP,NL,DS,BNF,BBF,KX,KY)
C     ******************************************************************
C     * COMPUTE POWER IN MAIN BEAM.                                    *
C     ******************************************************************
      CALL POWI(NE,U,V,X,Y,0.0,0.0,PO)
C     ******************************************************************
C     * COMPUTE PEAK SIDELOBE POWER.                                   *
C     ******************************************************************
      NP=0
      CALL PEAKSL(NE,U,V,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
C     ******************************************************************
C     * COMPUTE NORMALIZED PEAK SIDELOBE POWER IN DB.                  *
C     ******************************************************************
      PMAXDB=10.0*ALOG10(PMAX/PO)
C     ******************************************************************
C     * PERFORM CONJUGATE GRADIENT STEPS.                              *
C     ******************************************************************
      NS=300
```

```
      NI=3
      CALL CGRAD(NE,NSP,NMP,NS,NI,U,V,X,Y,SU,SV,NP,IP,KX,KY,GU,GV,HUU
     &,HUV,HVU,HVV,PU,PV,QU,QV,RU,RV,ZU,ZV)
      REWIND 4
      WRITE(4,101) (I,U(I),V(I),I=1,NE)
C     ****************************************************************
C     * POWER PATTERN GRAPHICS OUTPUT                                *
C     ****************************************************************
      REWIND 4
      READ(4,101) (I,U(I),V(I),I=1,NE)
C       ************************************************************
C       * PRINT THE POWER PATTERN IN CYLINDRICAL COORDINATES.      *
C       ************************************************************
        CALL POWPAT(NE,U,V,X,Y)
C       ************************************************************
C       * PRINT THE POWER PATTERN IN SPHERICAL COORDINATES.        *
C       ************************************************************
        CALL SPOWPAT(NE,U,V,X,Y)
C     ****************************************************************
C     * FORMATS.                                                     *
C     ****************************************************************
100   FORMAT(I6,1X,F16.6,1X,F16.6)
101   FORMAT(I6,1X,F16.7,1X,F16.7)
      END
C
      SUBROUTINE GEOMET(NL,S,NE,NBE,BAD,X,Y)
C     ****************************************************************
C     * COMPUTE THE POSITION OF EACH ELEMENT IN A HEXOGONAL ARRAY.   *
C     ****************************************************************
      REAL*4 X(NE),Y(NE)
      INTEGER BAD(NE),IFLAG
      HGT=(SQRT(3.0)/2.0)*S
C     ****************************************************************
C     * CENTER ROW.                                                  *
C     ****************************************************************
      K=0
      L=0
      XMIN=-NL*S
      DO 1 I=0,2*NL
       K=K+1
       L=L+1
       CALL CHECK(L,NE,NBE,BAD,IFLAG)
       IF (IFLAG .EQ. 1) THEN
         K=K-1
       ELSE
         X(K)=XMIN+I*S
         Y(K)=0.0
       END IF
1     CONTINUE
```

```
C     ****************************************************************
C     * TOP ROWS.                                                    *
C     ****************************************************************
      INUM=2*NL+1
      DO 2 J=1,NL
        INUM=INUM-1
        XMIN=-NL*S+J*S/2.0
        DO 3 I=0,INUM-1
          K=K+1
          L=L+1
          CALL CHECK(L,NE,NBE,BAD,IFLAG)
          IF (IFLAG .EQ. 1) THEN
            K=K-1
          ELSE
            X(K)=XMIN+I*S
            Y(K)=HGT*J
          END IF
3       CONTINUE
2     CONTINUE
C     ****************************************************************
C     * BOTTOM ROWS.                                                 *
C     ****************************************************************
      INUM=2*NL+1
      DO 4 J=1,NL
        INUM=INUM-1
        XMIN=-NL*S+J*S/2.0
        DO 5 I=0,INUM-1
          K=K+1
          L=L+1
          CALL CHECK(L,NE,NBE,BAD,IFLAG)
          IF (IFLAG .EQ. 1) THEN
            K=K-1
          ELSE
            X(K)=XMIN+I*S
            Y(K)=-HGT*J
          END IF
5       CONTINUE
4     CONTINUE
      RETURN
      END
C
      SUBROUTINE CHECK(L,NE,NBE,BAD,IFLAG)
      INTEGER BAD(NE)
      IFLAG=0
      DO 1 I=1,NBE
         IF (L .EQ. BAD(I)) THEN
            IFLAG=1
         ELSE
         END IF
```

```
1      CONTINUE
       RETURN
       END
C
       SUBROUTINE BEAM(NE,U,V,X,Y,BWFN)
C     ****************************************************************
C     * COMPUTE THE BEAM WIDTH BETWEEN FIRST NULLS.                  *
C     ****************************************************************
       REAL U(NE),V(NE),X(NE),Y(NE),KXI,KYI,KXP,KYP
       RAD=.17453293E-01
       TP=.6283185E+01
       TMIN=0.0
       TMAX=10.0
       P=0.0
       NT=140
       DT=(TMAX-TMIN)/NT
C     ****************************************************************
C     * FIND THE MAXIMUM VALUE.                                      *
C     ****************************************************************
       CALL POWER(NE,U,V,X,Y,0.0,0.0,PEAK)
C     ****************************************************************
C     * RECOMPUTE AND NOW FIND THE POINT WHICH IS AT THE SLL BELOW THE *
C     * PEAK.                                                        *
C     ****************************************************************
       DO 1 K=1,90
         P=P+1.0
         DO 2 I=0,NT
           T=TMIN+I*DT
           TPS=TP*SIN(RAD*T)
           KXI=TPS*COS(RAD*P)
           KYI=TPS*SIN(RAD*P)
           CALL POWI(NE,U,V,X,Y,KXI,KYI,POW)
           IF (POW/PEAK .LE. 0.5) THEN
             WRITE(2,100) T,10.0*ALOG(POW/PEAK)
             GO TO 99
           ELSE
           END IF
 2       CONTINUE
99       CONTINUE
         BWFN=2.0*T
         WRITE(*,*) 'P BWFN ',P,BWFN
1      CONTINUE
100    FORMAT(F10.5,1X,F10.5)
       RETURN
       END
C
       SUBROUTINE KSPACE(NMP,NL,DS,BNF,BBF,KX,KY)
C     ****************************************************************
C     * THIS SUBROUTINE SAMPLES THE KX >0 REGION OF K SPACE.         *
```

```
C     ****************************************************************
      REAL*4 KX(*),KY(*),KC,KR,KXX,KYY,KXMIN,KXMAX,KYMIN,KYMAX
      RAD=.17453293E-01
      PI=.3141593E+01
      TP=.6283185E+01
C     ****************************************************************
C     * COMPUTE THE APPROXIMATE DIAMETER OF THE APERTURE.            *
C     ****************************************************************
      D=2.0*DS*NL
C     ****************************************************************
C     * COMPUTE THE BEAM WIDTH BETWEEN FIRST NULLS OF THE DESIGN     *
C     * PATTERN.                                                     *
C     ****************************************************************
      BWFN=BNF*2.439272/D
C     ****************************************************************
C     * COMPUTE THE SMALLEST BEAM WIDTH BETWEEN FIRST NULLS ALLOWED BY *
C     * THE SYNTHESIS ALGORITHM.                                     *
C     ****************************************************************
      BWFNS=BBF*BWFN
C     ****************************************************************
C     * COMPUTE THE K SPACE RADIUS OF THE ALLOWED MAIN BEAM.         *
C     ****************************************************************
C       KC=TP*0.34
      KC=BWFNS/2.0
C     ****************************************************************
C     * GENERATE THE RECTANGULAR LATTICE OF SAMPLE POINTS.           *
C     ****************************************************************
      KYMIN=-TP
      KYMAX=TP
      KXMIN=-TP
      KXMAX=TP
      NKX=45
      NKY=45
      DKX=(KXMAX-KXMIN)/(NKX-1)
      DKY=(KYMAX-KYMIN)/(NKY-1)
      NMP=0
      DO 1 I=0,NKY
        KYY=KYMIN+I*DKY
        DO 2 J=0,NKX
          KXX=KXMIN+J*DKX
          KR=SQRT(KXX*KXX+KYY*KYY)
          IF ((KR .GT. KC) .AND. (KR .LT. TP)) THEN
            NMP=NMP+1
            KX(NMP)=KXX
            KY(NMP)=KYY
          ELSE
          END IF
    2   CONTINUE
    1 CONTINUE
```

```
      RETURN
      END
C
      SUBROUTINE TAPER(NE,U,V,X,Y,ET,N,NL,S)
      REAL U(NE),V(NE),X(NE),Y(NE)
C     ***************************************************************
C     * COMPUTE APPROXIMATE ARRAY RADIUS.                           *
C     ***************************************************************
      A=S*NL
C     ***************************************************************
C     * COMPUTE AMPLITUDE.                                          *
C     ***************************************************************
      DO 1 K=1,NE
       R=SQRT(X(K)*X(K)+Y(K)*Y(K))
       AMP=ET+(1.0-ET)*((1.0-(R/A)**2)**N)
       U(K)=AMP
       V(K)=0.0
1     CONTINUE
      RETURN
      END
C
      SUBROUTINE CGRAD(NE,NSP,NMP,NS,NI,U,V,X,Y,SU,SV,NP,IP,KX,KY,GU,GV
     &                 ,HUU,HUV,HVU,HVV,PU,PV,QU,QV,RU,RV,ZU,ZV)
C     ***************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                        *
C     ***************************************************************
      REAL*4 U(NE),V(NE),X(NE),Y(NE),SU(NE),SV(NE),GU(NE),GV(NE)
      REAL*4 HUU(NE,NE),HUV(NE,NE),HVU(NE,NE),HVV(NE,NE)
      REAL*4 PU(NE),PV(NE),QU(NE),QV(NE),RU(NE),RV(NE),ZU(NE),ZV(NE)
      REAL*4 KX(NSP),KY(NSP)
      REAL*4 KXI,KYI
      INTEGER IP(NSP)
C     ***************************************************************
C     * INITIALIZE THE NUMBER OF SIDELOBE SAMPLE POINTS.            *
C     ***************************************************************
      NP=0
C     ***************************************************************
C     * PERFORM NS RE-STARTS.                                       *
C     ***************************************************************
      DO 1 L=1,NS
C         ***********************************************************
C         * FIND THE PEAK SIDELOBE POWER POINT AND UPDATE THE NUMBER *
C         * OF MATCH POINTS.                                        *
C         ***********************************************************
          CALL PEAKSL(NE,U,V,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
          WRITE(*,160) (IP(I),I=1,NP)
          WRITE(2,160) (IP(I),I=1,NP)
160       FORMAT(10I5)
C         ***********************************************************
```

```
C         * FIND THE POWER IN THE MAIN BEAM.                                *
C         ***************************************************************
          KXI=0.0
          KYI=0.0
          CALL POWI(NE,U,V,X,Y,KXI,KYI,PO)
C         ***************************************************************
C         * COMPUTE AND STORE AN ARRAY OF THE 1ST DERIVATIVE OF THE      *
C         * POWER IN THE MAIN BEAM DIRECTION WITH RESPECT TO THE MTH     *
C         * VARIABLE.                                                    *
C         ************************************************************ ***
          KXI=0.0
          KYI=0.0
          DO 2 M=1,NE
            CALL DPOWIU(NE,U,V,X,Y,M,KXI,KYI,DPWIU)
            SU(M)=DPWIU
            CALL DPOWIV(NE,U,V,X,Y,M,KXI,KYI,DPWIV)
            SV(M)=DPWIV
2         CONTINUE
C         ***************************************************************
C         * COMPUTE AND STORE THE AVERAGE SIDELOBE POWER.                *
C         ***************************************************************
          PA=0.0
          DO 3 I=1,NP
            CALL POWI(NE,U,V,X,Y,KX(IP(I)),KY(IP(I)),POWERI)
            PA=PA+POWERI
3         CONTINUE
          PA=PA/NP
          WRITE(*,200) L,10.0*ALOG10(PA/PO),10.0*ALOG10(PMAX/PO)
          WRITE(2,200) L,10.0*ALOG10(PA/PO),10.0*ALOG10(PMAX/PO)
200       FORMAT(1X,I5,1X,'AVE PEAK SLL',1X,F15.6,1X,
     &                                 'PEAK SLL',1X,F15.6)
C         ***************************************************************
C         * COMPUTE THE NEGATIVE OF THE GRADIENT.                        *
C         ***************************************************************
          DO 4 M=1,NE
C             ***************************************************************
C             * COMPUTE THE DERIVATIVE OF THE AVERAGE SIDELOBE POWER. *
C             ***************************************************************
              DPAMU=0.0
              DPAMV=0.0
              DO 5 I=1,NP
                CALL DPOWIU(NE,U,V,X,Y,M,KX(IP(I)),KY(IP(I)),DPWIU)
                DPAMU=DPAMU+DPWIU
                CALL DPOWIV(NE,U,V,X,Y,M,KX(IP(I)),KY(IP(I)),DPWIV)
                DPAMV=DPAMV+DPWIV
5             CONTINUE
              DPAMU=DPAMU/NP
              DPAMV=DPAMV/NP
C             ***************************************************************
```

```
C              * COMPUTE THE NEGATIVE GRADIENT.                          *
C              ***********************************************************
               GU(M)=(PA*SU(M)/PO-DPAMU)/PO
               GV(M)=(PA*SV(M)/PO-DPAMV)/PO
4          CONTINUE
C         ***********************************************************
C         * COMPUTE THE SYMMETRIC HESSIAN SUB MATRICES HUU AND HVV     *
C         ***********************************************************
           DO 6 M=1,NE
             DO 7 N=M,NE
               KXI=0.0
               KYI=0.0
               CALL DDPIUU(NE,X,Y,M,N,KXI,KYI,DDPUU)
               DDPOUU=DDPUU
               CALL DDPIVV(NE,X,Y,M,N,KXI,KYI,DDPVV)
               DDPOVV=DDPVV
               DDPAUU=0.0
               DDPAVV=0.0
               DO 8 I=1,NP
                 CALL DDPIUU(NE,X,Y,M,N,KX(IP(I)),KY(IP(I)),DDPUU)
                 DDPAUU=DDPAUU+DDPUU
                 CALL DDPIVV(NE,X,Y,M,N,KX(IP(I)),KY(IP(I)),DDPVV)
                 DDPAVV=DDPAVV+DDPVV
8              CONTINUE
               DDPAUU=DDPAUU/NP
               DDPAVV=DDPAVV/NP
               HUU(M,N)=(GU(N)*SU(M)+GU(M)*SU(N)+DDPAUU-PA*DDPOUU/PO)/PO
               HUU(N,M)=HUU(M,N)
               HVV(M,N)=(GV(N)*SV(M)+GV(M)*SV(N)+DDPAVV-PA*DDPOVV/PO)/PO
               HVV(N,M)=HVV(M,N)
7           CONTINUE
6          CONTINUE
C         ***********************************************************
C         * COMPUTE THE ASYMMETRIC HESSIAN SUB MATRICE HUV.            *
C         ***********************************************************
           DO 9 M=1,NE
             DO 10 N=1,NE
               KXI=0.0
               KYI=0.0
               CALL DDPIUV(NE,X,Y,M,N,KXI,KYI,DDPUV)
               DDPOUV=DDPUV
               DDPAUV=0.0
               DO 11 I=1,NP
                 CALL DDPIUV(NE,X,Y,M,N,KX(IP(I)),KY(IP(I)),DDPUV)
                 DDPAUV=DDPAUV+DDPUV
11             CONTINUE
               DDPAUV=DDPAUV/NP
               HUV(M,N)=(GV(N)*SU(M)+GU(M)*SV(N)+DDPAUV-PA*DDPOUV/PO)/PO
               HVU(N,M)=HUV(M,N)
```

```
10          CONTINUE
 9         CONTINUE
C         ************************************************************
C         * START THE CONJUGATE GRADIENT ALGORITHM.                  *
C         ************************************************************
C         ************************************************************
C         * INITIALIZE THE RESIDUAL.                                 *
C         ************************************************************
           DO 12 M=1,NE
             RU(M)=GU(M)
             RV(M)=GV(M)
             ZU(M)=0.0
             ZV(M)=0.0
12         CONTINUE
C         ************************************************************
C         * INITIALIZE SEARCH VECTOR.                                *
C         ************************************************************
           CALL MATVEC(NE,HUU,RU,HUV,RV,QU)
           CALL MATVEC(NE,HVU,RU,HVV,RV,QV)
           CALL NORM22(NE,QU,QUN)
           CALL NORM22(NE,QV,QVN)
           BEO=1.0/(QUN+QVN)
           DO 13 I=1,NE
             PU(I)=BEO*QU(I)
             PV(I)=BEO*QV(I)
13         CONTINUE
C         ************************************************************
C         * PERFORM CONJUGATE GRADIENT ITERATIONS.                   *
C         ************************************************************
           DO 14 K=1,NI
C         ************************************************************
C         * UPDATE AMPLITUDE VECTOR AND RESIDUAL.                    *
C         ************************************************************
           CALL MATVEC(NE,HUU,PU,HUV,PV,QU)
           CALL MATVEC(NE,HVU,PU,HVV,PV,QV)
           CALL NORM22(NE,QU,QUN)
           CALL NORM22(NE,QV,QVN)
           AK=1.0/(QUN+QVN)
           DO 15 I=1,NE
             ZU(I)=ZU(I)+AK*PU(I)
             ZV(I)=ZV(I)+AK*PV(I)
             RU(I)=RU(I)-AK*QU(I)
             RV(I)=RV(I)-AK*QV(I)
15         CONTINUE
           CALL NORM22(NE,RU,RUNM)
           CALL NORM22(NE,RV,RVNM)
           ERR=SQRT(RUNM+RVNM)
           WRITE(*,600) L,K,ERR
           WRITE(2,600) L,K,ERR
```

```
500         FORMAT(3X,'RESIDUAL',1X,I3,1X,I3,1X,E15.8)
C          ************************************************************
C          * UPDATE SEARCH VECTOR.                                    *
C          ************************************************************
            CALL MATVEC(NE,HUU,RU,HUV,RV,QU)
            CALL MATVEC(NE,HVU,RU,HVV,RV,QV)
            CALL NORM22(NE,QU,QUN)
            CALL NORM22(NE,QV,QVN)
            BEK=1.0/(QUN+QVN)
            DO 16 I=1,NE
              PU(I)=PU(I)+BEK*QU(I)
              PV(I)=PV(I)+BEK*QV(I)
  16        CONTINUE
14          CONTINUE
C          ************************************************************
C          * UPDATE AMPLITUDE VECTOR.                                 *
C          ************************************************************
            DO 17 I=1,NE
              U(I)=U(I)+ZU(I)
              V(I)=V(I)+ZV(I)
  17        CONTINUE
            CALL NORMAL(NE,U,V)
C          ************************************************************
C          * WRITE OUT THE NEW ESTIMATE TO A FILE.                    *
C          ************************************************************
          REWIND 4
          WRITE(4,101) (I,U(I),V(I),I=1,NE)
101     FORMAT(I5,1X,F15.7,1X,F15.7)
        IF (NP .EQ. NMP) THEN
          GO TO 99
        ELSE
        END IF
1       CONTINUE
99      CONTINUE
        RETURN
        END
C
        SUBROUTINE PEAKSL(NE,U,V,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
C      ***************************************************************
C      * THIS SUBROUTINE FINDS THE PEAK SIDELOBE POWER LEVEL COORDINATE *
C      * POINT (KX(LMAX),KY(LMAX)).  IF THE POINT IS NEW THEN IT IS     *
C      * ACCUMULATED INTO THE POINTER ARRAY IP. THE NUMBER OF POINTS IN *
C      * THE INTEGER POINTER ARRAY IP IS NP SUCH THAT IP(I),I=1...NP.   *
C      ***************************************************************
        REAL U(NE),V(NE),X(NE),Y(NE),KX(NSP),KY(NSP)
        INTEGER IP(NSP)
C          ************************************************************
C          * COMPUTE THE POWER AT EACH POINT AND FIND THE MAX.        *
C          ************************************************************
```

```
          PMAX=-1000.0
          LMAX=-1
          DO 1 I=1,NMP
            CALL POWI(NE,U,V,X,Y,KX(I),KY(I),PWI)
            IF (PWI .GT. PMAX) THEN
              LMAX=I
              PMAX=PWI
            ELSE
            END IF
1         CONTINUE
C        ************************************************************
C        * COMPARE LMAX TO ALL THE PREVIOUS POINTERS IP(I),I=1...NP.  *
C        * IF LMAX MATCHES ONE OF THE PREVIOUS POINTERS THEN THE      *
C        * POINTER ARRAY NEED NOT BE UPDATED (IFLAG=1).  IF LMAX DOES *
C        * NOT MATCH A PREVIOUS POINTER VALUE (IFLAG=0) THEN IT       *
C        ************************************************************
          IFLAG=0
          DO 2 I=1,NP
            IF (IP(I) .EQ. LMAX) THEN
              IFLAG=1
            ELSE
            END IF
2         CONTINUE
          IF (IFLAG .EQ. 0) THEN
            NP=NP+1
            IP(NP)=LMAX
          ELSE
          END IF
      RETURN
      END
C
      SUBROUTINE POWI(NE,U,V,X,Y,KXI,KYI,POWERI)
C    ***************************************************************
C    * COMPUTE POWER IN A SINGLE DIRECTION.                        *
C    ***************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE),KXI,KYI
      SUM1=0.0
      SUM2=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       CP=COS(PSI)
       SP=SIN(PSI)
       SUM1=SUM1+U(K)*CP-V(K)*SP
       SUM2=SUM2+U(K)*SP+V(K)*CP
1     CONTINUE
      TI=SUM1*SUM1
      VI=SUM2*SUM2
      POWERI=TI+VI
      RETURN
```

```
      END
C
      SUBROUTINE DPOWIU(NE,U,V,X,Y,M,KXI,KYI,DPWIU)
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF POWER WITH RESPECT TO THE MTH VARIABLE*
C     * IN THE ITH DIRECTION.                                          *
C     ****************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE),KXI,KYI
C     ****************************************************************
C     * COMPUTE SUMS.                                                  *
C     ****************************************************************
      SUM1=0.0
      SUM2=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       CP=COS(PSI)
       SP=SIN(PSI)
       SUM1=SUM1+U(K)*CP-V(K)*SP
       SUM2=SUM2+U(K)*SP+V(K)*CP
1     CONTINUE
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF POWER.                               *
C     ****************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      DPWIU=2.0*(COS(PSIM)*SUM1+SIN(PSIM)*SUM2)
      RETURN
      END
C
      SUBROUTINE DPOWIV(NE,U,V,X,Y,M,KXI,KYI,DPWIV)
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF POWER WITH RESPECT TO THE MTH VARIABLE*
C     * IN THE ITH DIRECTION.                                          *
C     ****************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE),KXI,KYI
C     ****************************************************************
C     * COMPUTE SUMS.                                                  *
C     ****************************************************************
      SUM1=0.0
      SUM2=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       CP=COS(PSI)
       SP=SIN(PSI)
       SUM1=SUM1+U(K)*CP-V(K)*SP
       SUM2=SUM2+U(K)*SP+V(K)*CP
1     CONTINUE
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF POWER.                               *
C     ****************************************************************
```

```
      PSIM=KXI*X(M)+KYI*Y(M)
      DPWIV=2.0*(COS(PSIM)*SUM2-SIN(PSIM)*SUM1)
      RETURN
      END
C
      SUBROUTINE DDPIUU(NE,X,Y,M,N,KXI,KYI,DDPUU)
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER WITH RESPECT TO THE M AND NTH   *
C     * VARIABLES IN THE ITH DIRECTION.                                 *
C     ******************************************************************
      REAL*4 X(NE),Y(NE),KXI,KYI
C     ******************************************************************
C     * COMPUTE DIFFERENCE OF PSI FUNCTIONS.                            *
C     ******************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      PSIN=KXI*X(N)+KYI*Y(N)
      DIF=PSIM-PSIN
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER.                                *
C     ******************************************************************
      DDPUU=2.0*COS(DIF)
      RETURN
      END
C
      SUBROUTINE DDPIUV(NE,X,Y,M,N,KXI,KYI,DDPUV)
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER WITH RESPECT TO THE M AND NTH   *
C     * VARIABLES IN THE ITH DIRECTION.                                 *
C     ******************************************************************
      REAL*4 X(NE),Y(NE),KXI,KYI
C     ******************************************************************
C     * COMPUTE DIFFERENCE OF PSI FUNCTIONS.                            *
C     ******************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      PSIN=KXI*X(N)+KYI*Y(N)
      DIF=PSIM-PSIN
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER.                                *
C     ******************************************************************
      DDPUV=2.0*SIN(DIF)
      RETURN
      END
C
      SUBROUTINE DDPIVV(NE,X,Y,M,N,KXI,KYI,DDPVV)
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER WITH RESPECT TO THE M AND NTH   *
C     * VARIABLES IN THE ITH DIRECTION.                                 *
C     ******************************************************************
      REAL*4 X(NE),Y(NE),KXI,KYI
```

```
C     ******************************************************************
C     * COMPUTE DIFFERENCE OF PSI FUNCTIONS.                           *
C     ******************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      PSIN=KXI*X(N)+KYI*Y(N)
      DIF=PSIM-PSIN
C     ******************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER.                               *
C     ******************************************************************
      DDPVV=2.0*COS(DIF)
      RETURN
      END
C
      SUBROUTINE NORM22(N,A,AN)
C     ******************************************************************
C     * THIS SUBROUTINE COMPUTES THE EUCLIDIAN NORM SQUARED OF A.      *
C     ******************************************************************
      REAL*4 A(N)
      AN=0.0
      DO 1 I=1,N
       AN=AN+A(I)*A(I)
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE MATVEC(N,A,B,C,D,E)
C     ******************************************************************
C     * THIS SUBROUTINE COMPUTES THE MATRIX PRODUCT E=AB+CD.           *
C     ******************************************************************
      REAL*4 A(N,N),B(N),C(N,N),D(N),E(N)
      DO 1 I=1,N
       SUM=0.0
       DO 2 J=1,N
        SUM=SUM+A(I,J)*B(J)+C(I,J)*D(J)
    2  CONTINUE
       E(I)=SUM
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE NORMAL(N,A,B)
C     ******************************************************************
C     * THIS SUBROUTINE NORMALIZES A VECTOR OF LENGTH N.               *
C     ******************************************************************
      REAL*4 A(N),B(N)
      VMAX=SQRT(A(1)*A(1)+B(1)*B(1))
      DO 1 I=1,N
        V=SQRT(A(I)*A(I)+B(I)*B(I))
        IF (V .GT. VMAX) THEN
```

```
          VMAX=V
        ELSE
        END IF
  1   CONTINUE
      DO 2 I=1,N
       A(I)=A(I)/ABS(VMAX)
       B(I)=B(I)/ABS(VMAX)
2     CONTINUE
      RETURN
      END
C
      SUBROUTINE GAIN(NE,U,V,X,Y,GDB)
C     ***************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                        *
C     ***************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE)
      RAD=.17453293E-01
      PI=.3141593E+01
      TP .0283185E+01
C     ***************************************************************
C     * GRAPH INPUTS                                                *
C     ***************************************************************
      NP=41
      NR=31
      PMIN=0.0
      PMAX=360.0
      TMIN=0.0
      TMAX=90.0
      DP=(PMAX-PMIN)/(NP-1)
      DR=(TMAX-TMIN)/(NR-1)
C         ***********************************************************
C         * COMPUTE PEAK VALUE.                                     *
C         ***********************************************************
           CALL POWER(NE,U,V,X,Y,0.0,0.0,PEAK)
C         ***********************************************************
C         * COMPUTE THE SOLID ANGLE.                                *
C         ***********************************************************
           SUM=0.0
           DO 1 I=0,NR-2
             T=TMIN+DR/2.0+I*DR
             ST=SIN(RAD*T)
             DO 2 J=0,NP-2
               P=PMIN+DP/2.0+J*DP
               CALL POWER(NE,U,V,X,Y,T,P,POW)
               SUM=SUM+ST*POW
2            CONTINUE
1          CONTINUE
           TEST=RAD*RAD*DR*DP*SUM
           SOLID=RAD*RAD*DR*DP*SUM/PEAK
```

```
785            GDB=10.0*ALOG10(TP/SOLID)
786        RETURN
787        END
788 C
789        SUBROUTINE POWPAT(NE,U,V,X,Y)
790 C     ****************************************************************
791 C     * COMPUTE POWER IN A SINGLE DIRECTION.                         *
792 C     ****************************************************************
793        REAL U(NE),V(NE),X(NE),Y(NE)
794        OPEN(UNIT=7,FILE='DATA')
795        RAD=.17453293E-01
796        PI=.3141593E+01
797        TP=.6283185E+01
798 C     ****************************************************************
799 C     * GRAPH INPUTS                                                 *
800 C     ****************************************************************
801        NP=55
802        NR=30
803        NS=0
804        PMIN=0.0
805        PMAX=360.0
806        TMIN=0.0
807        TMAX=90.0
808        DP=(PMAX-PMIN)/(NP-1)
809        DR=(TMAX-TMIN)/(NR-1)
810        AX=150.0
811        AY=AX
812        AZ=680.0
813        P0=55.0
814        Q0=60.0
815        XA=-2.0
816        XB=2.0
817        YA=-2.0
818        YB=2.0
819        ZA=-2.0
820        ZB=2.3
821        AT=.3
822        WRITE(7,*) NP,NR,NS,AX,AY,AZ
823        WRITE(7,*) XA,XB,YA,YB,ZA,ZB
824        WRITE(7,*) AT,P0,Q0
825 C     ****************************************************************
826 C     * PHI VALUES.                                                  *
827 C     ****************************************************************
828        DO 1 I=0,NP-1
829         P=PMIN+I*DP
830         WRITE(7,*) RAD*P
831   1    CONTINUE
832 C     ****************************************************************
833 C     * RADIAL (THETA) VALUES.                                       *
```

```
C     ****************************************************************
      DO 2 I=0,NR-1
       T=TMIN+I*DR
       WRITE(7,*) RAD*T
 2    CONTINUE
C     ****************************************************************
C     * FUNCTION VALUES.                                             *
C     ****************************************************************
C         ************************************************************
C         * COMPUTE PEAK VALUE.                                      *
C         ************************************************************
          CALL POWER(NE,U,V,X,Y,0.0,0.0,PEAK)
C         ************************************************************
C         * COMPUTE EACH VALUE.                                      *
C         ************************************************************
          DO 3 I=0,NR-1
            T=TMIN+I*DR
            DO 4 J=0,NP-1
              P=PMIN+J*DP
              CALL POWER(NE,U,V,X,Y,T,P,POW)
              WRITE(7,*) POW/PEAK
 4          CONTINUE
 3        CONTINUE
      RETURN
      END
C
      SUBROUTINE SPOWPAT(NE,U,V,X,Y)
C     ****************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                         *
C     ****************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE)
      OPEN(UNIT=8,FILE='SDATA')
      RAD=.17453293E-01
      PI=.3141593E+01
      TP=.6283185E+01
C     ****************************************************************
C     * GRAPH INPUTS                                                 *
C     ****************************************************************
      NP=115
      NR=63
      NS=0
      PMIN=0.0
      PMAX=360.0
      TMIN=0.0
      TMAX=90.0
      DP=(PMAX-PMIN)/(NP-1)
      DR=(TMAX-TMIN)/(NR-1)
      AX=600.0
      AY=AX
```

```
      AZ=AX
      PO=26.0
      QO=60.0
      XA=-2.0
      XB=2.0
      YA=-2.0
      YB=2.0
      ZA=-2.0
      ZB=2.3
      AT=.3
      IT=4
      WRITE(8,*) NP,NR,NS,IT,AX,AY,AZ
      WRITE(8,*) XA,XB,YA,YB,ZA,ZB
      WRITE(8,*) AT,PO,QO
C     ******************************************************************
C     * PHI VALUES.                                                    *
C     ******************************************************************
      DO 1 I=0,NP-1
       P=PMIN+I*DP
       WRITE(8,*) RAD*P
 1    CONTINUE
C     ******************************************************************
C     * RADIAL (THETA) VALUES.                                         *
C     ******************************************************************
      DO 2 I=0,NR-1
       T=TMIN+I*DR
       WRITE(8,*) RAD*T
 2    CONTINUE
C     ******************************************************************
C     * FUNCTION VALUES.                                               *
C     ******************************************************************
C         ***********************************************************
C         * COMPUTE PEAK VALUE.                                     *
C         ***********************************************************
          CALL POWER(NE,U,V,X,Y,0.0,0.0,PEAK)
C         ***********************************************************
C         * COMPUTE EACH VALUE.                                     *
C         ***********************************************************
          DO 3 I=0,NR-1
            T=TMIN+I*DR
            DO 4 J=0,NP-1
              P=PMIN+J*DP
              CALL POWER(NE,U,V,X,Y,T,P,POW)
              WRITE(8,*) POW/PEAK
 4          CONTINUE
 3        CONTINUE
      RETURN
      END
C
```

```
      SUBROUTINE POWER(NE,U,V,X,Y,T,P,POW)
C     ******************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION GIVEN A THETA AND PHI.     *
C     ******************************************************************
      REAL U(NE),V(NE),X(NE),Y(NE),KXI,KYI
      RAD=.17453293E-01
      TP=.6283185E+01
      TPS=TP*SIN(RAD*T)
      RP=RAD*P
      KXI=TPS*COS(RP)
      KYI=TPS*SIN(RP)
      SUM1=0.0
      SUM2=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       CP=COS(PSI)
       SP=SIN(PSI)
       SUM1=SUM1+U(K)*CP-V(K)*SP
       SUM2=SUM2+U(K)*SP+V(K)*CP
1     CONTINUE
      TI=SUM1*SUM1
      VI=SUM2*SUM2
      POW=TI+VI
      RETURN
      END
```

```
      PROGRAM DIFSYN
C     ****************************************************************
C     * THIS PROGRAM SYNTHESIZES THE AMPLITUDE AND PHASE DISTRIBUTION *
C     * NECESSARY TO GENERATE A DIFFERENCE PATTERN OVER A HEXAGONAL   *
C     * ARRAY OF POINT SOURCES WITH SOME ELEMENT FAILURES.            *
C     ****************************************************************
C     * TIMOTHY J. PETERS                                LAST UPDATED *
C     * THE AEROSPACE CORPORATION                          ./91       *
C     * 2350 EAST EL SEGUNDO BOULEVARD.                               *
C     * EL SEGUNDO, CA 90245                                          *
C     ****************************************************************
C     * INPUTS:                                                       *
C     *                                                               *
C     * NL  - NUMBER OF LAYERS FORMING THE HEXAGONAL LATTICE          *
C     * NTE - NUMBER OF TOTAL ELEMENTS =(1+3*NL*(NL+1)-NL)/2 FOR NL ODD *
C     *                                =(3*NL*(NL+1)-NL)/2 FOR NL EVEN *
C     * NBE - NUMBER OF BAD ELEMENTS                                  *
C     * BAD(1..NBE) - INTEGER ARRAY HOLDING THE NUMBER POSITION OF EACH *
C     *       BAD ELEMENT IN THE ARRAY.                               *
C     * DS  - THE ELEMENT SPACING IN WAVELENGTHS.                     *
C     * NE  - NUMBER OF ACTUAL ELEMENTS = NTE-NBE.                    *
C     * SLL - SIDELOBE LEVEL OF DESIGN BAYLISS DISTRIBUTION.          *
C     * NBAR- PARAMETER FOR BAYLISS DISTRIBUTION.                     *
C     * BBF - THE BEAM WIDTH BROADENING FACTOR.  THIS ALLOWS THE BEAM *
C     *       TO BROADEN BEYOND THE DESIGN VALUE IN ORDER TO REDUCE THE *
C     *       SIDELOBES WHEN ELEMENTS FAIL.                           *
C     * NSP - NUMBER OF SAMPLE POINTS IN THE SIDELOBE REGION.         *
C     * NS  - NUMBER OF RESTARTS.                                     *
C     * NI  - NUMBER OF CONJUGATE GRADIENT ITERATIONS PER RESTART.    *
C     *                                                               *
C     * OUTPUTS:                                                      *
C     *                                                               *
C     * U(NE) - REAL PART OF THE EXCITATION OF EACH ELEMENT.          *
C     * V(NE) - IMAGINARY PART OF THE EXCITATION OF EACH ELEMENT.     *
C     ****************************************************************
      PARAMETER (NL=6,NTE=43,NBE=0,NE=43,NSP=700)
      REAL*4 U(NE),X(NE),Y(NE),S(NE),GN(NE),H(NE,NE),P(NE),Q(NE),R(NE)
      REAL*4 Z(NE),KX(NSP),KY(NSP),KXP,KYP,NWBP
      INTEGER IP(NSP),BAD(NE)
      PI=.3141593E+01
      OPEN(UNIT=2,FILE='OUTPUT',STATUS='UNKNOWN')
      OPEN(UNIT=4,FILE='ESTIMATE',STATUS='UNKNOWN')
C     ****************************************************************
C     * GENERATE THE GEOMETRY.                                        *
C     ****************************************************************
      DS=0.6
      BAD(1)=15
      BAD(2)=17
      BAD(3)=20
```

```
      CALL GEOMET(NL,DS,NE,NBE,BAD,X,Y)
C     ******************************************************************
C     * INITIALIZE THE TAPER DISTRIBUTION.                             *
C     ******************************************************************
      IFLAG1=1
      SLL=-40.0
      NBAR=10
      IF (IFLAG1 .EQ. 1) THEN
        CALL TAPER(NE,U,X,Y,SLL,NBAR,NL,DS)
        REWIND 4
        WRITE(4,101) (I,U(I),I=1,NE)
      ELSE
      END IF
      REWIND 4
      READ(4,101) (I,U(I),I=1,NE)
C     ******************************************************************
C     * COMPUTE THE GAIN (DIRECTIVITY) OF THE ARRAY.                   *
C     ******************************************************************
C       CALL GAIN(NE,U,X,Y,GDB)
C     ******************************************************************
C     * FIND THE HALF POWER NULL WIDTH BETWEEN PEAKS.                  *
C     ******************************************************************
C       CALL BEAM(NE,U,X,Y,NWBP)
C     ******************************************************************
C     * GENERATE THE K-SPACE SAMPLE POINTS.                            *
C     ******************************************************************
      BBF=1.24
      CALL KSPACE(NMP,NL,DS,SLL,NBAR,BBF,KX,KY)
C     ******************************************************************
C     * COMPUTE NORMALIZED PEAK SIDELOBE POWER IN DB.                  *
C     ******************************************************************
      CALL PEKPOW(NE,U,X,Y,KXP,KYP,PO)
      NP=0
      CALL PEAKSL(NE,U,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
      PMAXDB=10.0*ALOG10(PMAX/PO)
C     ******************************************************************
C     * PERFORM CONJUGATE GRADIENT STEPS.                              *
C     ******************************************************************
      NS=150
      NI=3
      CALL CGRAD(NE,NSP,NMP,NS,NI,U,X,Y,S,NP,IP,KX,KY,GN,H,P,Q,R,Z)
      REWIND 4
      WRITE(4,101) (I,U(I),I=1,NE)
C     ******************************************************************
C     * POWER PATTERN GRAPHICS OUTPUT.                                 *
C     ******************************************************************
      REWIND 4
      READ(4,101) (I,U(I),I=1,NE)
C       ****************************************************************
```

```
C       * PRINT THE POWER PATTERN IN CYLINDRICAL COORDINATES.          *
C       ****************************************************************
         CALL POWPAT(NE,U,X,Y)
C       ****************************************************************
C       * PRINT THE POWER PATTERN IN SPHERICAL COORDINATES.             *
C       ****************************************************************
         CALL SPOWPAT(NE,U,X,Y)
C     ******************************************************************
C     * FORMATS.                                                       *
C     ******************************************************************
100   FORMAT(I5,F15.5,1X,F15.5)
101   FORMAT(I5,F15.7)
      END
C
      SUBROUTINE GEOMET(NL,S,NE,NBE,BAD,X,Y)
C     ******************************************************************
C     * COMPUTE THE POSITION OF EACH ELEMENT IN A HEXOGONAL ARRAY.     *
C     ******************************************************************
      REAL*4 X(NE),Y(NE)
      INTEGER BAD(NE),IFLAG
      HGT=(SQRT(3.0)/2.0)*S
C     ******************************************************************
C     * CENTER ROW.                                                    *
C     ******************************************************************
      K=0
      L=0
      XMIN=-NL*S
      DO 1 I=0,2*NL
       XX=XMIN+I*S
       IF (XX .GT. 0.001) THEN
         L=L+1
         CALL CHECK(L,NE,NBE,BAD,IFLAG)
         IF (IFLAG .EQ. 0) THEN
           K=K+1
           X(K)=XX
           Y(K)=0.0
         ELSE
         END IF
       ELSE
       END IF
1     CONTINUE
C     ******************************************************************
C     * TOP ROWS.                                                      *
C     ******************************************************************
      INUM=2*NL+1
      DO 2 J=1,NL
        INUM=INUM-1
        XMIN=-NL*S+J*S/2.0
        DO 3 I=0,INUM-1
```

```
          XX=XMIN+I*S
          IF (XX .GT. 0.001) THEN
            L=L+1
            CALL CHECK(L,NE,NBE,BAD,IFLAG)
            IF (IFLAG .EQ. 0) THEN
              K=K+1
              X(K)=XX
              Y(K)=HGT*J
            ELSE
            END IF
          ELSE
          END IF
3       CONTINUE
2     CONTINUE
C     ***************************************************************
C     * BOTTOM ROWS.                                                *
C     ***************************************************************
      INUM=2*NL+1
      DO 4 J=1,NL
        INUM=INUM-1
        XMIN=-NL*S+J*S/2.0
        DO 5 I=0,INUM-1
          XX=XMIN+I*S
          IF (XX .GT. 0.001) THEN
            L=L+1
            CALL CHECK(L,NE,NBE,BAD,IFLAG)
            IF (IFLAG .EQ. 0) THEN
              K=K+1
              X(K)=XX
              Y(K)=-HGT*J
            ELSE
            END IF
          ELSE
          END IF
5       CONTINUE
4     CONTINUE
      RETURN
      END
C
      SUBROUTINE CHECK(L,NE,NBE,BAD,IFLAG)
C     ***************************************************************
C     * CHECK TO SEE IF THAT ELEMENT IS BAD.                        *
C     ***************************************************************
      INTEGER BAD(NE)
      IFLAG=0
      DO 1 I=1,NBE
         IF (L .EQ. BAD(I)) THEN
            IFLAG=1
         ELSE
```

```
197          END IF
198 1      CONTINUE
199        RETURN
200        END
201 C
202        SUBROUTINE BEAM(NE,U,X,Y,BWFN)
203 C     ****************************************************************
204 C     * COMPUTE THE BEAM WIDTH BETWEEN FIRST NULLS.                  *
205 C     ****************************************************************
206        REAL U(NE),X(NE),Y(NE),KXI,KYI,KIP,KYP
207        RAD=.17453293E-01
208        TP=.6283185E+01
209        TMIN=0.0
210        TMAX=12.0
211        P=0.0
212        NT=140
213        DT=(TMAX-TMIN)/NT
214 C     ****************************************************************
215 C     * FIND THE MAXIMUM VALUE.                                      *
216 C     ****************************************************************
217        CALL PEKPOW(NE,U,X,Y,KIP,KYP,PEAK)
218 C     ****************************************************************
219 C     * RECOMPUTE AND NOW FIND THE POINT WHICH IS AT THE SLL BELOW THE *
220 C     * PEAK.                                                        *
221 C     ****************************************************************
222        DO 3 I=0,NT
223          T=TMIN+I*DT
224          TPS=TP*SIN(RAD*T)
225          KXI=TPS
226          CALL POWI(NE,U,X,Y,KXI,0.0,POW)
227          IF (POW/PEAK .GE. 0.5) THEN
228            WRITE(2,100) T,10.0*ALOG(POW/PEAK)
229            GO TO 99
230          ELSE
231          END IF
232  3     CONTINUE
233  100   FORMAT(F10.5,1X,F10.5)
234 99     CONTINUE
235        BWFN=2.0*T
236        RETURN
237        END
238 C
239        SUBROUTINE KSPACE(NMP,NL,DS,SLL,NBAR,BBF,KX,KY)
240 C     ****************************************************************
241 C     * THIS SUBROUTINE SAMPLES THE KX >0 REGION OF K SPACE.         *
242 C     ****************************************************************
243        REAL*4 KX(*),KY(*),KC,KR,KXX,KYY,KXMIN,KXMAX,KYMIN,KYMAX,KNULL
244        RAD=.17453293E-01
245        PI=.3141593E+01
```

```
      TP=.6283185E+01
C     ******************************************************************
C     * COMPUTE THE FIRST ZERO OF THE THEORETICAL BAYLISS PATTERN.     *
C     ******************************************************************
      S=ABS(SLL)
      C1=3.038753E-01
      C2=5.042922E-02
      C3=-2.7989E-04
      C4=3.43E-06
      C5=-2.0E-8
      A=C1+S*(C2+S*(C3+S*(C4+S*C5)))
      C1=9.858302E-01
      C2=3.33885E-02
      C3=1.4064E-04
      C4=-1.9E-06
      C5=1.0E-08
      ZETA1=C1+S*(C2+S*(C3+S*(C4+S*C5)))
      U1=(NBAR+0.5)*SQRT(ZETA1*ZETA1/(A*A+NBAR))
C     ******************************************************************
C     * COMPUTE THE K VALUE OF THE FIRST NULL.                         *
C     ******************************************************************
      R=NL*DS
      KNULL=U1/R
C     ******************************************************************
C     * COMPUTE THE MAXIMUM K VALUE ALLOWED FOR THE FIRST NULL.        *
C     ******************************************************************
C       KC=TP*0.45
      KC=BBF*KNULL
C     ******************************************************************
C     * GENERATE THE RECTANGULAR LATTICE OF SAMPLE POINTS.             *
C     ******************************************************************
      KYMIN=-TP
      KYMAX=TP
      KXMIN=0.0
      KXMAX=TP
      NKX=27
      NKY=45
      DKX=(KXMAX-KXMIN)/(NKX-1)
      DKY=(KYMAX-KYMIN)/(NKY-1)
      NMP=0
      DO 1 I=0,NKY
        KYY=KYMIN+I*DKY
        DO 2 J=0,NKX
          KXX=KXMIN+J*DKX
          KR=SQRT(KXX*KXX+KYY*KYY)
          IF ((KR .GT. KC) .AND. (KR .LT. TP)
     &                                    .AND. (KXX .GT. 0.0)) THEN
            NMP=NMP+1
            KX(NMP)=KXX
```

```
295             KY(NMP)=KYY
296           ELSE
297           END IF
298   2    CONTINUE
299   1   CONTINUE
300       RETURN
301       END
302 C
303       SUBROUTINE TAPER(NE,U,X,Y,SLL,NBAR,NL,DS)
304 C     ****************************************************************
305 C     * INITIALIZE THE AMPLITUDE DISTRIBUTION.                       *
306 C     * NOTE THAT THE AMPLITUDE=U*U. THE BAYLISS DISTRIBUTION FORMULAS *
307 C     * ARE TAKEN FROM MODERN ANTENNA DESIGN BY T. MILLIGAN.         *
308 C     ****************************************************************
309       REAL*4 U(NE),X(NE),Y(NE),ROOT(20),ZETA(4),UZER(20),B(20)
310       PI=.3141593E+01
311 C     ****************************************************************
312 C     * DEFINE THE ROOTS.                                            *
313 C     ****************************************************************
314       ROOT(1)=0.5860670
315       ROOT(2)=1.6970509
316       ROOT(3)=2.7171939
317       ROOT(4)=3.7261370
318       ROOT(5)=4.7312271
319       ROOT(6)=5.7345205
320       ROOT(7)=6.7368221
321       ROOT(8)=7.7385356
322       ROOT(9)=8.7398505
323       ROOT(10)=9.740895
324       ROOT(11)=10.7417435
325       ROOT(12)=11.7424475
326       ROOT(13)=12.7430408
327       ROOT(14)=13.7435477
328       ROOT(15)=14.7439856
329       ROOT(16)=15.7443679
330       ROOT(17)=16.7447044
331       ROOT(18) 17.7450030
332       ROOT(19)=18.7452697
333       ROOT(20)=19.7455093
334 C     ****************************************************************
335 C     * COMPUTE APPROXIMATE ARRAY RADIUS.                            *
336 C     ****************************************************************
337       RADIUS=DS*NL
338 C     ****************************************************************
339 C     * CHOOSE SIDELOBE LEVEL.                                       *
340 C     ****************************************************************
341       CALL COEFF(SLL,A,ZETA)
342       CALL NEWZ(A,ZETA,NBAR,ROOT,UZER)
343       CALL FBCOF(ROOT,UZER,NBAR,B)
```

```
C     ******************************************************************
C     * COMPUTE AMPLITUDE.                                             *
C     ******************************************************************
      DO 1 K=1,NE
       R=SQRT(X(K)*X(K)+Y(K)*Y(K))
       COSPHI=X(K)/R
       SUM=0.0
       DO 2 M=1,NBAR
         RHO=R/RADIUS
         ARG=PI*ROOT(M)*RHO
         CALL J1(ARG,BESS1)
         SUM=SUM+B(M)*BESS1
2      CONTINUE
       AMP=COSPHI*SUM
       U(K)=AMP
1     CONTINUE
      CALL NORMAL(NE,U)
      DO 3 I=1,NE
      WRITE(2,*) I,X(I),Y(I),R,RADIUS,U(I)
3     CONTINUE
100   FORMAT(I5,1X,F10.4,1X,F10.4,1X,F10.4)
      RETURN
      END
C
      SUBROUTINE COEFF(SLL,A,ZETA)
      REAL*4 ZETA(4)
      S=ABS(SLL)
      C1=3.038763E-01
      C2=5.042922E-02
      C3=-2.7989E-04
      C4=3.43E-06
      C5=-2.0E-8
      A=C1+S*(C2+S*(C3+S*(C4+S*C5)))
      C1=9.858302E-01
      C2=3.33885E-02
      C3=1.4064E-04
      C4=-1.9E-06
      C5=1.0E-08
      ZETA(1)=C1+S*(C2+S*(C3+S*(C4+S*C5)))
      C1=2.00337487
      C2=1.141548E-02
      C3=4.169E-04
      C4=-3.73E-06
      C5=1.0E-08
      ZETA(2)=C1+S*(C2+S*(C3+S*(C4+S*C5)))
      C1=3.00636321
      C2=6.83394E-03
      C3=2.9281E-04
      C4=-1.61E-06
```

```
      ZETA(3)=C1+S*(C2+S*(C3+S*C4))
      C1=4.00518432
      C2=5.01795E-03
      C3=2.1735E-04
      C4=-8.8E-07
      ZETA(4)=C1+S*(C2+S*(C3+S*C4))
      RETURN
      END
C
      SUBROUTINE NEWZ(A,ZETA,NBAR,ROOT,UZER)
      REAL*4 ZETA(4),ROOT(20),UZER(20)
      DO 1 N=1,4
        ARG=ZETA(N)*ZETA(N)/(A*A+NBAR*NBAR)
        UZER(N)=ROOT(NBAR+1)*SQRT(ARG)
1     CONTINUE
      DO 2 N=5,NBAR-1
        ARG=(A*A+N*N)/(A*A+NBAR*NBAR)
        UZER(N)=ROOT(NBAR+1)*SQRT(ARG)
2     CONTINUE
      RETURN
      END
C
      SUBROUTINE FBCOF(ROOT,UZER,NBAR,B)
      REAL*4 ROOT(20),UZER(20),B(20)
      REAL*8 RAT,UN2,RM2,RN2,PR1,PR2
      PI=3.141593
      DO 1 M=1,NBAR
        RM2=ROOT(M)*ROOT(M)
        PR1=1.0
        DO 2 N=1,NBAR-1
           UN2=UZER(N)*UZER(N)
           RAT=1.0-RM2/UN2
           PR1=PR1*RAT
2       CONTINUE
        PR2=1.0
        DO 3 N=1,NBAR
           IF (N .NE. M) THEN
             RN2=ROOT(N)*ROOT(N)
             RAT=1.0-RM2/RN2
             PR2=PR2*RAT
           ELSE
           END IF
3       CONTINUE
        ARG=PI*ROOT(M)
        CALL J1(ARG,BESS1)
        B(M)=(RM2/BESS1)*(PR1/PR2)
1     CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE J1(X,BESS1)
C*********************************************************************C
C THIS SUBROUTINE COMPUTES J1(X)                                       C
C*********************************************************************C
      A0=-0.56249985
      A1=0.21093573
      A2=-0.03954289
      A3=0.00443319
      A4=-0.00031761
      A5=0.00001109
      B0=0.79788456
      B1=0.00000156
      B2=0.01659667
      B3=0.00017105
      B4=-0.00249511
      B5=0.00113653
      B6=-0.00020033
      C0=-2.35619449
      C1=0.12499612
      C2=0.00005650
      C3=-0.00637879
      C4=0.00074348
      C5=0.00079824
      C6=-0.00029166
      IF ((X .GE. 0.0) .AND. (X .LE. 3.0)) THEN
        S=(X/3.0)*(X/3.0)
        P1=S*(A0+S*(A1+S*(A2+S*(A3+S*(A4+A5*S)))))
        BESS1=X*(0.5+P1)
      ELSE IF (X .GT. 3.0) THEN
        T=3.0/X
        Q1=B0+T*(B1+T*(B2+T*(B3+T*(B4+T*(B5+B6*T)))))
        V1=X+C0+T*(C1+T*(C2+T*(C3+T*(C4+T*(C5+C6*T)))))
        BESS1=Q1*COS(V1)/SQRT(X)
      ELSE
      END IF
      RETURN
      END
C
      SUBROUTINE CGRAD(NE,NSP,NMP,NS,NI,U,X,Y,S,NP,IP,KX,KY,GN,H,P,Q,R
     &,Z)
C     ******************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                           *
C     ******************************************************************
      REAL*4 U(NE),X(NE),Y(NE),S(NE),GN(NE),H(NE,NE),P(NE),Q(NE),R(NE)
      REAL*4 Z(NE),KX(NSP),KY(NSP),KXI,KYI,KXP,KYP
      INTEGER IP(NSP)
C     ******************************************************************
C     * INITIALIZE THE NUMBER OF SIDELOBE SAMPLE POINTS.               *
```

```
C     ***************************************************************
      NP=0
C     ***************************************************************
C     * PERFORM NS RE-STARTS.                                       *
C     ***************************************************************
      DO 1 L=1,NS
C         ***********************************************************
C         * FIND THE PEAK SIDELOBE POWER POINT AND UPDATE THE NUMBER *
C         * OF MATCH POINTS.                                        *
C         ***********************************************************
          CALL PEAKSL(NE,U,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
C         ***********************************************************
C         * FIND THE POWER IN THE MAIN BEAM.                        *
C         ***********************************************************
          CALL PEKPOW(NE,U,X,Y,KXP,KYP,PEAK)
          PO=PEAK
C         ***********************************************************
C         * COMPUTE AND STORE AN ARRAY OF THE 1ST DERIVATIVE OF THE *
C         * POWER IN THE MAIN BEAM DIRECTION WITH RESPECT TO THE MTH *
C         * VARIABLE.                                               *
C         ***********************************************************
          DO 2 M=1,NE
            CALL DPOWI(NE,U,X,Y,M,KXP,KYP,DPWI)
            S(M)=DPWI
2         CONTINUE
C         ***********************************************************
C         * COMPUTE AND STORE THE AVERAGE SIDELOBE POWER.           *
C         ***********************************************************
          PA=0.0
          DO 3 I=1,NP
            CALL POWI(NE,U,X,Y,KX(IP(I)),KY(IP(I)),POWERI)
            PA=PA+POWERI
3         CONTINUE
          PA=PA/NP
          WRITE(*,200) L,10.0*ALOG10(PA/PO),10.0*ALOG10(PMAX/PO)
          WRITE(2,200) L,10.0*ALOG10(PA/PO),10.0*ALOG10(PMAX/PO)
200       FORMAT(1X,I5,1X,'AVE PEAK SLL',1X,F15.6,1X,
     &                                 'PEAK SLL',1X,F15.6)
C         ***********************************************************
C         * COMPUTE THE NEGATIVE OF THE GRADIENT.                   *
C         ***********************************************************
          DO 4 M=1,NE
C             *******************************************************
C             * COMPUTE THE DERIVATIVE OF THE AVERAGE SIDELOBE POWER. *
C             *******************************************************
              DPAM=0.0
              DO 5 I=1,NP
                CALL DPOWI(NE,U,X,Y,M,KX(IP(I)),KY(IP(I)),DPWI)
                DPAM=DPAM+DPWI
```

```
5              CONTINUE
               DPAM=DPAM/NP
C             ***********************************************************
C             * COMPUTE THE NEGATIVE GRADIENT.                          *
C             ***********************************************************
               GN(M)=(PA*S(M)/PO-DPAM)/PO
4          CONTINUE
C         ***********************************************************
C         * COMPUTE THE HESSIAN.                                      *
C         ***********************************************************
           DO 6 M=1,NE
             DO 7 N=M,NE
               CALL DDPOWI(NE,U,X,Y,M,N,KXP,KYP,DDPWI)
               DDPO=DDPWI
               DDPA=0.0
               DO 8 I=1,NP
                 CALL DDPOWI(NE,U,X,Y,M,N,KX(IP(I)),KY(IP(I)),DDPWI)
                 DDPA=DDPA+DDPWI
8              CONTINUE
               DDPA=DDPA/NP
               H(M,N)=(GN(N)*S(M)+GN(M)*S(N)+DDPA-PA*DDPO/PO)/PO
               H(N,M)=H(M,N)
7           CONTINUE
6          CONTINUE
C         ***********************************************************
C         * START THE CONJUGATE GRADIENT ALGORITHM.                   *
C         ***********************************************************
C         ***********************************************************
C         * INITIALIZE THE RESIDUAL.                                  *
C         ***********************************************************
           DO 12 M=1,NE
             R(M)=GN(M)
             Z(M)=0.0
12         CONTINUE
C         ***********************************************************
C         * INITIALIZE SEARCH VECTOR.                                 *
C         ***********************************************************
           CALL MATVEC(NE,H,R,Q)
           CALL NORM22(NE,Q,QN)
           BEO=1.0/QN
           DO 13 I=1,NE
             P(I)=BEO*Q(I)
13         CONTINUE
C         ***********************************************************
C         * PERFORM CONJUGATE GRADIENT ITERATIONS.                    *
C         ***********************************************************
           DO 14 K=1,NI
C         ***********************************************************
C         * UPDATE AMPLITUDE VECTOR AND RESIDUAL.                     *
```

```
C          ***************************************************************
            CALL MATVEC(NE,H,P,Q)
            CALL NORM22(NE,Q,QN)
            AK=1.0/QN
            DO 15 I=1,NE
              Z(I)=Z(I)+AK*P(I)
              R(I)=R(I)-AK*Q(I)
  15        CONTINUE
            CALL NORM22(NE,R,RN)
            ERR=SQRT(RN)
            WRITE(*,500) L,K,ERR
            WRITE(2,500) L,K,ERR
500         FORMAT(3X,'RESIDUAL',1X,I3,1X,I3,1X,E15.8)
C          ***************************************************************
C          * UPDATE SEARCH VECTOR.                                       *
C          ***************************************************************
            CALL MATVEC(NE,H,R,Q)
            CALL NORM22(NE,Q,QN)
            BEK=1.0/QN
            DO 16 I=1,NE
              P(I)=P(I)+BEK*Q(I)
  16        CONTINUE
14          CONTINUE
C          ***************************************************************
C          * UPDATE AMPLITUDE VECTOR.                                    *
C          ***************************************************************
            DO 17 I=1,NE
              U(I)=U(I)+Z(I)
  17        CONTINUE
            CALL NORMAL(NE,U)
C          ***************************************************************
C          * WRITE OUT THE NEW ESTIMATE TO A FILE.      *
C          ***************************************************************
          REWIND 4
          WRITE(4,101) (I,U(I),I=1,NE)
101       FORMAT(I5,F15.7)
        IF (NP .EQ. NMP) THEN
          GO TO 99
        ELSE
        END IF
1       CONTINUE
99      CONTINUE
        RETURN
        END
C
        SUBROUTINE PEAKSL(NE,U,X,Y,NSP,NMP,KX,KY,IP,NP,PMAX)
C     ***************************************************************
C     * THIS SUBROUTINE FINDS THE PEAK SIDELOBE POWER LEVEL COORDINATE  *
C     * POINT (KX(LMAX),KY(LMAX)).  IF THE POINT IS NEW THEN IT IS      *
```

```
C     * ACCUMULATED INTO THE POINTER ARRAY IP. THE NUMBER OF POINTS IN  *
C     * THE INTEGER POINTER ARRAY IP IS NP SUCH THAT IP(I),I=1...NP.    *
C     ******************************************************************
      REAL U(NE),X(NE),Y(NE) KX(NSP),KY(NSP)
      INTEGER IP(NSP)
C         ***********************************************************
C         * COMPUTE THE POWER AT EACH POINT AND FIND THE MAX.       *
C         ***********************************************************
          CALL POWI(NE,U,X,Y,KX(1),KY(1),PWI)
          PMAX=PWI
          LMAX=1
          DO 1 I=1,NMP
            CALL POWI(NE,U,X,Y,KX(I),KY(I),PWI)
            IF (PWI .GT. PMAX) THEN
              LMAX=I
              PMAX=PWI
            ELSE
            END IF
1         CONTINUE
C         ***********************************************************
C         * COMPARE LMAX TO ALL THE PREVIOUS POINTERS IP(I),I=1...NP. *
C         * IF LMAX MATCHES ONE OF THE PREVIOUS POINTERS THEN THE    *
C         * POINTER ARRAY NEED NOT BE UPDATED (IFLAG=1).  IF LMAX DOES *
C         * NOT MATCH A PREVIOUS POINTER VALUE (IFLAG=0) THEN IT     *
C         * BECOMES THE NP+1TH ELEMENT OF THE POINTER ARRAY.         *
C         ***********************************************************
          IFLAG=0
          DO 2 I=1,NP
            IF (IP(I) .EQ. LMAX) THEN
              IFLAG=1
            ELSE
            END IF
2         CONTINUE
          IF (IFLAG .EQ. 0) THEN
            NP=NP+1
            IP(NP)=LMAX
          ELSE
          END IF
      RETURN
      END
C
      SUBROUTINE POWI(NE,U,X,Y,KXI,KYI,POWERI)
C     ******************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                           *
C     ******************************************************************
      REAL U(NE),X(NE),Y(NE),KXI,KYI
      SUM=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
```

```
      SUM=SUM+U(K)*SIN(PSI)
1     CONTINUE
      POWERI=SUM*SUM
      RETURN
      END
C
      SUBROUTINE DPOWI(NE,U,X,Y,M,KXI,KYI,DPWI)
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF POWER WITH RESPECT TO THE MTH VARIABLE*
C     * IN THE ITH DIRECTION.                                          *
C     ****************************************************************
      REAL U(NE),X(NE),Y(NE),KXI,KYI
C     ****************************************************************
C     * COMPUTE TI AND VI.                                             *
C     ****************************************************************
      SUM=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       SUM=SUM+U(K)*SIN(PSI)
1     CONTINUE
C     ****************************************************************
C     * COMPUTE 1ST DERIVATIVE OF TI AND VI.                           *
C     ****************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      DPWI=2.0*SIN(PSIM)*SUM
      RETURN
      END
C
      SUBROUTINE DDPOWI(NE,U,X,Y,M,N,KXI,KYI,DDPWI)
C     ****************************************************************
C     * COMPUTE 2ND DERI[illegible] OF POWER WITH RESPECT TO THE M AND NTH  *
C     * VARIABLES IN TH[illegible] DIRECTION.                          *
C     ****************[illegible]*****************************************
      REAL*4 U(NE),X(NE),Y(NE),KXI,KYI
C     ****************************************************************
C     * COMPUTE 2ND DERIVATIVE OF TI AND VI WITH RESPECT TO M,N.       *
C     ****************************************************************
      PSIM=KXI*X(M)+KYI*Y(M)
      PSIN=KXI*X(N)+KYI*Y(N)
C     ****************************************************************
C     * COMPUTE 2ND DERIVATIVE OF POWER.                               *
C     ****************************************************************
      DDPWI=2.0*SIN(PSIM)*SIN(PSIN)
      RETURN
      END
C
      SUBROUTINE NORM22(N,A,AN)
C     ****************************************************************
C     * THIS SUBROUTINE COMPUTES THE EUCLIDIAN NORM SQUARED OF A.      *
```

```
C     ****************************************************************
      REAL*4 A(*)
      AN=0.0
      DO 1 I=1,N
       AN=AN+A(I)*A(I)
1     CONTINUE
      RETURN
      END
C
      SUBROUTINE MATVEC(N,H,P,Q)
C     ****************************************************************
C     * THIS SUBROUTINE COMPUTES THE MATRIX PRODUCT Q=HP.            *
C     ****************************************************************
      REAL*4 H(N,N),P(N),Q(N)
      DO 1 I=1,N
       SUM=0.0
       DO 2 J=1,N
        SUM=SUM+H(I,J)*P(J)
2      CONTINUE
       Q(I)=SUM
1     CONTINUE
      RETURN
      END
C
      SUBROUTINE NORMAL(N,A)
C     ****************************************************************
C     * THIS SUBROUTINE NORMALIZES A VECTOR OF LENGTH N.             *
C     ****************************************************************
      REAL*4 A(N)
      VMAX=A(1)
      DO 1 I=1,N
        IF (A(1) .GT. VMAX) THEN
          VMAX=A(I)
        ELSE
        END IF
1     CONTINUE
      DO 2 I=1,N
       A(I)=A(I)/ABS(VMAX)
2     CONTINUE
      RETURN
      END
C

      SUBROUTINE PEKPOW(NE,U,X,Y,KXP,KYP,PEAK)
C     ****************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION GIVEN A THETA AND PHI.   *
C     ****************************************************************
      REAL U(NE),X(NE),Y(NE),KXI,KYI,KXP,KYP
      RAD=.17453293E-01
```

```
      TP=.6283185E+01
      TMIN=0.0
      TMAX=30.0
      PMIN=-10.0
      PMAX=10.0
      NT=65
      NP=45
      DT=(TMAX-TMIN)/NT
      DP=(PMAX-PMIN)/NP
      DO 1 I=0,NT
        T=TMIN+I*DT
        TPS=TP*SIN(RAD*T)
        DO 2 J=0,NP
          P=PMIN+J*DP
          RP=RAD*P
          KXI=TPS*COS(RP)
          KYI=TPS*SIN(RP)
          CALL POWI(NE,U,X,Y,KXI,KYI,POW)
          IF (I .EQ. 1) THEN
             PEAK=POW
             KXP=KXI
             KYP=KYI
             THEPK=T
             PHIPK=P
          ELSE IF (POW .GT. PEAK) THEN
             PEAK=POW
             KXP=KXI
             KYP=KYI
             THEPK=T
             PHIPK=P
          ELSE
          END IF
2       CONTINUE
1     CONTINUE
      WRITE(*,*) THEPK,PHIPK,KXP,KYP,PEAK
      WRITE(2,*) THEPK,PHIPK,KXP,KYP,PEAK
      RETURN
      END
C
      SUBROUTINE GAIN(NE,U,X,Y,GDB)
C     ***************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                        *
C     ***************************************************************
      REAL U(NE),X(NE),Y(NE),KXP,KYP
      RAD=.17453293E-01
      PI=.3141593E+01
      TP=.6283185E+01
C     ***************************************************************
C     * GRAPH INPUTS                                                *
```

```
C     ****************************************************************
      NP=41
      NR=35
      PMIN=0.0
      PMAX=360.0
      TMIN=0.0
      TMAX=90.0
      DP=(PMAX-PMIN)/(NP-1)
      DR=(TMAX-TMIN)/(NR-1)
C         ************************************************************
C         * COMPUTE PEAK VALUE.                                      *
C         ************************************************************
          CALL PEKPOW(NE,U,X,Y,KXP,KYP,PEAK)
C         ************************************************************
C         * COMPUTE THE SOLID ANGLE.                                 *
C         ************************************************************
          SUM=0.0
          DO 1 I=0,NR-2
            T=TMIN+DR/2.0+I*DR
            ST=SIN(RAD*T)
            DO 2 J=0,NP-2
              P=PMIN+DP/2.0+J*DP
              CALL POWER(NE,U,X,Y,T,P,POW)
              SUM=SUM+ST*POW
2           CONTINUE
1         CONTINUE
          SOLID=RAD*RAD*DR*DP*SUM/PEAK
          GDB=10.0*ALOG10(TP/SOLID)
      RETURN
      END
C
      SUBROUTINE POWPAT(NE,U,X,Y)
C     ****************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION.                         *
C     ****************************************************************
      REAL U(NE),X(NE),Y(NE),KXP,KYP
      OPEN(UNIT=7,FILE='DATA')
      RAD=.17453293E-01
      PI=.3141593E+01
      TP=.6283185E+01
C     ****************************************************************
C     * GRAPH INPUTS                                                 *
C     ****************************************************************
      NP=41
      NR=25
      NS=0
      PMIN=0.0
      PMAX=360.0
      TMIN=0.0
```

```
      TMAX=90.0
      DP=(PMAX-PMIN)/(NP-1)
      DR=(TMAX-TMIN)/(NR-1)
      AX=150.0
      AY=AX
      AZ=680.0
      PO=65.0
      QO=60.0
      XA=-2.0
      XB=2.0
      YA=-2.0
      YB=2.0
      ZA=-2.0
      ZB=2.3
      AT=.3
      WRITE(7,*) NP,NR,NS,AX,AY,AZ
      WRITE(7,*) XA,XB,YA,YB,ZA,ZB
      WRITE(7,*) AT,PO,QO
C     ******************************************************************
C     * PHI VALUES.                                                    *
C     ******************************************************************
      DO 1 I=0,NP-1
       P=PMIN+I*DP
       WRITE(7,*) RAD*P
 1    CONTINUE
C     ******************************************************************
C     * RADIAL (THETA) VALUES.                                         *
C     ******************************************************************
      DO 2 I=0,NR-1
       T=TMIN+I*DR
       WRITE(7,*) RAD*T
 2    CONTINUE
C     ******************************************************************
C     * FUNCTION VALUES.                                               *
C     ******************************************************************
C         ***********************************************************
C         * COMPUTE PEAK VALUE.                                     *
C         ***********************************************************
          CALL PEKPOW(NE,U,X,Y,KXP,KYP,PEAK)
C         ***********************************************************
C         * WRITE OUT THE NORMALIZED POWER.                         *
C         ***********************************************************
          DO 5 I=0,NR-1
            T=TMIN+I*DR
            DO 6 J=0,NP-1
              P=PMIN+J*DP
              CALL POWER(NE,U,X,Y,T,P,POW)
              WRITE(7,*) POW/PEAK
 6          CONTINUE
```

```
932 5          CONTINUE
933        RETURN
934        END
935 C
936        SUBROUTINE SPOWPAT(NE,U,X,Y)
937 C     ******************************************************************
938 C     * COMPUTE POWER IN A SINGLE DIRECTION.                           *
939 C     ******************************************************************
940        REAL U(NE),X(NE),Y(NE)
941        OPEN(UNIT=8,FILE='SDATA')
942        RAD=.17453293E-01
943        PI=.3141593E+01
944        TP=.6283185E+01
945 C     ******************************************************************
946 C     * GRAPH INPUTS                                                   *
947 C     ******************************************************************
948        NP=119
949        NR=53
950        NS=0
951        PMIN=0.0
952        PMAX=360.0
953        TMIN=0.0
954        TMAX=90.0
955        DP=(PMAX-PMIN)/(NP-1)
956        DR=(TMAX-TMIN)/(NR-1)
957        AX=600.0
958        AY=AX
959        AZ=AX
960        PO=65.0
961        QO=60.0
962        XA=-2.0
963        XB=2.0
964        YA=-2.0
965        YB=2.0
966        ZA=-2.0
967        ZB=2.3
968        AT=.3
969        IT=4
970        WRITE(8,*) NP,NR,NS,IT,AX,AY,AZ
971        WRITE(8,*) XA,XB,YA,YB,ZA,ZB
972        WRITE(8,*) AT,PO,QO
973 C     ******************************************************************
974 C     * PHI VALUES.                                                    *
975 C     ******************************************************************
976        DO 1 I=0,NP-1
977         P=PMIN+I*DP
978         WRITE(8,*) RAD*P
979  1     CONTINUE
980 C     ******************************************************************
```

```
C     * RADIAL (THETA) VALUES.                                          *
C     ******************************************************************
      DO 2 I=0,NR-1
       T=TMIN+I*DR
       WRITE(8,*) RAD*T
 2    CONTINUE
C     ******************************************************************
C     * FUNCTION VALUES.                                                *
C     ******************************************************************
C          ***********************************************************
C          * COMPUTE PEAK VALUE.                                      *
C          ***********************************************************
           PEAK=-100.0
           DO 3 I=0,NR-1
             T=TMIN+I*DR
             DO 4 J=0,NP-1
               P=PMIN+J*DP
               CALL POWER(NE,U,X,Y,T,P,POW)
               IF (POW .GT. PEAK) THEN
                  PEAK=POW
               ELSE
               END IF
 4           CONTINUE
 3         CONTINUE
C          ***********************************************************
C          * WRITE OUT THE NORMALIZED POWER.                          *
C          ***********************************************************
           DO 5 I=0,NR-1
             T=TMIN+I*DR
             DO 6 J=0,NP-1
               P=PMIN+J*DP
               CALL POWER(NE,U,X,Y,T,P,POW)
               WRITE(8,*) POW/PEAK
 6           CONTINUE
 5         CONTINUE
      RETURN
      END
C
      SUBROUTINE POWER(NE,U,X,Y,T,P,POW)
C     ******************************************************************
C     * COMPUTE POWER IN A SINGLE DIRECTION GIVEN A THETA AND PHI.      *
C     ******************************************************************
      REAL U(NE),X(NE),Y(NE),KXI,KYI
      RAD=.17453293E-01
      TP=.6283185E+01
      TPS=TP*SIN(RAD*T)
      RP=RAD*P
      KXI=TPS*COS(RP)
      KYI=TPS*SIN(RP)
```

```
      SUM=0.0
      DO 1 K=1,NE
       PSI=KXI*X(K)+KYI*Y(K)
       SUM=SUM+U(K)*SIN(PSI)
    1 CONTINUE
      POW=SUM*SUM
      RETURN
      END
```

END

FILMED

DATE:
10-91

DTIC