(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ELECTROMAGNETIC SCATTERING FROM
TWO DIMENSIONAL OBJECTS USING THE
FIELD FEEDBACK FORMULATION

by

Thaddeus B. Welch III

March 1989

Thesis Advisor: Michael A. Morgan

Approved for public release; distribution is unlimited

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 is unlimited.
2b Declassification Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(if applicable)* 62	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization — 8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element No — Project No — Task No — Work Unit Accession No

11 Title *(include security classification)* ELECTROMAGNETIC SCATTERING FROM TWO DIMENSIONAL OBJECTS USING THE FIELD FEEDBACK FORMULATION

12 Personal Author(s) Thaddeus B. Welch III

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Engineer's Thesis	From To	March 1989	150

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

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	electromagnetics,scattering,numerical methods,FFF

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

Integral equations (IE's) are widely utilized to calculate induced currents on antennas and scatterers, but they are seriously restricted in their ability to handle inhomogeneous penetrable structures having multiwavelength dimensions. The utilization of finite element (FE) techniques has not been as pervasive as the use of IE's. The IE representation matrix is "full", containing few, if any, zero valued elements. The techniques for operating on these large-sized full matrices require undesirable amounts of processor time. FE techniques produce sparse matrices due to the strictly <u>local</u> interactions between discrete unknowns. The application of FE's to unbounded problems, however, requires supplementary enforcement of the far-field radiation conditions. The Field Feedback Formulation (F^3) circumvents the full-matrix computational "bottleneck" by allowing FE based numerical methods to be employed. Even though the resultant sparse matrices may be larger than the "full" matrices discussed earlier, most elements have a value of zero. Numerical procedures exist to optimize operations with these sparse matrices. Calculational speeds can be orders of magnitude faster. Computer techniques to implement and validate this new technique are the basis for this thesis. Excellent agreement with classical results are demonstrated.

FFF

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Michael A. Morgan	22b Telephone *(include Area code)* (408) 646-2677	22c Office Symbol 62Mw

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

Unclassified

Approved for public release; distribution is unlimited.

ELECTROMAGNETIC SCATTERING FROM TWO DIMENSIONAL OBJECTS USING THE FIELD FEEDBACK FORMULATION

by

Thaddeus B. Welch III
Lieutenant, United States Navy
B.E.E., The Georgia Institute of Technology, 1979

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

Author: ______________________________
Thaddeus B. Welch III

Approved by: ______________________________
Michael A. Morgan, Thesis Advisor

____
Ramakrishna Janaswamy, Second Reader

____
John P. Powers, Chairman,
Department of Electrical and Computer Engineering

____
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

Integral equations (IE's) are widely utilized to calculate induced currents on antennas and scatterers, but they are seriously restricted in their ability to handle inhomogeneous penetrable structures having multiwavelength dimensions. The utilization of finite element (FE) techniques has not been as pervasive as the use of IE's. The IE representation matrix is "full", containing few, if any, zero valued elements. The techniques for operating on these large-sized full matrices require undesirable amounts of processor time. FE techniques produce sparse matrices due to the strictly local interactions between discrete unknowns. The application of FE's to unbounded problems, however, requires supplementary enforcement of the far-field radiation conditions. The Field Feedback Formulation (F^3) circumvents the full-matrix computational "bottleneck" by allowing FE based numerical methods to be employed. Even though the resultant sparse matrices may be larger than the "full" matrices discussed earlier, most elements have a value of zero. Numerical procedures exist to optimize operations with these sparse matrices. Calculational speeds can be orders of magnitude faster. Computer techniques to implement and validate this new technique are the basis for this thesis. Excellent agreement with classical results are demonstrated.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. HISTORY

The application of finite element techniques to evaluate the solution of differential equations is well documented. The utilization of these techniques by the electromagnetics community has not been as pervasive as the use of integral equations (IE's). IE's are widely utilized to calculate induced currents on antennas and scatterers, but they are seriously restricted in their ability to handle inhomogeneous penetrable structures having multiwavelength dimensions. As the complexity and number of nodal degrees of freedom grow, the size and dimension of the representative matrix must also grow. This matrix is "full", containing few, if any, zero-valued elements. The available numerical techniques for operating on these large-sized full matrices require undesirable amounts of processor time.

Differential equation (DE) based techniques, such as the finite element method, produce sparse matrices due to the strictly <u>local</u> interactions between discrete unknowns which result. The application of DE's to unbounded problems, such as those of scattering and radiation, require some form of supplementary enforcement of the proper far-field conditions. These radiation boundary conditions are innately incorporated into integral equations.

B. FIELD FEEDBACK FORMULATION

The Field Feedback Formulation (F^3) circumvents the full-matrix "bottleneck" in the computational process by allowing DE based numerical methods to be employed. Even though the resultant sparse matrices may be larger than the "full" matrices discussed earlier, most elements have a value of zero. Numerical procedures exist to optimize operations with these sparse matrices. Calculational speeds can be orders of magnitude faster than with full matrices [Ref. 1]. Although the F^3 employs sparse matrices to represent the fields in the materials being considered, it does require augmentation to enforce the radiation condition at infinity on the scattered fields. This comes in the form of a feedback matrix composed of surface integration generated elements. A concept evaluation, for a special axisymmetric case, was already accomplished, as detailed in [Ref. 2] and [Ref. 3]. Computer techniques to implement and validate this new technique are the basis for this thesis.

C. POTENTIAL BENEFITS

This thesis will lead to an increased understanding of the advantages and disadvantages of this novel computational procedure for handling geometrically complex material scatterers. This method may ultimately allow computer-aided design of important electromagnetic structures such as low-observable aircraft, high efficiency dielectric lens antennas, and other electromagnetic scattering occurrences due to atmospheric anomalies. Structural details and material inhomogeneities, as well as physical dimensions (in multiple wavelengths), can be accommodated using the Field Feedback Formulation. These capabilities far surpass those which are possible with contemporary integral equation techniques for the case of inhomogeneous penetrable scatterers and antennas.

II. FORMULATION

A. INITIAL NOMENCLATURE

Assume there is a three dimensional object that is infinite in one direction. Such an object, in cross section could look like Figure 1. This object only varies in two dimensions, and therefore, is actually a two dimensional (2-D) object.

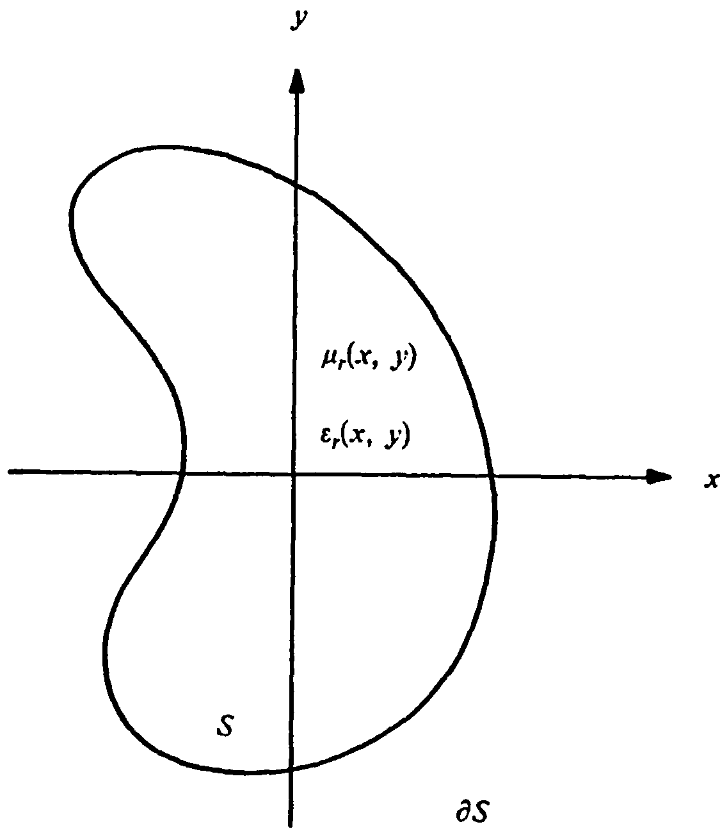

Figure 1. A Typical Object

The wavenumber k_0 is defined as

$$k_0 = \frac{2\pi}{\lambda_0} = \frac{2\pi f_0}{c},$$

where λ_0 is the free space wavelength associated with an electromagnetic wave of frequency f_0 and c is the speed of light. The X and Y Cartesian coordinates are wavenumber normalized, such that $X = k_0x$ and $Y = k_0y$. This coordinate normalization will be used throughout this development. A similar technique could also be developed in the polar coordinate system. The magnetic field will also be normalized such that $\overline{H} = -j\eta_0\overline{\mathscr{H}}$, where $\eta_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}} = 120\pi \approx 377\Omega$, is the impedance of free space and $\overline{\mathscr{H}}$ is the usual magnetic field in units of A/m. Thus the normalized $\overline{H}$ has the same V/m units as $\overline{E}$. Potentials may be defined as,

$$E_z(x,y) = \psi_1(X,Y) \tag{2.1}$$

for the transverse magnetic (TM) case, with E_x, E_y, and $H_z = 0$, and

$$H_z(x,y) = \psi_2(X,Y) \tag{2.2}$$

for the transverse electric (TE) case, with H_x, H_y and $E_z = 0$.

Our objective is to calculate the scattered fields for an arbitrary (2-D) penetrable object using the Field Feedback Formulation (F^3). As shown in Figure 2, a familiar closed loop system illustrates the relationship between the incident, scattered, total and far fields.

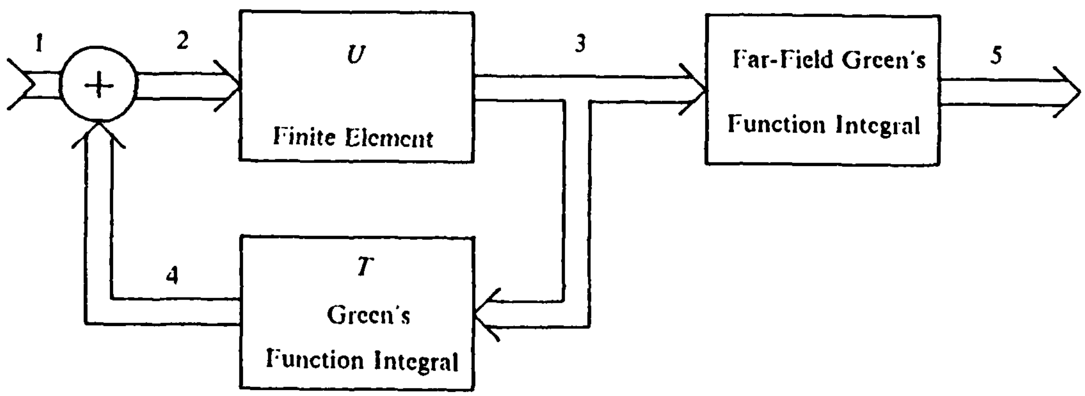

Figure 2. Field Feedback Formulation

At point 1, the incident field drives the total system. The incident field at 1, combined with the scattered field at 4, forms the total field at 2. This total field forms the boundary conditions that drive the finite element program at 2. At point 2, initially, the incident field drives the U operator. U represents the feed forward operator in the F^3. This operator uses a finite element technique to solve the boundary value problem. At point 3, the boundary conditions are solved for the object perimeter potentials and the normal derivative of these potentials. T represents the field feedback operator that takes the perimeter potentials and associated derivatives and provides the scattered fields at point 4, the offset boundary. The fields at point 1 and 4 are added (unlike the familiar feedback or control system where negative feedback is employed). At point 2, a combined incident and scattered field exists. These fields are the combined boundary conditions for the finite element boundary value program. These combined fields (total fields) are then applied to the U operator to calculate the perimeter fields and the associated derivatives on the objects perimeter. This looping may be repeated until a steady state condition, at point 3, is reached. The existence of a steady state condition assumes stability. Stability for physical systems should not be a problem. However, when mathematically modeled, instabilities may result. The error magnification or condition number of the system must also be seriously considered if this iterative looping process is to be used. The alternative approach is to form an equivalent system, where:

$$\text{equivalent operator} = U \bullet [I - T \bullet U]^{-1},$$

with

$$I = \text{Identity Matrix}$$

and

$$T \bullet U = \text{Combined Effects of the T and U operators} .$$

Either approach is viable since,

$$\psi_{total} = \psi_{Incident} + T \bullet U \bullet \psi_{Incident} + (T \bullet U)^2 \bullet \psi_{Incident} + \ . \ . \ . \ = [I - T \bullet U]^{-1} \psi_{Incident}$$

This becomes more obvious when $\psi_{Incident}$ is factored from the equality leaving,

$$1 + T \bullet U + (T \bullet U)^2 + \ . \ . \ .$$

Placing this in closed form,

$$\sum_{n=0}^{\infty} (T \bullet U)^n = \frac{1}{I - T \bullet U} = (I - T \bullet U)^{-1}.$$

Finding the equivalent operator will require a matrix inversion and for very large problems this may lead to excessive computation times. The matrix inversion technique will be investigated. The fields may then be extended to any point in space using a far field Green's function surface contour integral. This far field pattern is available at point 5.

The incident field is usually produced by a plane wave generator. This provides the boundary conditions on the offset contour. This contour is called "offset" since it is approximately the same "shape" as the objects perimeter but is slightly larger. The distance between the perimeter and this contour is called the offset distance and will be discussed in Chapter III. The boundary conditions may be any desired field or wave that satisfies Maxwell's equations. These waves may arrive from any direction and be of any magnitude. The boundary conditions may also be a composite of any number of waves since superposition does apply to these systems. A user provided subroutine is necessary if conditions other than a single plane wave, cylindrical mode (of arbitrary mode number) or individual input boundary condition is desired.

B. MAXWELL'S EQUATIONS

Maxwell's equations can be written using our previous normalizations as,

$$\nabla \times \overline{E} = \mu_r \overline{H} \tag{2.3}$$

and

$$\nabla \times \overline{H} = \varepsilon_r \overline{E}. \tag{2.4}$$

[Ref. 4] Let $D_X = \frac{\partial}{\partial x}$, with similar definitions for D_Y and D_Z. Equations 2.3 and 2.4 can be further expanded into the D_X, D_Y and D_Z components such that,

$$\mu_r H_x = D_Y E_z \tag{2.5}$$

$$\mu_r H_y = -D_X E_z \tag{2.6}$$

$$\mu_r H_z = D_X E_y - D_Y E_x \tag{2.7}$$

$$\varepsilon_r E_x = D_Y H_z \tag{2.8}$$

$$\varepsilon_r E_y = -D_X H_z \tag{2.9}$$

$$\varepsilon_r E_z = D_X H_y - D_Y H_x. \tag{2.10}$$

For the TM case, with propagation in the z direction,

$$H_x = \frac{D_Y E_z}{\mu_r}$$

and

$$H_y = \frac{-D_X E_z}{\mu_r}.$$

Note that the H_z field = 0. These two equations can be combined to form,

$$\overline{H} = \frac{1}{\mu_r} \nabla \psi_1 \times \hat{z}. \tag{2.11}$$

Similarly for the TE case, with propagation in the z direction,

$$\overline{E} = \frac{1}{\varepsilon_r} \nabla \psi_2 \times \hat{z}. \tag{2.12}$$

Substituting equations 2.5 and 2.6 into equation 2.10 yields,

$$\varepsilon_r E_z = D_X \left(\frac{-D_X E_z}{\mu_r} \right). \tag{2.13}$$

Substituting equation 2.1 into equation 2.13 yields,

$$\varepsilon_r \psi_1 + D_X \left(\frac{D_X}{\mu_r} \psi_1 \right) + D_Y \left(\frac{D_Y}{\mu_r} \psi_1 \right) = 0. \tag{2.14}$$

Equation 2.14 can be further simplified to,

$$\nabla \bullet \left[\frac{1}{\mu_r} \nabla \psi_1 \right] + \varepsilon_r \psi_1 = 0. \tag{2.15}$$

Similarly, equations 2.2, 2.8, 2.9 may be substituted into equation 2.7. This yields,

$$\nabla \bullet \left[\frac{1}{\varepsilon_r} \nabla \psi_2 \right] + \mu_r \psi_2 = 0. \tag{2.16}$$

Equations 2.15 and 2.16 are TM and TE duals. These two differential equations describe the potentials inside the object of interest. Defining $\frac{1}{\mu_r} = \alpha$ and $\varepsilon_r = \beta$ for the TM case and $\frac{1}{\varepsilon_r} = \alpha$ and $\mu_r = \beta$ for the TE case and substituting these new definitions into equations 2.15 and 2.16 yields one differential equation,

$$\nabla \bullet [\alpha \nabla \psi] + \beta \psi = 0. \tag{2.17}$$

C. VARIATIONAL EQUIVALENCE TO THE DIFFERENTIAL EQUATION

The Euler-Lagrange variational formulation is based on the stationarity of a functional, of the function ψ and its first derivatives. [Ref. 4]

$$I = \iint_{\text{inside S}} F(X, Y, \psi, \nabla \psi)\, dX\, dY \tag{2.18}$$

It can be shown that the first variation of the functional is zero, $\delta I = 0$, if the Lagrangian, F, satisfies the Euler - Lagrange equation:

$$\frac{\partial}{\partial X}\left(\frac{\partial F}{\partial (D_X \psi)}\right) + \frac{\partial}{\partial Y}\left(\frac{\partial F}{\partial (D_Y \psi)}\right) - \frac{\partial F}{\partial \psi} = 0. \tag{2.19}$$

The problem thus becomes to find the F, which when substituted into equation 2.19, yields the original differential equation. The Lagrangian,

$$F = \alpha\left\{(D_X \psi)^2 + (D_Y \psi)^2\right\} - \beta \psi^2$$

can be simplified to,

$$F = \alpha \nabla \psi \bullet \nabla \psi - \beta \psi^2. \tag{2.20}$$

When equation 2.20 is substituted into equation 2.19,

$$\frac{\partial F}{\partial (D_X \psi)} = 2\alpha D_X \psi$$

$$\frac{\partial F}{\partial (D_Y \psi)} = 2\alpha D_Y \psi$$

$$\frac{\partial F}{\partial \psi} = 2\beta \psi.$$

Therefore,

$$\frac{\partial}{\partial X}(2\alpha D_X\psi) + \frac{\partial}{\partial Y}(2\alpha D_Y\psi) + 2\beta\psi = 0$$

or,

$$D_X[\alpha D_X\psi] + D_Y[\alpha D_Y\psi] + \beta\psi = 0$$

which simplifies to,

$$\nabla \cdot [\alpha\nabla\psi] + \beta\psi = 0. \tag{2.21}$$

Therefore, the general functional has been found since equation 2.21 and equation 2.17 are identical. Substituting the α and β definitions into equation 2.20 yields,

$$F_1 = \frac{1}{\mu_r}\nabla\psi_1 \cdot \nabla\psi_1 - \varepsilon_r\psi_1^2 \tag{2.22}$$

and

$$F_2 = \frac{1}{\varepsilon_r}\nabla\psi_2 \cdot \nabla\psi_2 - \mu_r\psi_2^2. \tag{2.23}$$

Equations 2.22 and 2.23 are integrated over the interior to S with known boundary conditions for either ψ_1 or ψ_2 on S. To physically interpret the variational formulation, it is noted that,

$$\nabla\psi_1 = \mu_r\left\{H_X\hat{Y} - H_Y\hat{X}\right\}$$

and

$$\nabla\psi_2 = \varepsilon_r\left\{E_X\hat{Y} - E_Y\hat{X}\right\}.$$

Therefore,

$$I_1 = \iint_S \mu_r\overline{H} \cdot \overline{H} - \varepsilon_r\overline{E} \cdot \overline{E}\; dX\, dY$$

and

$$I_2 = \int\int_S \varepsilon_r \overline{E} \bullet \overline{E} - \mu_r \overline{H} \bullet \overline{H} \, dX \, dY.$$

Substituting Maxwell's equations into I_1 gives,

$$I_1 = \int\int_S (\nabla \times \overline{E}) \bullet \overline{H} - (\nabla \times \overline{H}) \bullet \overline{E} \, dX \, dY$$

$$I_1 = \int\int_S \nabla \bullet (\overline{E} \times \overline{H}) \, dX \, dY$$

$$I_1 = \int_{\partial S} \overline{E} \times \overline{H} \bullet \hat{n} \, dl.$$

Note that $\overline{E} \times \overline{H}$ is the complex oscillatory Poynting vector. It is different from the usual Poynting vector of $\overline{E} \times \overline{H}^*$. Thus, both functionals, I_1 and I_2 are proportional to the complex phasors for oscillatory power. The oscillatory power is the phasor representing the excess of instantaneous radiated power minus the average radiated power.

D. FINITE ELEMENT BOUNDARY VALUE SOLUTION

With the discussion of the variational mathematics completed, a simple rectangular boundary value problem will be discussed in detail. The rectangular geometry allows for an easier formulation but in no way limits the solution from being extended to more complicated object geometries. Figure 3 on page 11 shows the region of concern.

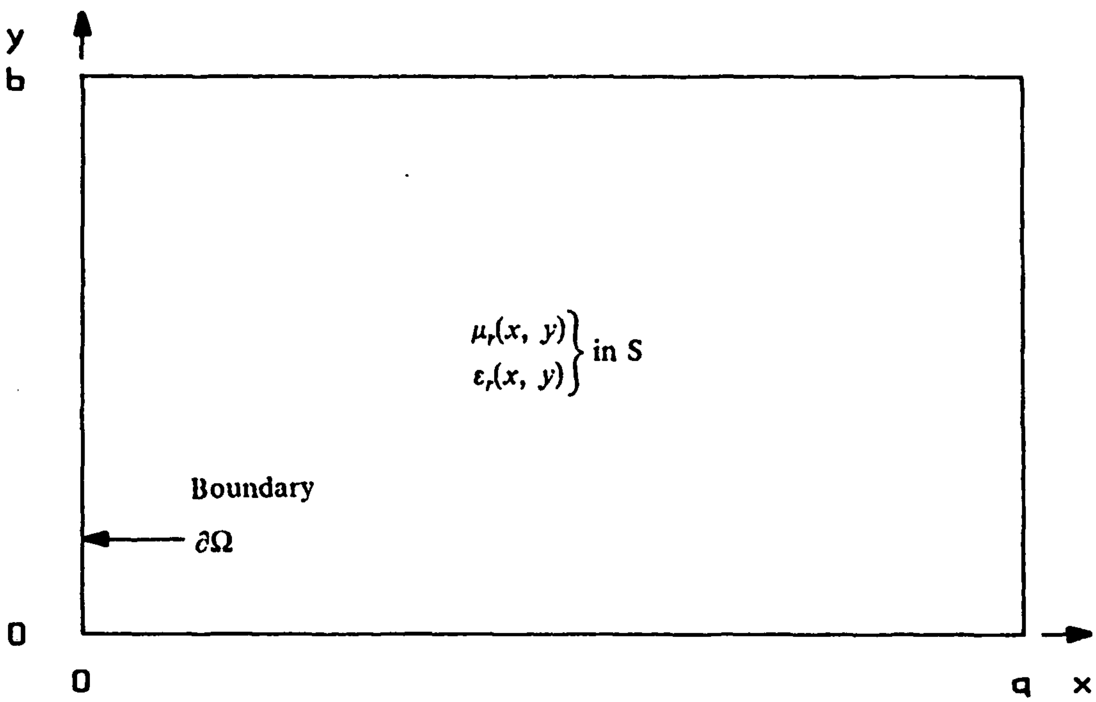

Figure 3. Rectangular Object

Consider spanning the rectangular region by a triangular mesh, as shown in Figure 4 on page 12. Both α and β may be functions of position but may not vary within an individual triangular element. Given a fine enough mesh structure, a smooth transition in material properties may be approximated. For notational simplicity this positional dependance will not be carried forward. The variational approach will yield the solution to the boundary value problem by finding the $\psi(x,y)$ which gives the stationary value of,

$$I = \int_0^a \int_0^b \left(\alpha\nabla\psi \cdot \nabla\psi - \beta\psi^2\right) dx\, dy$$

which is constrained on the boundary by the previously specified boundary conditions. [Ref. 5]

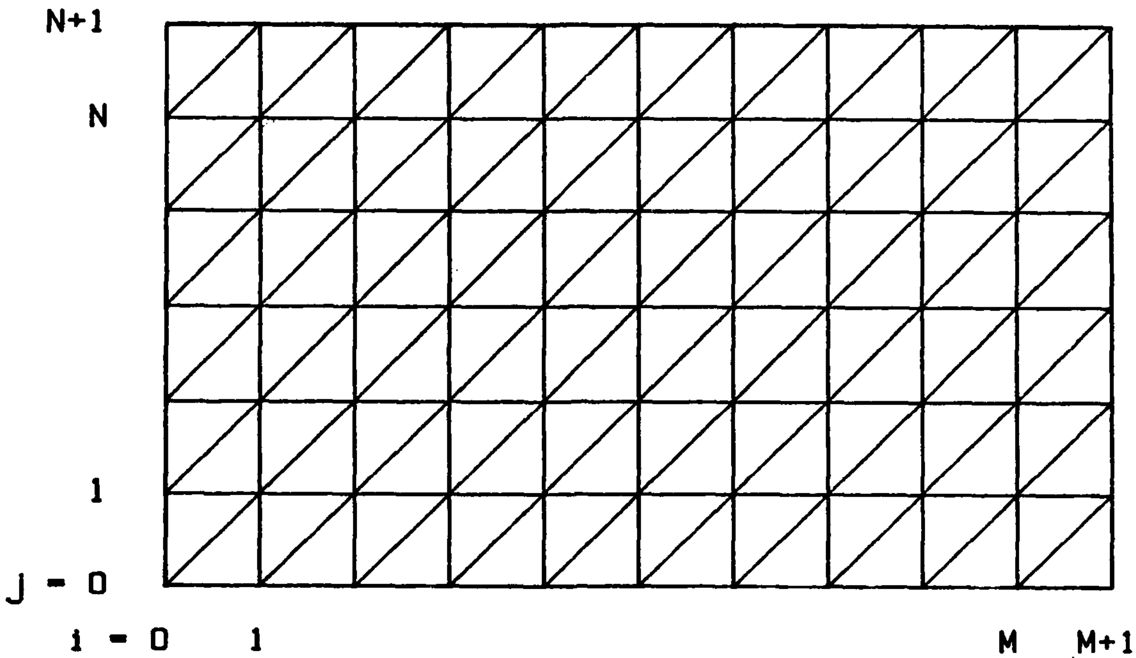

Figure 4. Rectangular Mesh

The values of ψ at the interior nodes become the discretized unknowns:

$$\psi_{ij} = \psi(X_i, Y_j) \quad i = 1 \ldots M \text{ and } j = 1 \ldots N$$

where

$$X_i = i \bullet \Delta X$$

and

$$Y_j = j \bullet \Delta Y$$

for the uniform mesh structure with $\Delta X = \frac{a}{(M+1)}$ and $\Delta Y = \frac{b}{(N+1)}$. Approximating,

$$\psi(x, y) = \sum_{i=0}^{M+1} \sum_{j=0}^{N+1} \psi_{ij}\, u_{ij}(x, y)$$

which includes the known boundary nodal values $\psi_{0,j}$ and $\psi_{M+1,j}$ for j = 0 and N + 1, and $\psi_{i,0}$ and $\psi_{i,N+1}$ for i = 0 and M + 1, linear pyramidal basis functions, $u_{ij}(x,y)$, which have unit value at the (i,j) node and zero value at all surrounding nodes. Figure 5(a) is a top view of the pyramidal basis function, while Figure 5(b) is a perspective view.

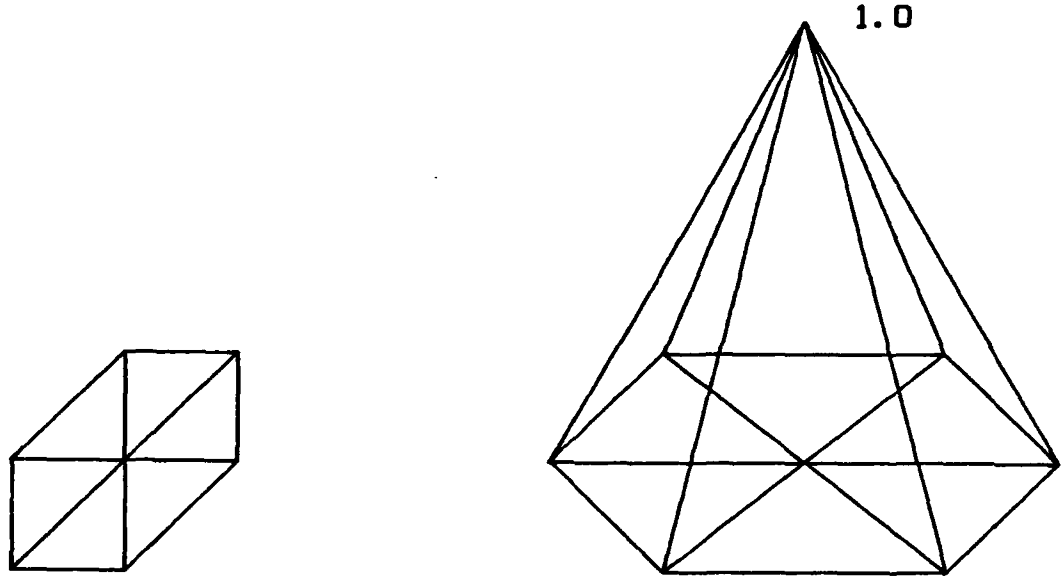

Figure 5. Pyramidal Basis Function, (a) top view, (b) perspective view

The functional I will thus be a discrete function of each of the nodal values of ψ,

$$I = I(\psi_{0,0}, \psi_{0,1}, \ldots, \psi_{i,j}, \ldots, \psi_{M+1,N+1}).$$

The approximate discrete solution will be found by the system,

$$\frac{\partial I}{\partial \psi_{m,n}} = 0, \quad \text{for} \quad m = 1 \ldots M \text{ and } n = 1 \ldots N.$$

Now,

$$\frac{\partial I}{\partial \psi_{m,n}} = 2\int_0^a \int_0^b \left(\alpha \frac{\partial \nabla \psi}{\partial \psi_{m,n}} \bullet \nabla \psi - \beta \frac{\partial \psi}{\partial \psi_{m,n}} \psi \right) dx\, dy = 0$$

where,

$$\nabla\psi(x,y) = \sum_{i=0}^{M+1}\sum_{j=0}^{N+1}\psi_{ij}\nabla u_{ij}(x,y).$$

The gradient of the basis function is,

$$\nabla u_{m,n}(x,y) = \frac{\partial\nabla\psi}{\partial\psi_{m,n}}$$

and the basis function is,

$$u_{m,n}(x,y) = \frac{\partial\psi}{\partial\psi_{m,n}}.$$

Therefore, the system of equations to solve becomes,

$$\int_0^a\int_0^b(\alpha\nabla u_{m,n}\bullet\nabla\psi - \beta u_{m,n}\psi)\,dx\,dy = 0,\ \text{for } m = 1\ldots M \text{ and } n = 1\ldots N.$$

Substituting for $\nabla\psi$ and ψ in terms of the nodal values of ψ for $m = 1\ldots M$ and $n = 1\ldots N$ gives,

$$\sum_{i=1}^{M+1}\sum_{j=0}^{N+1}\psi_{ij}\int_0^a\int_0^b(\alpha\nabla u_{m,n}\bullet\nabla u_{i,j} - \beta u_{m,n}u_{i,j})\,dx\,dy = 0$$

where,

$$\int_0^a\int_0^b(\alpha\nabla u_{m,n}\bullet\nabla u_{i,j} - \beta u_{m,n}u_{i,j})\,dx\,dy = F\{(m,n),(i,j)\}.$$

Regrouping this to put the known nodal values on the right hand side gives for $m = 1\ldots M$ and $n = 1\ldots N$ (interior nodes),

$$\sum_{i=1}^{M}\sum_{j=1}^{N}\psi_{ij}F[(m,n),(i,j)] = -\sum_{\substack{\text{Boundary}\\ \text{Nodes Only}\\ \text{for } (i,j)}}\sum\psi_{ij}F[(m,n),(i,j)].$$

By renumbering the nodes using a single index for the unknown nodal values, $k = N(i-1)+j$ and $l = N(m-1)+n$ and,

$$\sum_{k=1}^{M \bullet N} F(l,k)\psi_k = -\sum_{k'} F(l,k')\psi_{k'}.$$

The functional $F(l,k) = 0$ if nodes l and k are not both associated with at least one common triangular element. Therefore, $\nabla u_l \bullet \nabla u_k$ and $u_l u_k$ will be zero except in triangles where l and k both appear as nodes. For the example mesh structure of Figure 3 on page 11, this produces a banded matrix, F. This sparse matrix can be easily displayed by first defining column vectors of the unknown nodal values in the mesh

$$\overline{\Psi}_l = [\psi_{l,1}, \psi_{l,2}, \ldots, \psi_{l,N}]^T.$$

The F - matrix elements $F\,[(m,n),(i,j)]$ are zero unless the (m,n) node shares at least one element with the (i,j) node. Thus, the node values in $\overline{\psi}_i$ will be coupled only to $\overline{\psi}_{i-1}$, $\overline{\psi}_{i-1}$ and any associated boundary nodes. This is written as,

$$[A_l]\overline{\Psi}_{l-1} + [B_l]\overline{\Psi}_l + [C_l]\overline{\Psi}_{l+1} = -[P_l]\overline{\Psi}_{B_l}$$

where A_i, B_i and C_i are $N \times N$ complex arrays and P_i is a $N \times N_{B_i}$ array where N_{B_i} is the number of boundary nodes associated with the i-th column. This appears as,

$$\begin{bmatrix} B_1 & C_1 & & & & \\ A_2 & B_2 & C_2 & & 0 & \\ & A_3 & B_3 & C_3 & & \\ & & & & & \\ & 0 & & A_{M-1} & B_{M-1} & C_{M-1} \\ & & & & A_M & B_M \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \bullet \\ \bullet \\ \bullet \\ \Psi_{M-1} \\ \Psi_M \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \bullet \\ \bullet \\ \bullet \\ P_{M-1} \\ P_M \end{bmatrix} \begin{bmatrix} \Psi_{B_1} \\ \Psi_{B_2} \\ \Psi_{B_3} \\ \bullet \\ \bullet \\ \bullet \\ \Psi_{B_{M-1}} \\ \Psi_{B_M} \end{bmatrix}$$

If we denote $\overline{\psi}_0$ as the initial boundary values and $\overline{\psi}_{M+1}$ as the final boundary values then $\overline{\psi}_{B_i}$ becomes only the boundary conditions on the top and bottom at $j \neq 0$ and

$N+1$. Note that the above system is tri-block in nature and has a large number of zero valued elements. The zero valued elements were omitted for clarity. Each element is actually a matrix and therefore it is evident how sparse this system is. Each of the A_i, B_i, C_i and P_i matrices are equally sparse, however, a global symmetry does not appear.

E. EVALUATION OF THE F - MATRIX CONTRIBUTIONS

Given an arbitrary element as shown in Figure 6, the potential, ψ, can be linearly approximated by [Ref. 5],

$$\psi(x,y) = (x,y,1) \cdot [T] \cdot \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix}$$

$$= \sum_{k=1}^{3} \psi_k u_k(x,y),$$

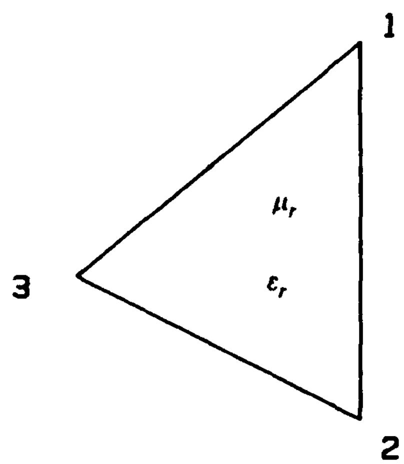

Figure 6. Mesh Element

where $\psi_k = \psi(x_k, y_k)$ is the nodal value at the k-th node, $[T] = 3 \times 3$ Transform Array and

$$u_k(x, y) = \text{Linear basis functions for the k-th node}$$

$$= (x, y, 1)\begin{bmatrix} T_{1,k} \\ T_{2,k} \\ T_{3,k} \end{bmatrix}$$

$$= T_{1,k}x + T_{2,k}y + T_{3,k}.$$

Note that,

$$u_k(x_m, y_m) = \begin{Bmatrix} 1 & , & for\ k = m \\ 0 & , & for\ k \neq m \end{Bmatrix}.$$

It can be shown that,

$$[T] = \frac{1}{2A}\begin{bmatrix} (y_2 - y_3) & (y_3 - y_1) & (y_1 - y_2) \\ (x_3 - x_2) & (x_1 - x_3) & (x_2 - x_1) \\ (x_2y_3 - x_3y_2) & (x_3y_1 - x_1y_3) & (x_1y_2 - x_2y_1) \end{bmatrix}.$$

where $|A|$ = triangle area, and

$$2A = \det\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

$$2A = (x_2y_3 + x_3y_1 + x_1y_2) - (x_3y_2 + x_1y_3 + x_2y_1)$$

Furthermore, the linear basis functions, $u_k(x, y)$, can be interpreted as relative areas of the triangle shown in Figure 7 on page 18.

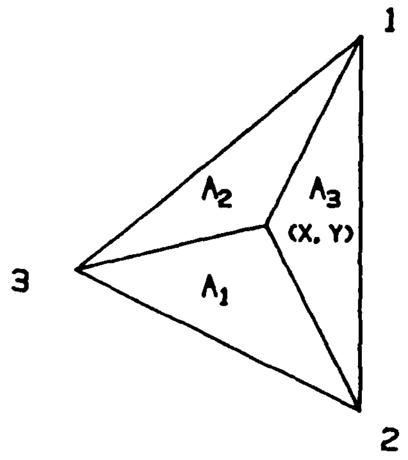

Figure 7. Basis Functions Interpretation

$$A = A_1 + A_2 + A_3$$

is constant as (x, y) varies and,

$$u_k(x,y) = \frac{A_k(x,y)}{A}$$

Within a given triangular element, the evaluation for k = 1, 2, 3 and l = 1, 2, 3 in the element is of interest and,

$$\underset{\substack{\text{inside}\\\text{triangle}\\q}}{\int\int} (\alpha_q \nabla u_l \cdot \nabla u_k - \beta_q u_l u_k)\, dx\, dy.$$

The assumption is made that α and β will be approximated as constants within the triangle. These material constants can, however, vary from element to element. Taking the gradient terms first,

$$\nabla u_k = T_{1,k}\hat{x} + T_{2,k}\hat{y}$$

$$\nabla u_l \cdot \nabla u_k = (T_{1,k} \cdot T_{1,l} + T_{2,k} \cdot T_{2,l}).$$

Therefore,

$$\iint\limits_{\text{triangle}} \alpha_q \nabla u_l \bullet \nabla u_k dxdy$$

$$= \alpha_q (T_{1,k} T_{1,l} + T_{2,k} T_{2,l}) \,|\, A_q \,|,$$

where A_q is the area of the q^{th} element. Next it can be derived that,

$$\iint\limits_{\text{triangle}} u_l u_k dxdy = \begin{Bmatrix} \frac{1}{12} \,|\, A \,| & , \; for\; k \neq l \\ \frac{1}{6} \,|\, A \,| & , \; for\; k = l \end{Bmatrix}.$$

More generally, with each u_i raised to an integer power, n_i,

$$\iint\limits_{\text{triangle}} u_1^{n_1} u_2^{n_2} u_3^{n_3} dxdy = 2 \,|\, A \,| \, \frac{n_1! n_2! n_3!}{(n_1 + n_2 + n_3 + 2)!}$$

The final result is,

$$F_q(l, k) = \iint\limits_{\substack{\text{triangle} \\ q}} (\alpha \nabla u_l \bullet \nabla u_k - \beta u_l u_k) dxdy$$

$$= |\, A \,| \left\{ \alpha (T_{1,k} T_{1,l} + T_{2,k} T_{2,l}) - \frac{1}{12} \beta \right\}, \quad k \neq l$$

$$= |\, A \,| \left\{ \alpha \left(T_{1,k}^2 + T_{2,k}^2 \right) - \frac{1}{6} \beta \right\}, \quad k = l.$$

F. GREEN'S FUNCTION CONTOUR INTEGRAL

The scattered fields, ψ, from an arbitrary object in a vacuum satisfying Helmholtz's equation (see Figure 8 on page 20),

$$\nabla^2 \psi + k^2 \psi = 0,$$

are [Ref. 6],

$$\psi_s(\bar{r}) = \int_{\text{contour}} \left\{ G(r\,|\,\bar{r}')\,\frac{\partial \psi}{\partial n} - \psi(\bar{r}')\,\frac{\partial G}{\partial n} \right\} dc'.$$

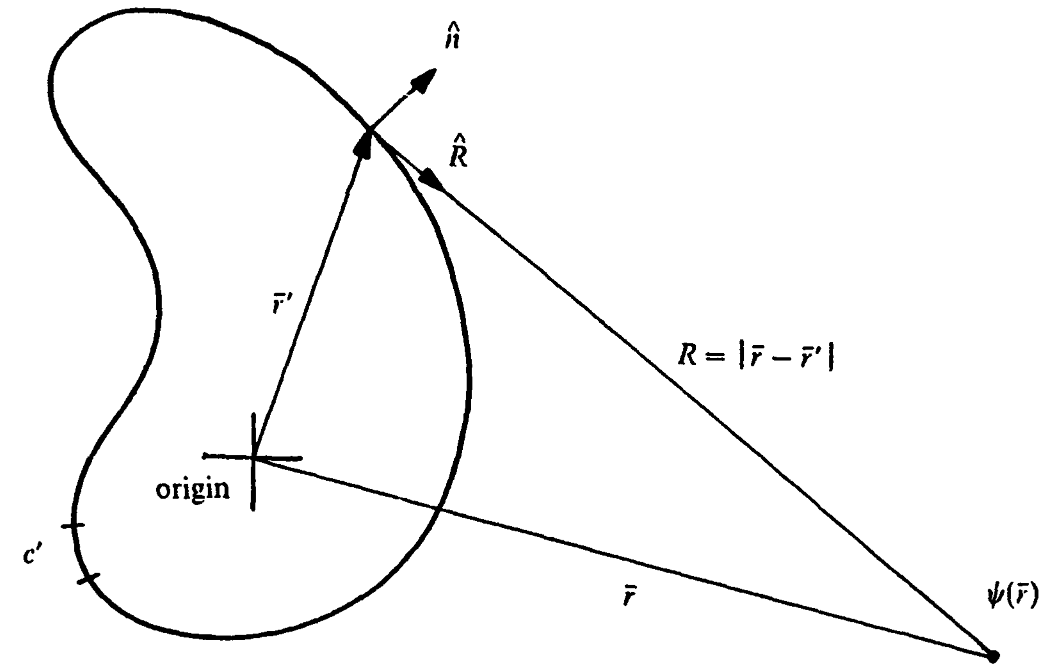

Figure 8. Green's Function Integration

where the Green's function is,

$$G(\bar{r}\,|\,\bar{r}') = \frac{j}{4}\,H_0^{(2)}\left(k_0\,|\,\bar{r} - \bar{r}'\,|\right)$$

and

$$\frac{\partial \psi}{\partial n} = \hat{n} \cdot \overline{\nabla \psi} \quad \text{on the contour}$$

and

$$\frac{\partial G}{\partial n} = \hat{n} \cdot \hat{r}\,\frac{jk_0}{4}\,H_1^{(2)}\left(k_0\,|\,\bar{r} - \bar{r}'\,|\right).$$

The Hankel functions of the second kind of order zero and one, $H_0^{(2)}$ and $H_1^{(2)}$, will present a problem for the numeric integration discussed in Chapter V. The imaginary portion of these functions rapidly approaches negative infinity as the argument approaches zero. The $\frac{\partial\psi}{\partial n}$ is obtained by a finite difference method using the field boundary conditions on surface B (boundary conditions) and the calculated field conditions on surface P (object perimeter). This results in,

$$\frac{\partial\psi}{\partial n} \approx \frac{\psi_{boundary} - \psi_{perimeter}}{\text{offset distance}}.$$

It will be shown that to maximize the accuracy of the Green's function the offset distance should be made as large as possible. This, however, causes the $\frac{\partial\psi}{\partial n}$ to be inaccurate. Thus, an optimal condition must be found that maximizes the accuracy of the entire numeric integration. Such a condition does not maximize the accuracy of any one of the contributing parts to the Green's function integrand.

G. FAR-FIELD EVALUATION

When the Green's function integral discussed above is used for far field calculations several simplifying relationships develop. These simplifications require a less demanding numerical integration. To be in the far field region three conditions must exist,

$$|\bar{r}| > D, \qquad |\bar{r}| > \lambda_0 \quad \text{and} \quad |\bar{r}| > \frac{2D^2}{\lambda_0},$$

where λ_0 is the free space wavelength and D is the maximum dimension of the object. [Ref. 7] As x approaches infinity,

$$H_0^{(2)}(x) \rightarrow \sqrt{\frac{2j}{\pi x}}\, e^{-jx}$$

and

$$H_1^{(2)}(x) \rightarrow \sqrt{\frac{2j}{\pi x}}\, je^{-jx}.$$

This requires,

$$G(\bar{r}\,|\,\bar{r}') = \frac{j}{4}\sqrt{\frac{2j}{\pi k_0 R}}\, e^{-jk_0 R}$$

and

$$\frac{\partial G}{\partial n} = -\hat{n} \bullet \hat{R}\frac{k_0}{4}\sqrt{\frac{2j}{\pi k_0 R}}\, e^{-jk_0 R}.$$

In the far field $\sqrt{\frac{1}{R}} \to \sqrt{\frac{1}{r}}$ and $\hat{n} \bullet \hat{R} \to \hat{n} \bullet \hat{r}$. Thus,

$$e^{-jk_0 R} \to e^{-jk_0 r} \bullet e^{jk_0 \hat{r} \bullet \bar{r}'}.$$

Therefore,

$$G(\bar{r}\,|\,\bar{r}') = \frac{j}{4}\sqrt{\frac{2j}{\pi k_0 r}}\, e^{-jk_0 r} e^{jk_0 r' \cos\theta}$$

and

$$\frac{\partial G}{\partial n}(\bar{r}\,|\,\bar{r}') = -\frac{k_0}{4}\sqrt{\frac{2j}{\pi k_0 r}}\, e^{-jk_0 r}(\hat{n} \bullet \hat{r}) e^{jk_0 r' \cos\theta}.$$

With these new definitions substituted into the original Green's function integral equation,

$$\psi_{scattered}(\bar{r}) = \sqrt{\frac{j}{8\pi k_0 r}}\, e^{-jk_0 r} \int_C \left[j\frac{\partial \psi}{\partial n} + k_0 \hat{n} \bullet \hat{r}\, \psi(\bar{r}') \right] e^{jk_0 r' \cos\theta} dc'.$$

Note that this equation is partitioned into a distance dependent term and a theta depend term. The theta dependent term may be defined as

$$I = \int_C \left[j\frac{\partial \psi}{\partial n} + k_0 \hat{n} \bullet \hat{r}\, \psi(\bar{r}') \right] e^{jk_0 r' \cos\theta} dc'.$$

The two dimensional bistatic radar cross section (RCS) per unit length of the cylindrical structure may now be defined as,

$$RCS = \sigma(\phi^s, \phi^i) = \lim_{r \to \infty} \frac{2\pi r P^s}{P^{inc}} = \lim_{r \to \infty} \frac{2\pi r\,|\,\psi^s\,|^2}{|\,\psi^i\,|^2}$$

$$\sigma = \lim_{r \to \infty} 2\pi r \frac{I^2}{8\pi h r} = \frac{|I|^2}{4k_0},$$

where the wavenumber, $k_0 = \frac{2\pi}{\lambda_0}$ and the incident field is assumed to be of unit magnitude, $|\psi^i| = 1.0$.

III. MESH GENERATION

A. INTRODUCTION

A display or plot of the computer generated mesh structure is not required for the problem solution, however, it provides an immediate visual confirmation that the intended problem geometry has been entered correctly into the computer. A large amount of initialization data is required for even the most rudimentary problem. For this reason, the input data is provided to the mesh generation program via a data file. Modifications are possible at a later time with minimum effort.

B. INPUT DATA

The input data file is called INPUT.DAT and contains 25 input fields. Of these fields, 14 relate directly to the generation or display of the mesh structure. Only the object surface coordinates require adherence to a specific format. All other data need only be of the correct type (i.e., character, integer or real). A brief description of each field is provided below.

- Field 1 is a label of no more than 12 characters. This label is for the plot and input data file.
- Field 2 is a character flag that specifies the input coordinate system. If set to "R" or "r", the rectangular system is used. If set to "P" or "p", the polar system is used. If a "P","p" ,"R" or "r" is not detected, an error is returned to the display.
- Field 3 is a character flag that if set to "I" or "i" will cause several intermediate values to be stored to disk during the Finite Element Boundary Value (FEBV) program execution. This option was used during the debug process.
- Field 4 is a character flag that if set to "D" or "d" will create a DISSPLA FORTRAN program capable of replicating the input object, in wavenumber normalized coordinates, on mainframe computers having a DISSPLA graphics package. DISSPLA is a subroutine-based language. The generated program, called DISSPLA.FOR, is a compilation of four subroutines calls per element. This file can get very large for dense mesh structures.
- Field 5 is a character flag that if set to "U" or "u" will cause a uniform material to be assumed. In the uniform case, no material interface exists. This option was only used to verify the Finite Element solution accuracy.
- Field 6 is a character flag that if set to "M" or "m" will cause only the mesh to be generated. This is very useful when first starting a problem and the optimal mesh structure has not been determined.
- Field 7 is a real number specifying the desired mesh resolution in wavelengths. The mesh resolution determines the dimension of the mesh elements.

- Field 8 is a real number specifying the distance, in wavelengths, between the object perimeter and the offset boundary contour.
- Field 9 is a real number that specifies a multiplicative scaling factor for the numerical integration stepping function discussed in Chapter V.
- Field 10 is a bias term that can be used to shift the numerical integration stepping function. This term is used for distances less than 1.0.
- Field 11 is a bias term that can be used to shift the numerical integration stepping function. This term is used for distances greater than 1.0.
- Field 12 is a real number specifying the maximum distance beyond which no further contribution to the Green's Function Integral is made. If this feature is not desired, this term should be made larger than the objects maximum dimension plus twice the offset distance.
- Field 13 is an integer specifying the number of input data points.
- Field 14 is an integer specifying the angular resolution, in degrees, desired for the final radar cross section calculation.
- Field 15 is an integer specifying the mesh generation technique.
- Field 16 is an integer specifying the perimeter node from which the bisection segment originates. This node is called the "start node".
- Field 17 is an integer specifying the perimeter node on which the bisection segment terminates. This node is called the "stop node".
- Field 18 is a pair of real numbers (on two lines) specifying the x and y coordinates by which the object will be displaced.
- Field 19 is a real number specifying, in wavelengths, the desired distance between the origin and the first input data point. Fields 18 and 19 when used together, allow an object to be placed at any position and scaled to any size.
- Field 20 is a pair of real numbers (on two lines) specifying the real and imaginary parts of $\frac{1}{\mu_r}$ for the TM case or $\frac{1}{\varepsilon_r}$ for the TE case.
- Field 21 is a pair of real numbers (on two lines) specifying the real and imaginary parts of ε_r for the TM case or μ_r for the TE case.
- Field 22 is a character flag that if set to "P" or "p" enables a plane wave generator. The plane wave is propagating down the y axis, and generates an $E_0 e^{jky}$ condition on the offset boundary contour. If the flag is set to "C" or "c", a cylindrical mode generator is enabled. This generates an $E_0 \cos n\phi$ condition on the offset boundary contour. If a "P" , "p", "C" or "c" is not detected, then manually input boundary conditions must follow, and fields 23 and 24 are not used.
- Field 23 is a real number specifying the wave amplitude.
- Field 24a is a real number specifying the wave frequency (in Hz).
- Field 24b (only for cylindrical case) is an integer specifying the mode number, n.
- Field 25 is the object perimeter data, in either polar or rectangular form.

An example of an input data file for a homogeneous circular cylinder is provided in Appendix A. The majority of the input data is echoed to the computer display and a system "pause" is initiated to allow for user inspection. The program may be aborted or continued at this time. The initial object dimensionalization has already occurred, and the number of unknowns and the maximum unknown width is displayed. These factors give an excellent indication of the expected run time for the FEBV routines. For example, a problem with 512 unknowns and a maximum unknown width of 31 took 806 seconds to execute while a run with 8 unknowns and a maximum unknown width of 3 took only 10 seconds to execute. These times are for a Intel 80386 based personal computer with an Intel 80287 co-processor chip calculating the fields for a circular cylinder. The FEBV routines are the next code block to execute after the pause is cleared.

C. MESH GENERATION PROGRAM

The mesh generation program consists of seven subroutines. These subroutines are an integral part of the finite element program and, therefore, were not separated. These routines are discussed below.

1. IO (Input/Output)

This subroutine reads the information contained in the INPUT.DAT file discussed earlier. A two dimensional object can be described in any number of ways, however, for simplicity, the polar and rectangular coordinate systems are used. In either case, the initial assumption is that all data points are referenced to a local origin. This local origin can be offset by any desired amount using field 18. This offset is independent of the entered data points or any size scaling provided by field 19. A plot label file, named TEXT.LBL, is created and the initial object perimeter (coded for a display in blue) is written to the output file, PLT.DAT. These data points describe the perimeter of the object. This will later prove helpful in determining the conformity of the generated mesh to the input perimeter. All subsequent screen writes are coded for a display in green. Two example objects are shown in Figure 9 on page 27. These objects will be used throughout this chapter. The circular cylinder was generated by a separate computer program. The "horseshoe" shaped object was manually input. Graph paper was used to determine the x and y coordinates of the 28 unequally spaced perimeter points.

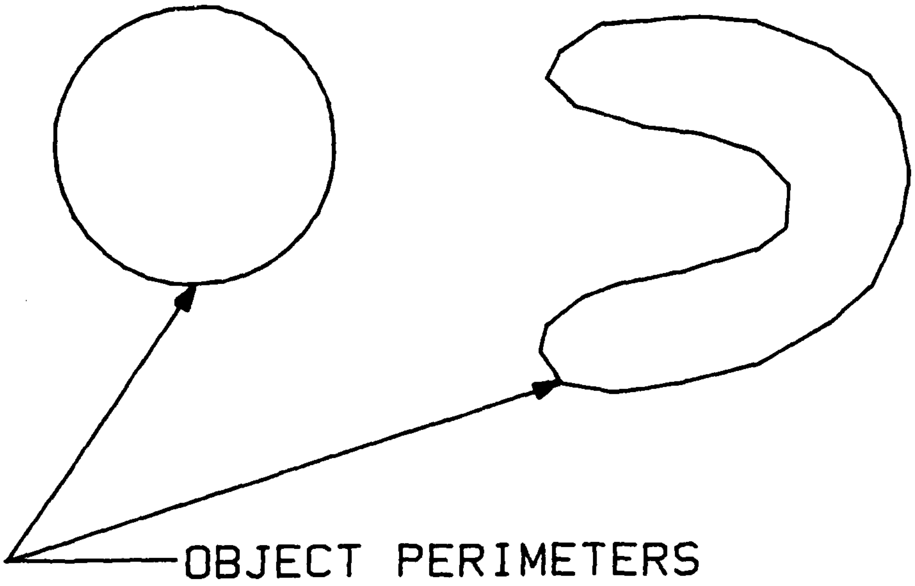

Figure 9. Typical Objects

2. Rotate

This subroutine reorders the input data points to allow for any desired bisection segment start and/or stop node. This subroutine is only used if manual selection of the bisection segment start and/or stop nodes is requested.

3. Bound (Boundary)

This subroutine sub-divides the object perimeter based on the mesh resolution specified in Field 7. The mesh resolution is the approximate length, specified in wavelengths, that the user desires the perimeter to be divided into. The division of the perimeter is based on linear interpolation between input data points. A new perimeter node is placed at each of the sub-division points. Additional input data points are necessary in areas of rapid change to allow for a correct object perimeter representation. The object bisection extends from the "start node" to the "stop node". These nodes are user specified in Fields 16 and 17 and, in general, divide the object in half. The bisection should be arranged such that the object width, perpendicular to the bisection segment, is minimized. Thus, a long slender object should be oriented for a major axis bisection. Whether the major axis is oriented vertically or horizontally is of no importance. This is demonstrated in Figure 10 on page 28 (right side).

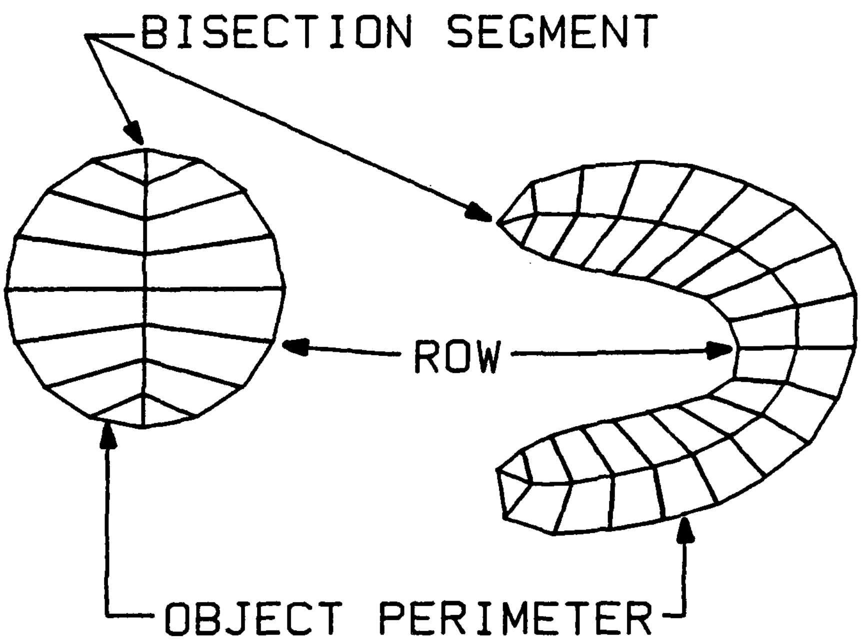

Figure 10. Bisection Segment and Row Arrangement

For more complex objects, the bisection is not a straight line, but rather a series of line segments that approximate a curve. The number of perimeter nodes on the left side of the bisection segment must equal the number of perimeter nodes on the right side of the bisection segment. The bisection segment also contains this same number of nodes. Thus, the program must adjust the requested mesh resolution to ensure proper nodal spacing. This allows for a piecewise continuous segment to cross the object.

4. Normal

This subroutine calculates the unit normals to the object perimeter at each newly established perimeter node. The normal is a perpendicular constructed to the chord connecting the two nodes adjacent to the node for which the calculation is occurring. This perpendicular originates at the current node and is of unit length. See Figure 11 on page 29.

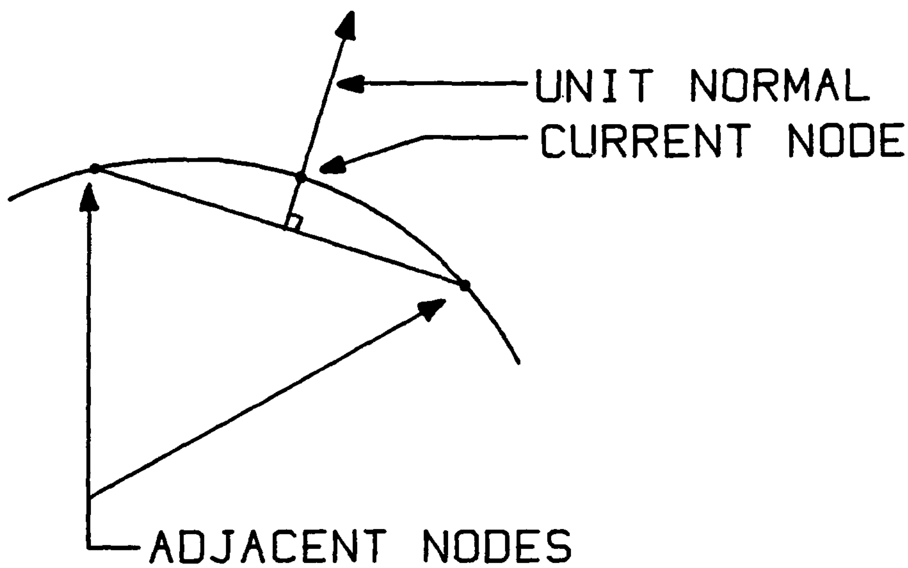

Figure 11. Unit Normal Calculation

5. Nodset (Node Set)

This subroutine uses a two sweep technique to compute the number of nodes on each nodal row. The rows are labeled I = 1, 2, 3, ..., I maximum (IMX) consecutively from the top of the object to the bottom. Segments normal to the perimeter surface, calculated in NORMAL, connect the perimeter nodes to the offset boundary contour. Two such completed normal segments, one on each end, complete a nodal row. When all rows are complete, a second contour has been established that approximates the object's perimeter. This contour is displaced from the perimeter by the distance specified in Field 8. This contour provides the boundary conditions for the FEBV routines. The optimal selection of this offset distance will be a topic of Chapter IV.

With the object divided into rows, each row can be further divided into equally spaced nodes. Based on the requested resolution and whether the objects dimensions are expanding or contracting, the nodal spacing is adjusted to keep the elements approximately the same size. Two adjacent nodal rows form an elemental row. These rows are given the same label, I = 1, 2, 3, ..., IMX as the upper nodal row. The nodes on an elemental row are numbered consecutively, starting with the left-most node. This node is always an offset boundary contour node. When the upper nodal row is numbered, the process is continued for the lower nodal row. An example of this element row numbering scheme is shown in Figure 12 on page 30.

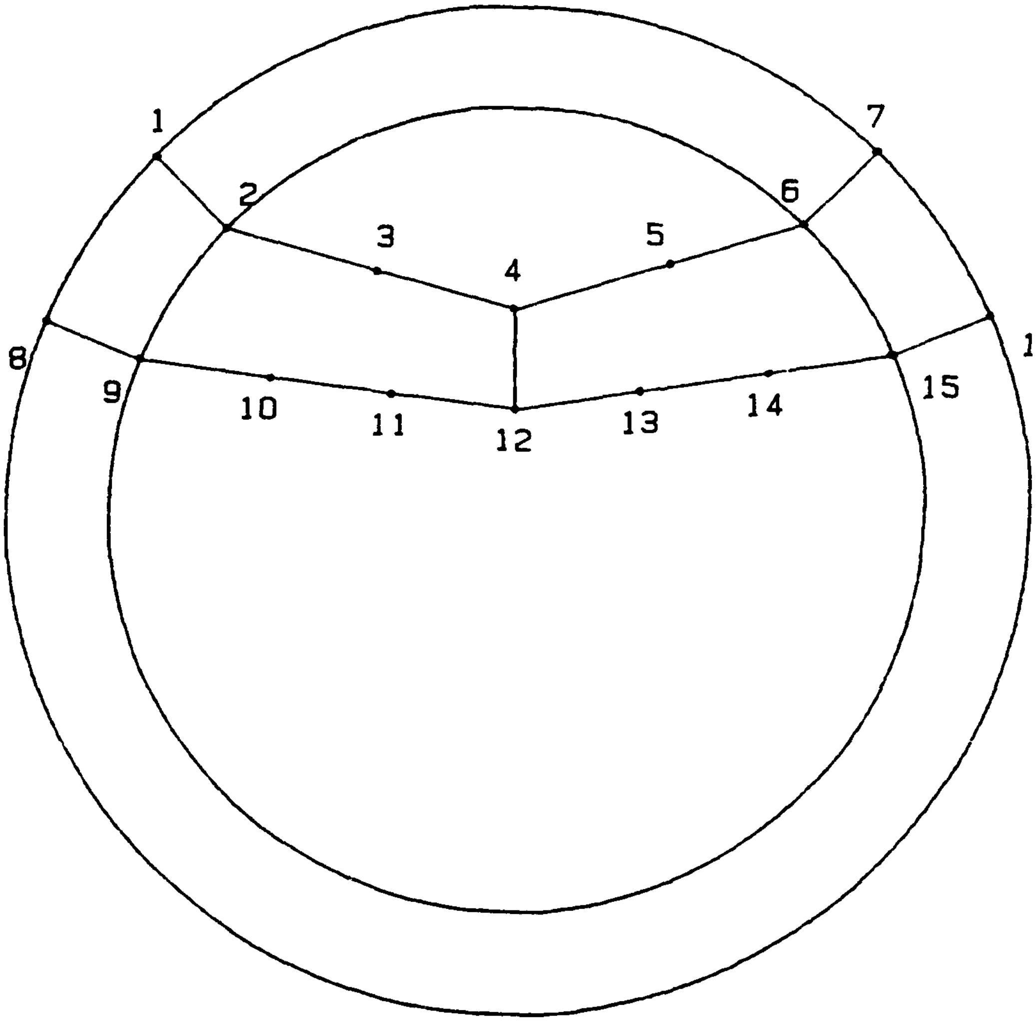

Figure 12. Element Row Numbering Scheme

A mesh orientation attribute is set for the left and right portion of each element row. This attribute determines if the object has a clockwise or counter-clockwise orientation. See Figure 13 on page 31. Switching between the four available mesh orientations allows for a mesh that more accurately conforms to the input object perimeter without resulting in a disproportionate mesh structure.

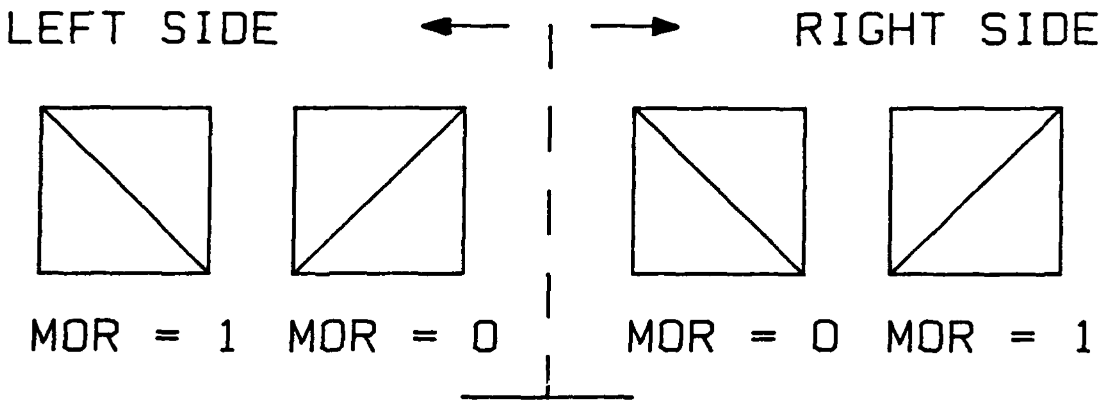

Figure 13. Mesh Orientation Attributes

There is an additional requirement that the number of nodes on a given left or right half row must be only one more, equal to, or one less than the number of nodes on the adjacent left or right half row. Thus, a two sweep process is utilized. This process ensures that this requirement is met and that the first and last row have only two nodes. The second and second from last row (if present) must have three nodes. It is possible, as shown in Figure 14 on page 32, to generate a mesh that has only three rows. This object was input as a circular cylinder. It is evident that, due to a small number of elements, the generated mesh more accurately resembles a square. This will be a problem for calculating the scattered fields, however, the internal fields can be accurately approximated. In this special case, the second and second from last rows are the same. It will be shown, in Chapter V, that since this mesh does not closely approximate the input objects perimeter the resulting Green's function contour integrals and subsequent far field calculations have reduced accuracy.

The nodes of the nodal rows form the vertices of triangular elements. An element, as defined in Chapter II, has three vertices, each assigned a unique number (1, 2 or 3). The ordering of these vertices depends on the mesh orientation attribute. Each element also has an associated relative dielectric constant (ε_r) and relative permeability (μ_r).

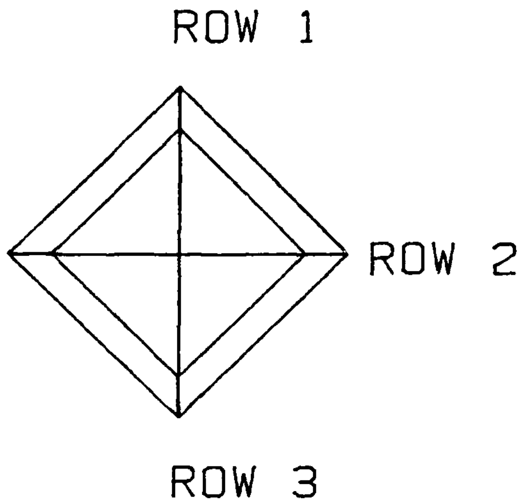

Figure 14. Three Row Cylinder

6. Sorter

This subroutine generates a complete mesh row and all the element/node interconnection relationships. With the nodal structure in place, individual nodes can be assigned to individual elements within the global mesh structure. Each element in a given row is assigned a unique local element number starting with the left most element (relative to the bisection segment). See Figure 15.

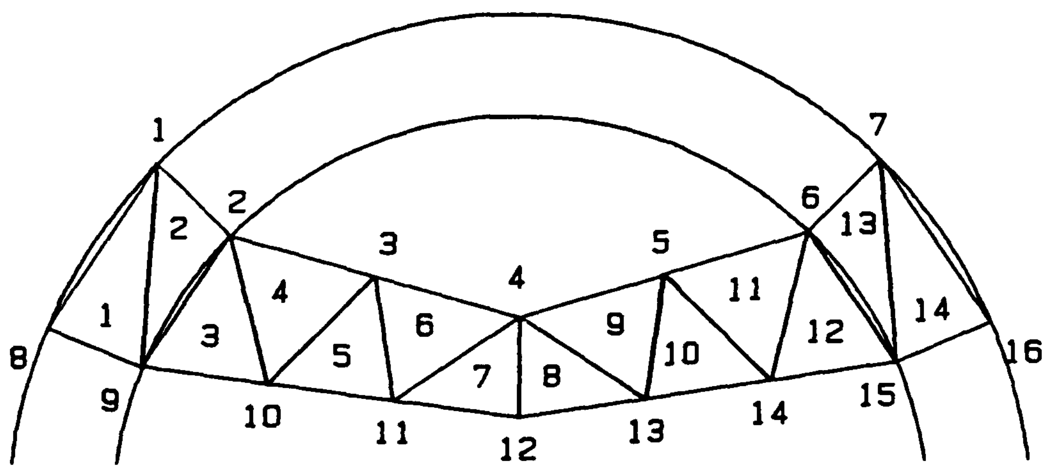

Figure 15. Element Numbering Scheme

It is vital that a method of determining which nodes form the vertices of a given element and which elements have a vertex attached to a given local node. The nodes that are not part of the offset contour have unknown field values. A single node can be connected to as many as six elements, only four of which can be in the current row. This relationship is shown in Figure 16(a). This hexagonal arrangement of six elements is very common within the mesh. Along the bisection segment or where the mesh orientation attribute changes from one row to another, this pattern is disrupted. An example of an extreme case (the center of a circular cylinder) is shown in Figure 16(b). After all elements and nodes in an elemental row are assigned, an ordered sweep is conducted of that elemental row to determine which nodes are connected to which elements, which elements are connected to which nodes, and how many elements a single node is connected too. A complete row has now been generated. See Figure 17 on page 34.

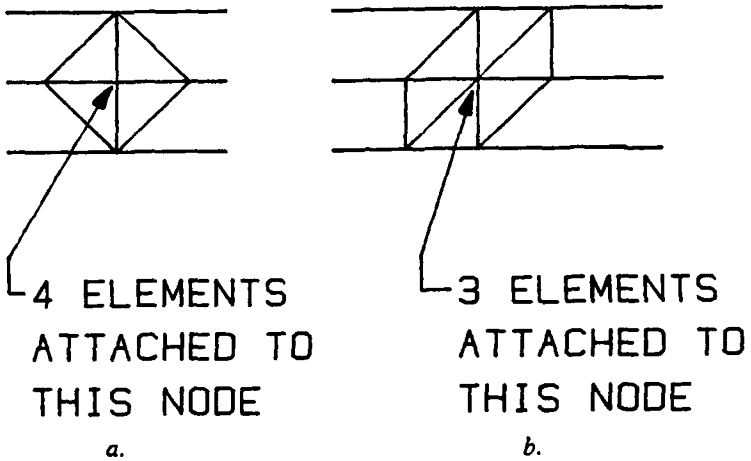

Figure 16. Two Possible Element Intersections, (a) extreme, (b) normal

The information necessary to generate the rows and elements will be used again, in Chapter IV, to solve the (FEBV) problem.

7. Finder

This subroutine determines the x, y coordinates of each node. This data is also necessary to solve the FEBV program of Chapter IV and provides the plotting coordinates for the PLT.DAT file. This process is repeated for all rows. Thus, the entire object is generated as the compilation of elemental rows made of triangles. See Figure 18 on page 34.

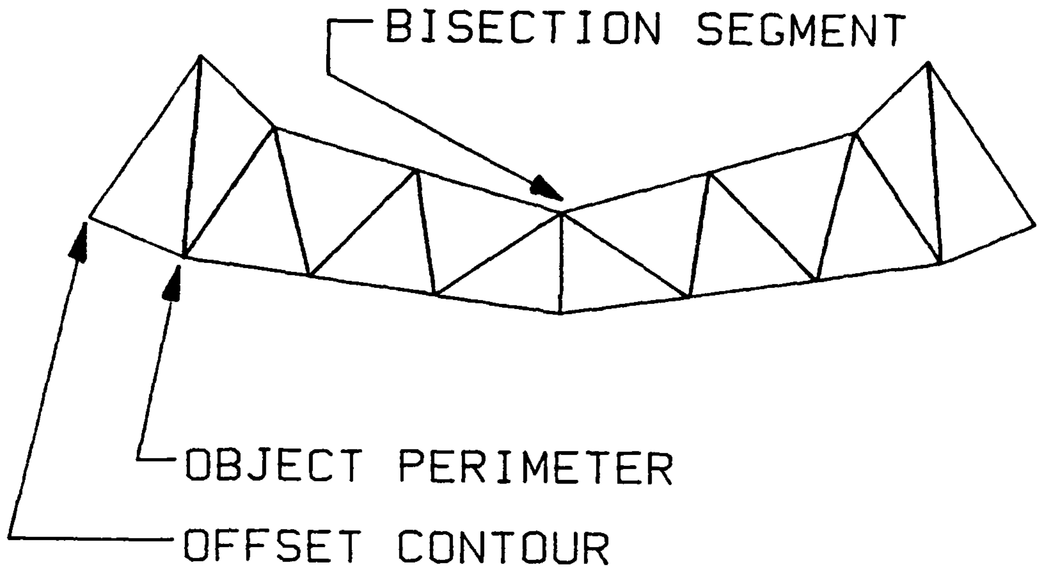

Figure 17. Typical Row Structure

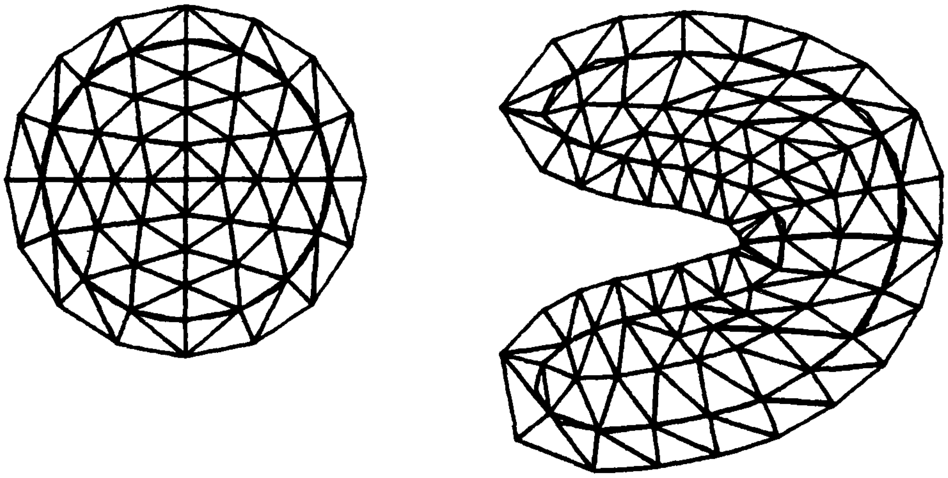

Figure 18. Global Mesh Structure

A file is now available for display using a commercially available program called "CURVE-DIGITIZER". Any program that can accept x, y coordinate data will accomplish the same display process. Minor changes to the MESH program provided in Appendix B may be necessary since several "CURVE-DIGITIZER" plotting codes are embedded in the file generation code.

D. OPTIMIZATION OF THE MESH

For most objects, it is recommended that the MESH program be executed in the mesh generation only mode (Field 6 set to "M" or "m") prior to the execution of the total finite element program. This will ensure that the desired mesh structure is obtained prior to solving for the unknown field values. The MESH (generation only) program requires only a few seconds for even the most dense mesh structures. It is readily seen that as the mesh resolution increases, the number of rows increases linearly while the number of unknowns increases geometrically. Therefore, the mesh calculation times may not change appreciably when the mesh is made more dense, however, the calculation time for the finite element program will rise geometrically. For this and other reasons, the mesh density should be kept low. This will lead to a smaller number of unknowns and result in faster program execution times. Figure 19 on page 36 plots this relationship for a circular cylinder. The solution for these unknown field values will be a topic of Chapter IV.

Six different mesh generation methods are available to create mesh structures. Some of these methods were evolutionary in nature and provide limited practical benefit. Methods 1 and 6 are by far the most useful. Each method is discussed below.

Method 1 constructs the bisection segment by connecting the first input data point to the midpoint of the perimeter. This segment is divided into equal length segments each separated by a node. This method is useful for very simple objects such as the circular cylinder shown in Figure 20 on page 37. This circular cylinder will be used in the explanations of all mesh generation methods. These illustrations are not intended to optimize the mesh structure but rather allow for easy comparison of the 6 basic methods.

Method 2 constructs the bisection segment as described in method 1. The bisection segment nodes are, however, ordered differently. This segment is divided by connecting line segments from corresponding nodes on the left and right side perimeter. Where these line segments intersect the bisection segment, a node is placed. This leads to unequally spaced bisection segments. This can be seen in Figure 21 on page 37.

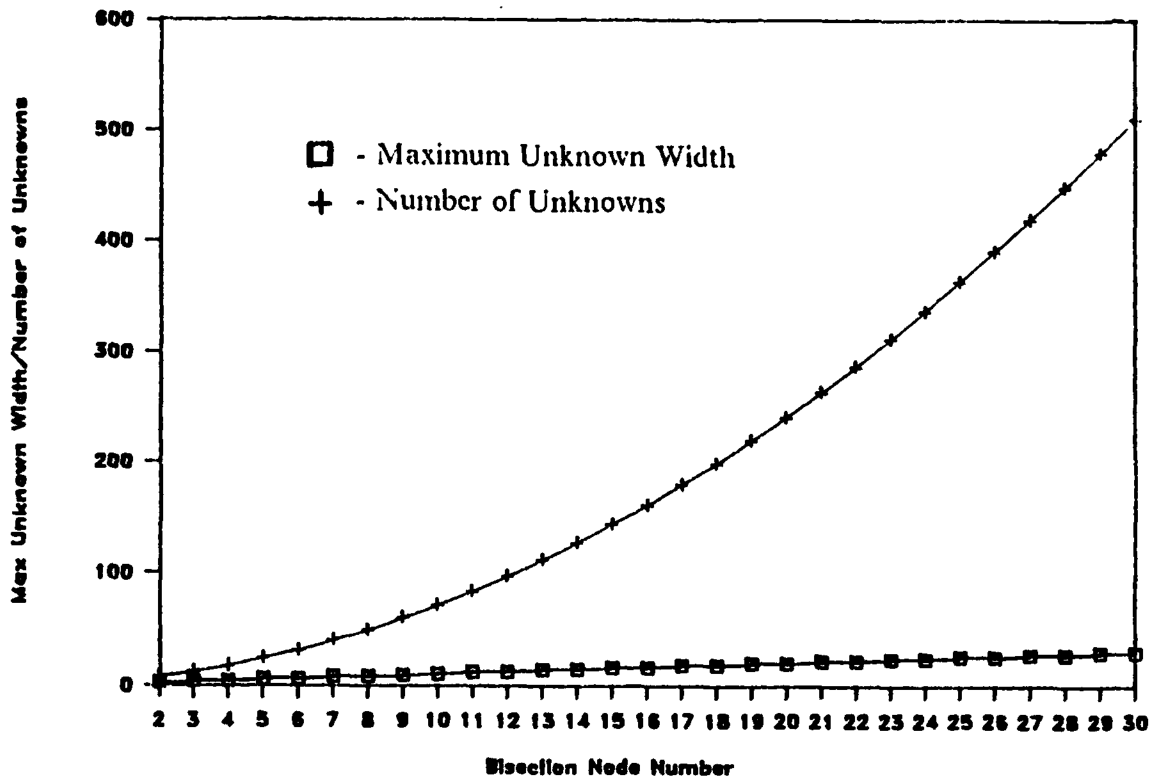

Figure 19. Unknowns and Number of Rows Versus Mesh Resolution

Method 3 modifies method 1 by allowing the user to specify, using Field 17, the stop node for the bisection segment. This method can be useful to rapidly adjust around slightly irregular objects. This can be seen in Figure 22 on page 38.

Method 4 combines methods 2 and 3 by using the connected line segment technique of method 2 to determine the node positions on the bisection segment, the stop node of which is specified as in method 3. This can be seen in Figure 23 on page 39.

Method 5 improves upon method 4 by repositioning the unequally spaced bisection nodes. Linearly interpolated positions for the nodes leads to equal spacing. This method reduces the node "bunching" that frequently occurs with methods 3 and 4. This can be seen in Figure 24 on page 40.

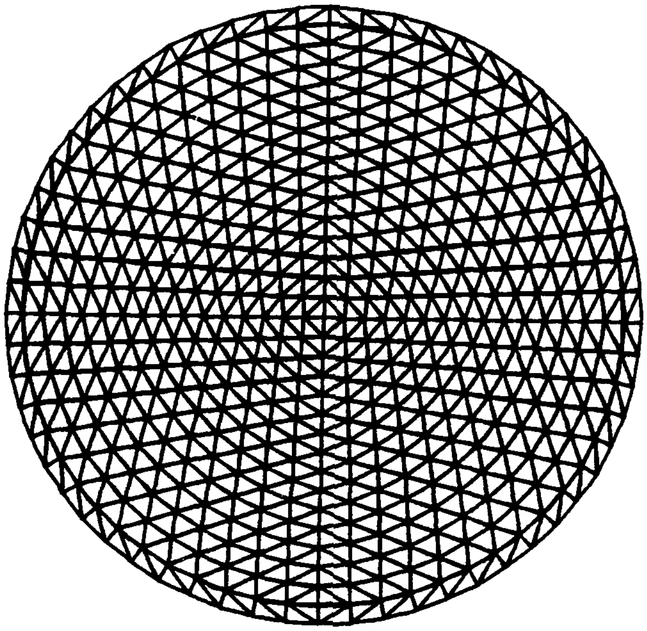

Figure 20. Method 1 Mesh Structure Example

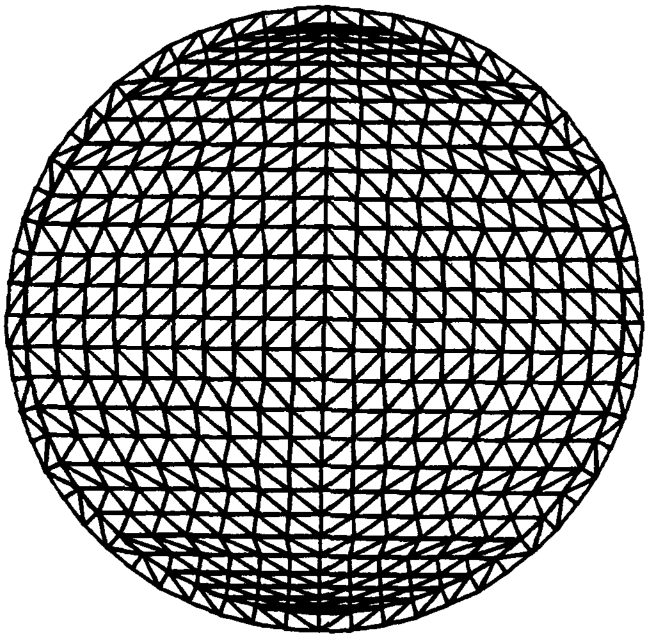

Figure 21. Method 2 Mesh Structure Example

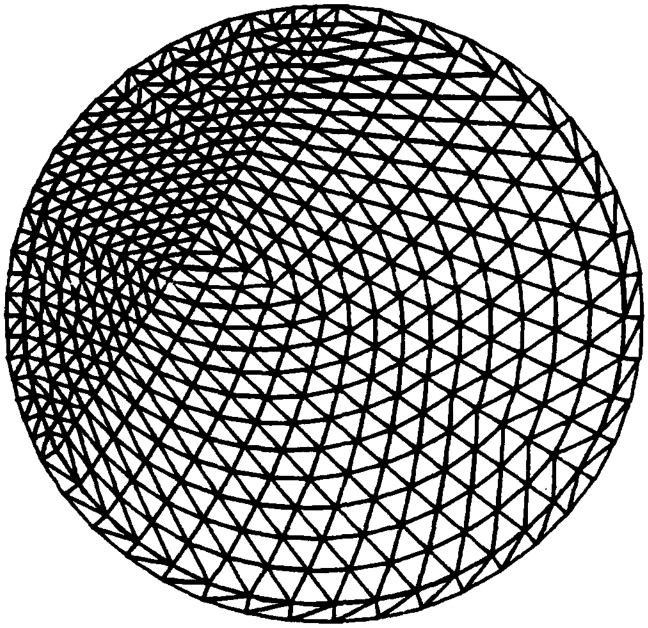

Figure 22. Method 3 Mesh Structure Example

Method 6 is the final improvement in which method 5 was modified to allow for a user-specified start node. This provides the user with the ability to start and stop the bisection segment at any input node without having to rearrange the input data. Since method 6 contains all of the capabilities of the other five methods, it is almost exclusively used for mesh generation. This can be seen in Figure 25 on page 40. There are situations that could be best served by one of the other methods. For example, a circular cylinder and other simple symmetric objects can be represented using method 1. Method 2 would work well with a square or diamond shape. These objects would be bisected by a diagonal. Method 3 would be most suited for a symmetric object with a planar material interface. The more dense mesh would be used for the higher permittivity material. Method 4 would work well with a rhombus or other slightly asymmetric object. Method 5 is almost as versatile as method 6, and would work well in any situation where the start node is fixed. A great deal of planning is not needed in designing most mesh structures since they are calculated and displayed in a matter of seconds.

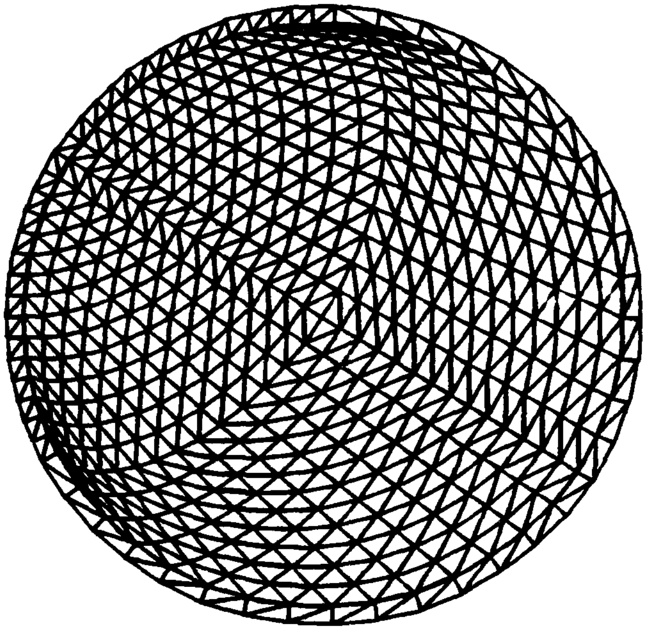

Figure 23. Method 4 Mesh Structure Example

Iterative selection of different mesh generation parameters has proven to be the best technique. A summary of all mesh generation capabilities is provided in Table 1.

Appendix C contains a program called READ.FOR. This program takes the output data file from the "Curve-Digitizer" CAD program, called FINALDWG.DAT, and after receiving the answers to several prompted questions, creates a new INPUT.DAT file. Typically, the CAD program is used to generate the perimeter of the object. This may be as a series of points, line segments or a combination of the two. The answers to the prompted questions provide the additional information needed to fill the remaining data fields. This is intended to be a first step towards allowing a user to specify or design an object and then be able to calculate the scattered fields from this object. Although it is far from efficient, it does serve a definite purpose. Further refinement of this program will allow for an easier user interface. This should enable technically trained personnel, without a detailed understanding of these programs, to benefit from the Field Feedback Formulation.

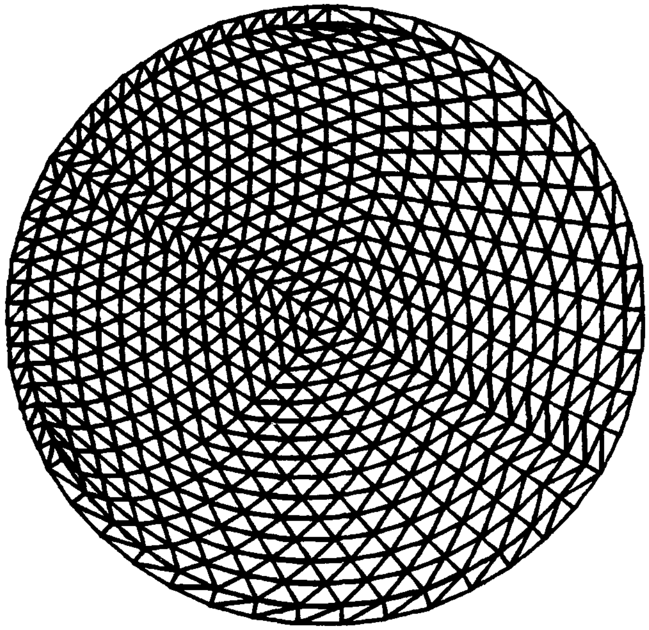

Figure 24. Method 5 Mesh Structure Example

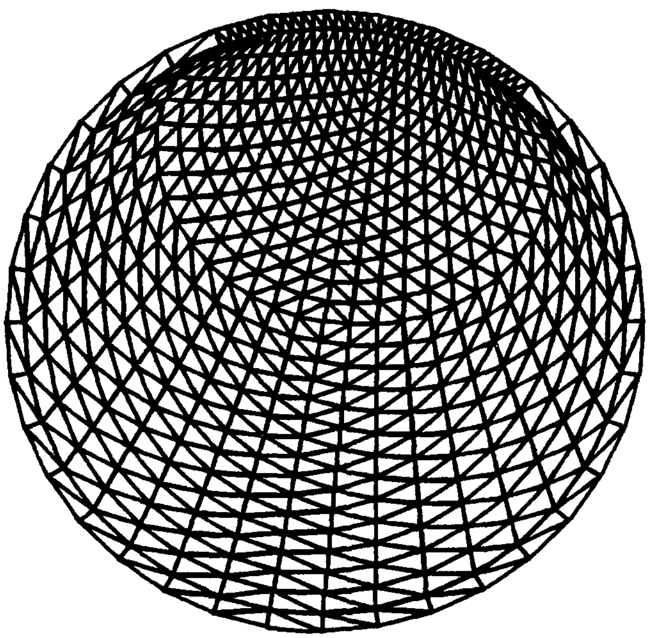

Figure 25. Method 6 Mesh Structure Example

Table 1. SUMMARY OF MESH GENERATION CAPABILITIES

Method Number	Straight Line Bisection	Straight Line From Left to Right Side	User Selected Stop Node	Equally Spaced Bisection Nodes	User Selected Start Node
1	X			X	
2		X			
3	X		X		
4		X	X		
5			X	X	
6			X	X	X

IV. FINITE ELEMENT BOUNDARY VALUE PROGRAM

A. INTRODUCTION

The Finite Element Boundary Value (FEBV) program is the feed forward (U) operator in the Field Feedback Formulation (F^3) as shown in Figure 2 on page 4. This program solves the Helmholtz equation, as discussed in Chapter II, for the Dirichlet boundary condition specified in the input data file, as discussed in Chapter III. These boundary conditions are imposed on the offset boundary contour. The purpose of this program is to find the unknown field values inside the offset boundary contour within the input object. Since the ultimate goal is to obtain the scattered far fields from the object, the unknown field values on the perimeter are of primary interest. It will also be necessary to approximate the normal derivative of the field at the object perimeter. As derived in Chapter II, the goal of this program is to solve,

$$[A_i]\Psi_{i-1} + [B_i]\Psi_i + [C_i]\Psi_{i+1} = -[P_i]\Psi_i.$$

The A matrix represents the effect that the I − 1 row has on the I^{th} row values. Similarly, the B matrix represents the effect that the I-th row has on itself, while the C matrix represents the effect that the I + 1 row has on the I-th row. All other rows do not effect the I-th row. This is due to the pyramidal basis functions, discussed in Chapter II, that all have zero value when a node is not directly connected to another node. The P matrix represents the combined effects of the boundary nodes on the I-th row. These values are transposed to the right side of the equality to form the system forcing function. It is worth remembering that for the I = 1 row, the A matrix = 0 and that for the I = IMX row, the C matrix = 0. Thus, using the row by row stepping process, first discussed in Chapter III of the mesh generation process, the A, B, C and P matrices can be filled. There is an A, B, C and P matrix for each row, and it is necessary to calculate the functionals for two elemental rows to fill one set of matrices. It is, therefore, obvious that the data is used twice, once for the current row and again for the next row. Specifically, the functionals necessary to complete the B matrix and totally fill the C matrix is used again to totally fill the next row's A matrix and partially fill the B matrix. Thus, for a row I - 1, (such that I - 1 ≠ 1), all of the A matrix and part of the B and P matrices are filled. When the row is stepped to I, the B and P matrices are completed and all of the C matrix is filled. The A, B, C and P matrices represent tri-block matrices within a much larger global matrix. This row of tri-block matrices in the global matrix can be partially solved using the forward portion of the Ricatti transform. The Ricatti transform is a numerical technique that optimizes the solution for banded (tri-block) systems of linear equations. The equations are,

$$R_{i+1} = -(B_i + A_i R_i)^{-1} C_i$$

and

$$S_{i+1} = (B_i = (B_i + A_i R_i)^{-1} \bullet (P - A_i S_i)$$

where R_{i-1} is the $i^{th}+1$ R matrix and the S_{i-1} is the $i^{th}+1$ S vector. Note that the next rows R matrix and S vector depends on the previous rows R matrix and S vector. During the forward step (MARCH subroutine) an R matrix and S vector is generated for each row. When all rows have been calculated, a back sweep computes the unknown field values, ψ_{ij}. The equation is

$$\psi_{i-1} = R_i \psi_i + S_i.$$

This back sweep must read the RS data from the disk backwards. This is a storage intensive process for all but the most trivial problem. The field values ψ_{i-1} is first found by remembering that the last row (I = IMX), $C_{IMX} = 0$. With $C_{IMX} = 0$, $R_{IMX-1} = 0$ and $\psi_{IMX} = S_{IMX-1}$, which is the last calculated S vector. The recursion continues until all of the ψ_i 's have been calculated.

B. FINITE ELEMENT BOUNDARY VALUE PROGRAM

The eight subroutines comprising the FEBV program are discussed below.

1. Zero

This subroutine fills all the array positions of the A, B, C and P matrices with 0 + j0. This zeroing is necessary after all calculations concerning a given row are completed.

2. Varint (Variational Integration)

This subroutine calculates the complex functionals for a given input element. The functionals reflect the effect that each node has on the three nodes associated with an element. The subroutine must be provided with the X, Y coordinates of the element nodes, and the material parameters ε_r and μ_r. These functionals are returned in a 3 x 3 complex matrix. The area of the input element is also provided as a by-product of this numerical integration. The areas of each element are sorted and used to find the largest and smallest elements. Once found, the largest and smallest areas are combined to form the area ratio, which is defined as,

$$\text{area ratio} = \frac{\text{maximum area}}{\text{minimum area}} .$$

This ratio should be kept as small as possible. For a circular cylinder, values slightly less than 2.0 are possible. For more complicated objects, the ratio can get considerably larger. A ratio $\geq$ 2.5 causes a screen message of "YOU SHOULD CONSIDER ABORTING THIS RUN AND LOOKING AT THE MESH IN CURVE DIGITIZER. A BETTER METHOD MAY BE AVAILABLE". It may or may not be possible to obtain an area ratio < 2.5. The intention of the area ratio is to insure that a uniform mesh is constructed prior to attempting a problem solution. Smaller area ratios are indicative of mesh structures that do not have grossly different element sizes. This will lead to more accurate finite element solution.

3. Fill

This subroutine calculates and stores all of the functionals for an entire elemental row. This is accomplished by repeated VARINT calls for each element that comprises an elemental row.

4. BNDC (Boundary Condition)

This subroutine calculates and stores the boundary conditions desired for the offset boundary contour. These conditions can be plane wave, cylindrical modes or input manually. For plane wave conditions, the percent error of the FEBV program will also be returned. This calculation only has meaning for the uniform case (Field 5 set to "U" or "u"). Memory limitations do not allow this feature for the cylindrical mode boundary conditions for modes other than zero and one. A separate routine could be made available for offline percent error calculations. No such feature is provided if the boundary conditions are manually input. Any desired boundary condition may be generated by modifying or appending the proper code to this subroutine.

5. Loader

This subroutine loads the A, B, C and P matrices for a given row. For all but the I = 1 and IMX rows, two calls of FILL/VARINT for two elemental rows of data are required to fill the A, B, C and P matrices. This is an additive process that starts after the ZERO subroutine initializes the matrices. Successive functionals are added to the existing data in the proper array positions.

6. March

This subroutine performs the forward portion of the Riccati transform. The output data, called RS.DAT (R matrix and S vector), are stored on disk.

7. CSMINV (Complex Square Matrix Inversion)

This subroutine accomplishes the complex matrix inversion required by the forward Riccati transform. A maximum dimension of 50×50 was established to limit memory utilization. This allows for a maximum unknown width of 50.

8. Sweep

All of the above subroutines are called at least once for each row. The sweep routine is called only after all of the row calculations are completed, and then only once. This subroutine conducts the Riccati back sweep by reading the RS data generated by the MARCH subroutine. This data is read off the disk backwards, using the FORTRAN "backspace" command. The returned field values are actually individual contributions due to a unit valued basis function being individually applied to each of the boundary nodes. In so doing, the problem need only be solved once. After the data is stored, in matrix form as the U.DAT file, the boundary value problem may be solved for any incident field by multiplying the U matrix by the new incident fields. Thus,

$$\psi_{perimeter} = [U] \bullet \psi_{incident}.$$

A summary of the input, output and error data is provided in the OUTPUT.DAT file.

9. Save

This subroutine was necessary to allow for reasonably sized problems. Since all of the Field Feedback Formulation code could not fit into 640 kilobytes of memory, the program was divided into two parts. All of the data necessary to perform the programs is saved in the F3.DAT file. This data is then read by the field feedback program. This technique, though very inefficient, circumvents the 640 kilobyte memory limitation imposed by the IBM Disk Operating System (DOS). Efforts to convert this code to run under a compiler that does not have a 640 kilobyte memory limitation, such as Microway NDP FORTRAN compiler, will be pursued at a later date.

C. VARINT VALIDATION

The first step in the program validation required an understanding of the error convergence of the variational elements as a function of element size, d and material properties, ε_r and μ_r. The test program provided in Appendix D varies the element size, in wavelengths, for a test mesh structure. This test structure shown in Figure 26 on page 46 has only one unknown at the center. The mesh is made of four adjacent elements. These elements are inside a uniform material having properties, ε_r and μ_r. The other four nodes are established as boundary nodes.

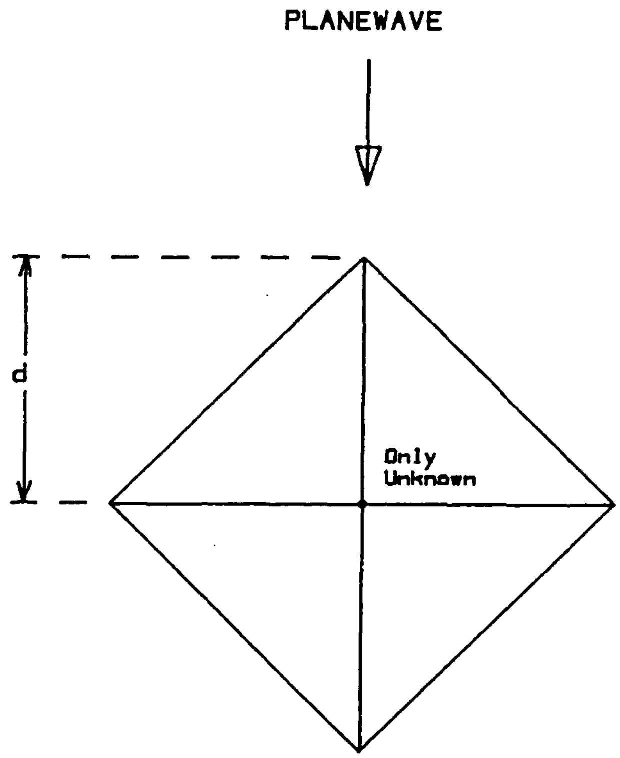

Figure 26. Test Mesh Structure

A plane wave, of user specified frequency and amplitude, is used to determine the boundary conditions on the four boundary nodes. This plane wave is propagating down the vertically oriented axis. The unknown nodal value is calculated and compared to the actual plane wave value (1 + j0). A percent error is calculated as the dimension of the elements, d are reduced. As expected, the solution became more accurate as the elements become smaller. A plot of the results of this test program is provided in Figure 27 on page 47. This data was generated with $\varepsilon_r = 1 + j0$, and $\mu_r = 1 + j0$.

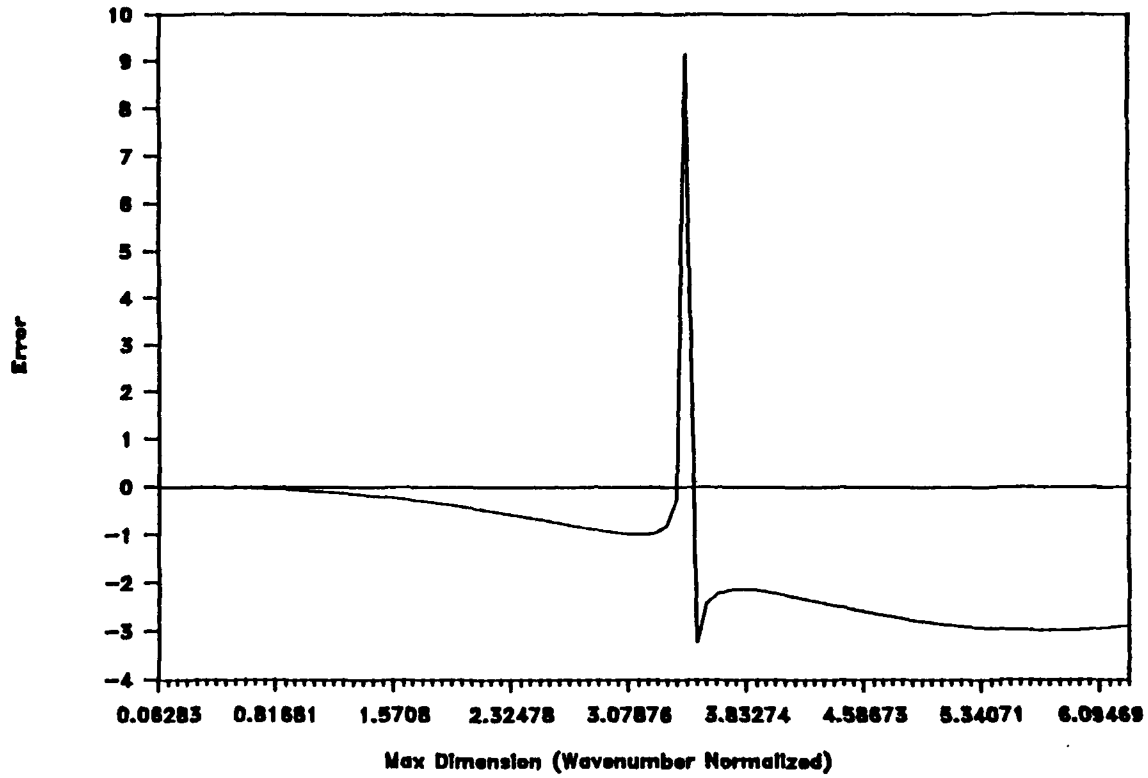

Figure 27. Solution Error for a Test Mesh Structure

The region of interest is where the error approaches zero. An expanded plot of this region is provided in Figure 28 on page 48. The accuracy of the solutions were desired within 1 percent of the exact value. This requires an element that is $\leq \frac{\lambda}{17}$, in the material. To ensure the desired error was achieved, elements were usually scaled to be $\leq \frac{\lambda}{20}$, in the material. The phase of the plane wave was also varied to determine the effect on convergence. No significant effect was noted.

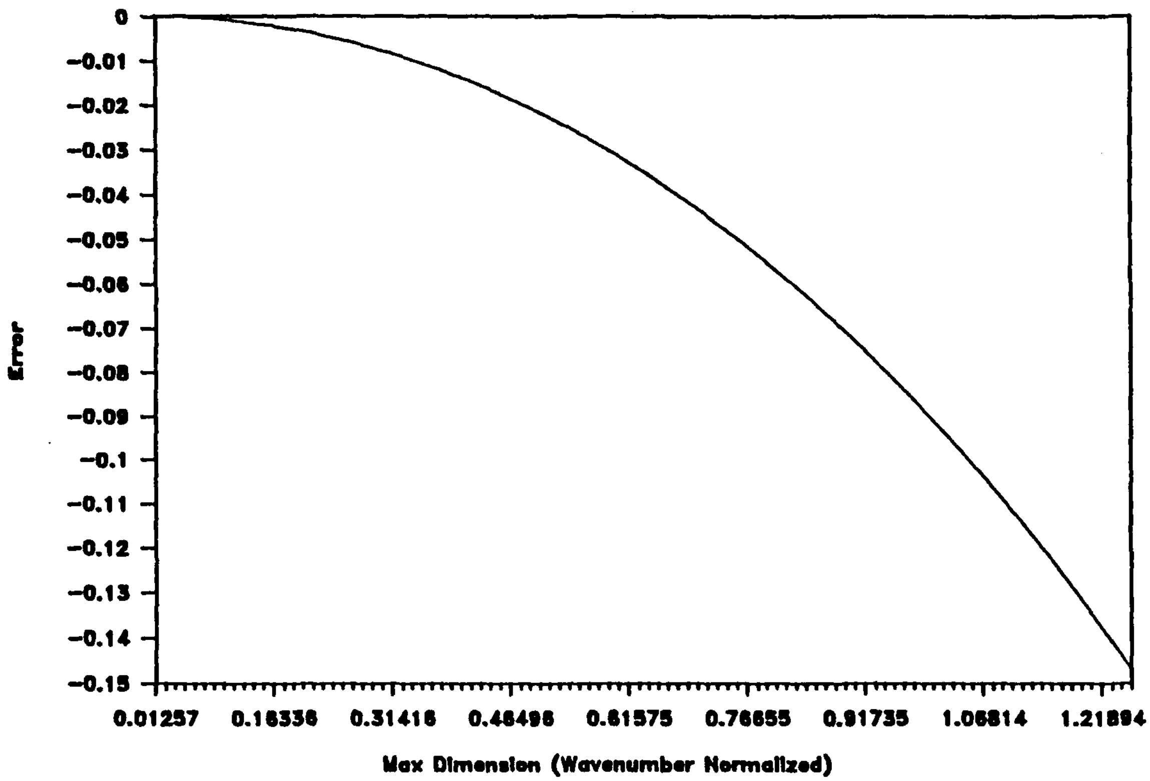

Figure 28. Solution Error for a Test Mesh Structure (Expanded)

D. FINITE ELEMENT BOUNDARY VALUE PROGRAM VALIDATION

The next validation step required an actual object to calculate the fields in and on. The simplest object was actually not an object at all, but rather an imaginary circular cylinder in free space. A plane wave was propagated through this free space. Since no material interface existed, the exact solution would be known for all positions. The plane wave established the boundary conditions on the offset contour. Trial runs were conducted varying the mesh resolution and boundary offset contour distance. With the error defined as,

$$\text{error} = \sqrt{\frac{\Sigma(\text{calculated value} - \text{actual value})^2}{\Sigma(\text{actual value})^2}},$$

two errors were defined. The perimeter error only considers the perimeter nodes in the error calculation. The bisection segment error only considers the bisection segment nodes, with the exception of the two end nodes, in the error calculation. The end nodes of the bisection segment are part of the perimeter and are, therefore, not considered. As expected, the perimeter error could be rapidly reduced by decreasing the offset distance. This is due to the fact that the node or nodes closest to an unknown node dominate the field contributions at this node. Again, a goal of < 1% error was desired. The reduced offset distance had a very small effect on the bisection segment error. The only way to significantly reduce this error was to increase the mesh resolution. An increased mesh resolution reduced both errors. Figure 29 shows the perimeter error as a function of the number of bisection segment nodes.

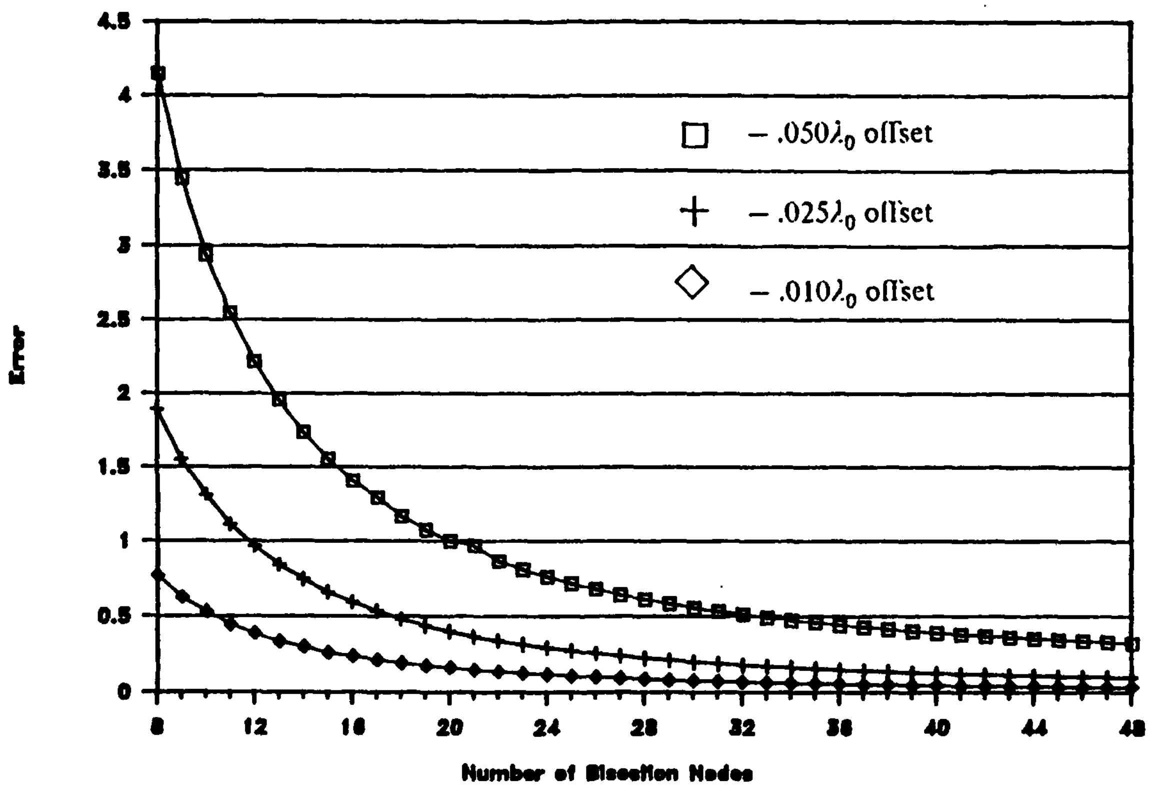

Figure 29. Perimeter Error

The three curves are for offset distances of 0.05, 0.025 and 0.01 λ . Figure 30 on page 50 shows the bisection segment error as a function of the number of bisection segment nodes. The two curves are for offset distances of 0.05 and 0.025 λ_0 . For offset distances less than 0.025 λ_0 , the curves are almost identical to the 0.025 λ_0 curve. These curves are omitted for clarity.

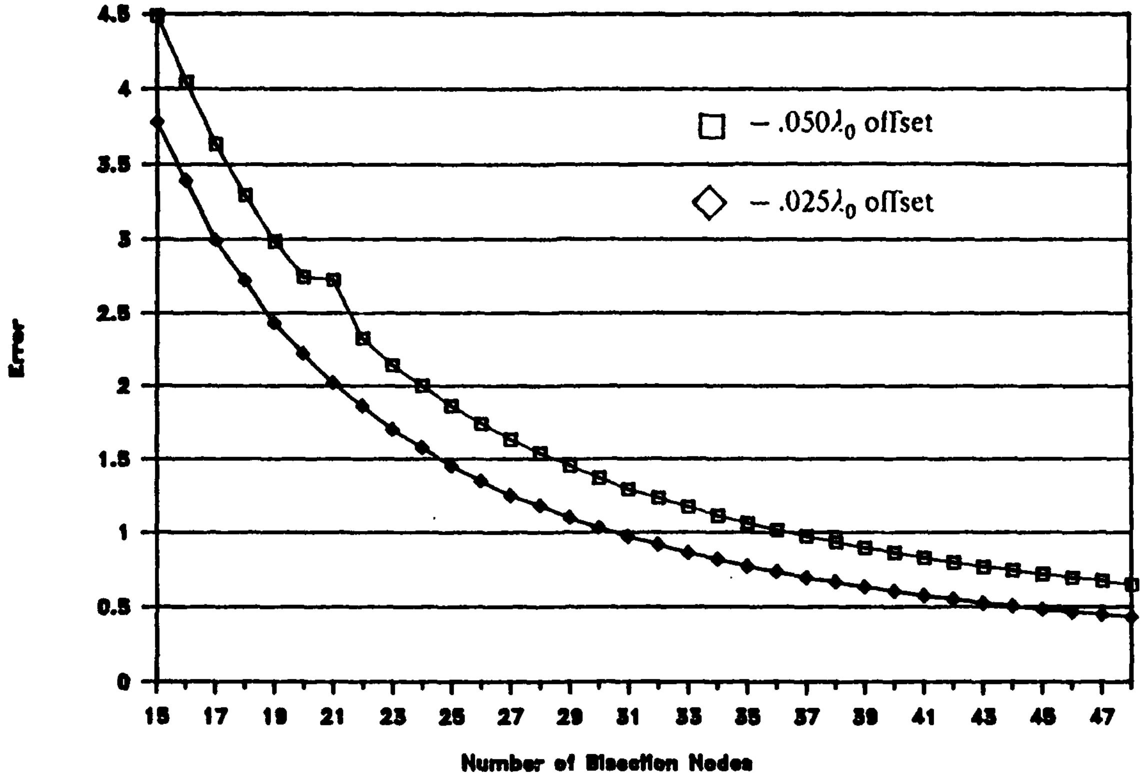

Figure 30. Bisection Segment Error

The calculated and exact field values for a 0.5 λ_0 diameter circular cylinder, with a $0.05\lambda_0$ mesh resolution and offset distance and $\varepsilon_r = 1 + j0$ is shown in Figure 31 on page 51. The exact fields values for the real and imaginary portion of the plane wave are shown as squares and diamonds, respectively. The solid curves are the calculated field values. The perimeter and bisection errors were 0.74 and 1.69 percent, respectively. Figure 32 on page 52 shows the effects of not properly adjusting the mesh resolution and offset distance when the material is changed. In this case, the permittivity was changed to $\varepsilon_r = 4 + j0$. This caused the perimeter and bisection segment errors to increase to 18.2 and 41.1 percent, respectively. The mesh resolution and offset distance were reduced to $0.021\lambda_0$ and the permittivity was changed to $\varepsilon_r = 4 - j4$. The results are shown in Figure 33 on page 53 . This proper selection of mesh resolution and offset distance reduced the perimeter error and bisection segment error to 0.44 and 0.56 percent, respectively. Note that in the lossy material case the two errors are very close in magnitude. This is due to the way that the error is defined. This definition, in a lossy material, overemphasizes the objects leading edge.

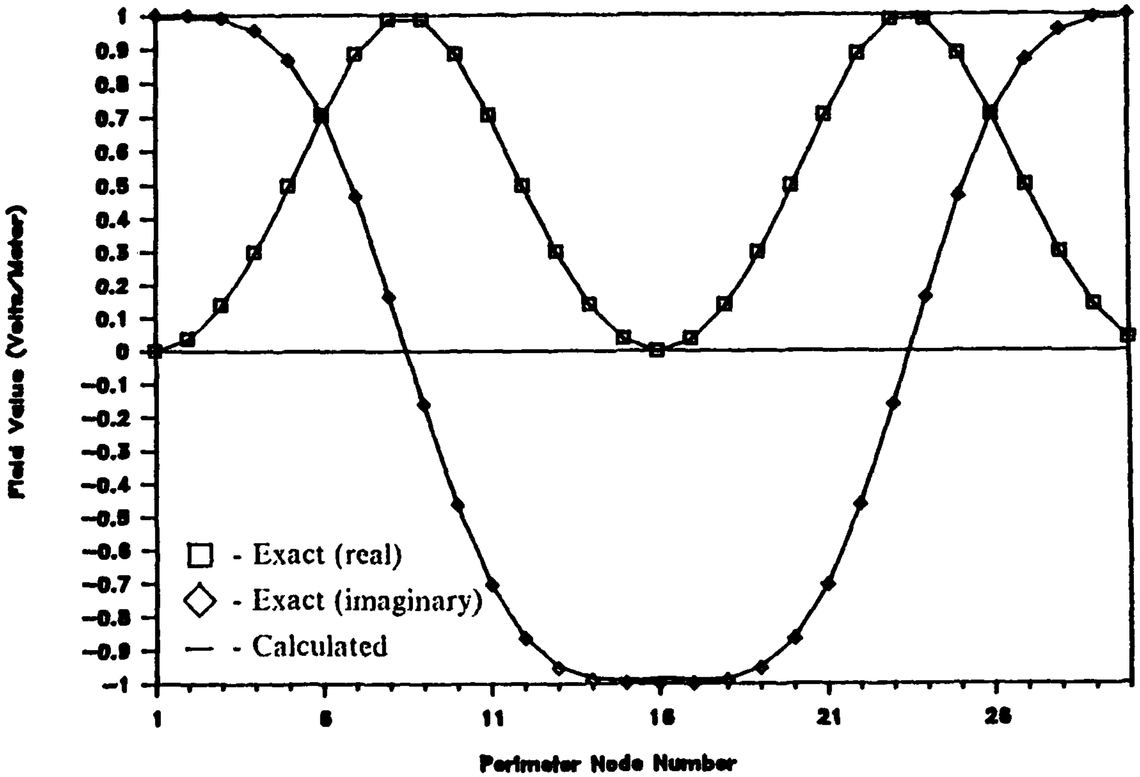

Figure 31. 0.5 λ cylinder, $\varepsilon_r = 1 + j0$

This portion of the object (as well as the trailing edge) is the most accurate for the bisection segment calculations. This is because of the relative closeness of these nodes to the perimeter. For the lossless material, the bisection segment error is typically two to three times the perimeter error, for the same number of bisection nodes. The conclusion that the offset distance should be made as small as possible in order to minimize the perimeter error is correct, but will prove to be counterproductive in the long run. There is a competing effect that will require this contour offset distance to be as large as possible. This effect, associated with the accuracy of the Green's function contour integral, will be discussed in Chapter V.

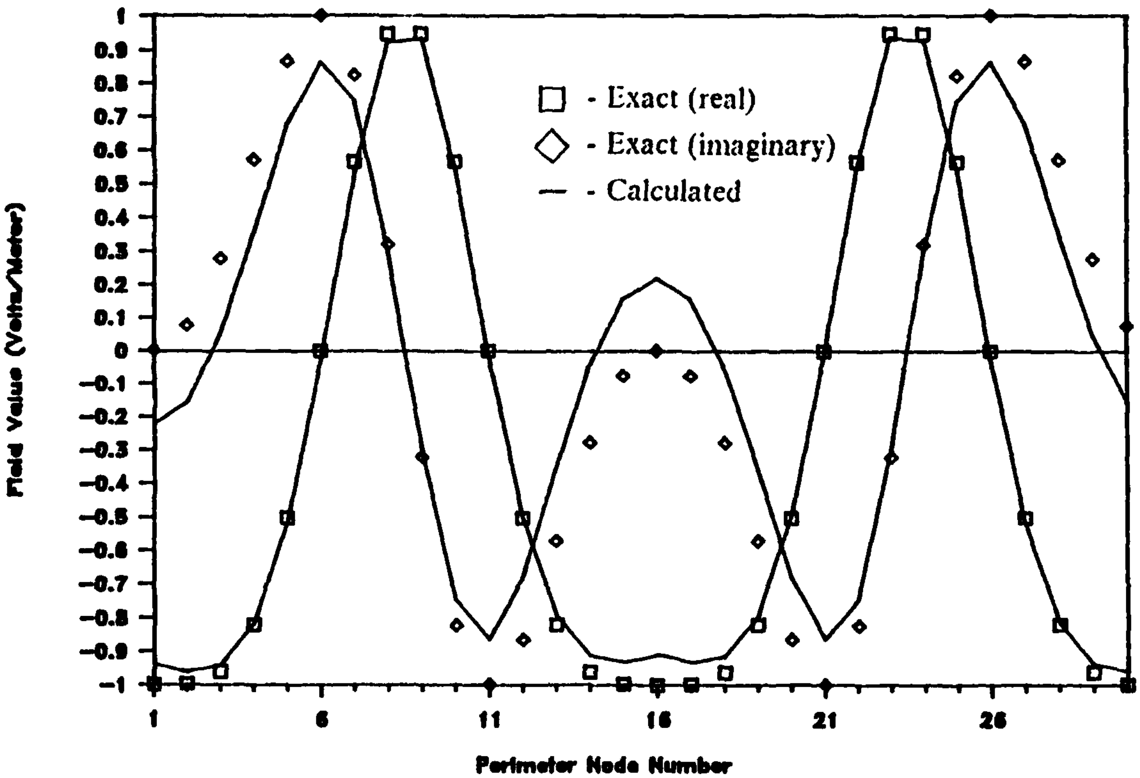

Figure 32. 0.5 λ cylinder, $\varepsilon_r = 4 + j0$

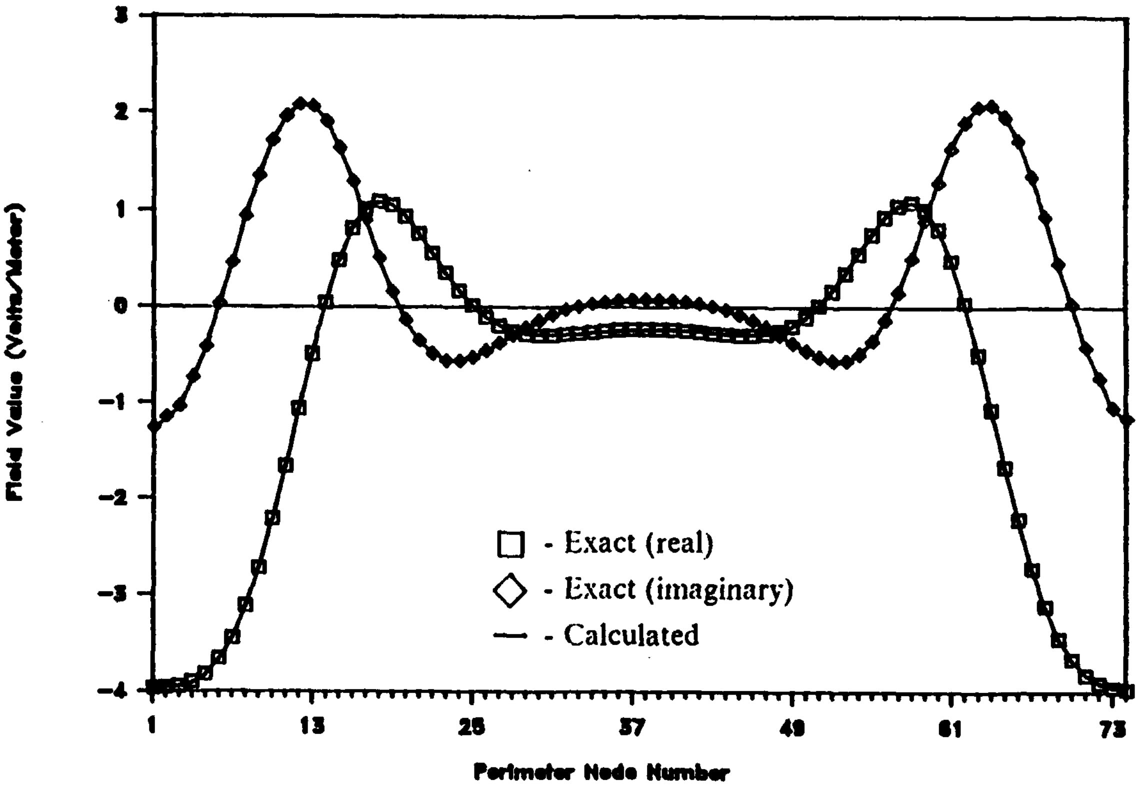

Figure 33. 0.5 λ cylinder, $\varepsilon_r = 4 - j4$

E. INHOMOGENEITY

Until this point, all testing was done in circular homogeneous dielectric materials. It was, at a minimum, desirable to place the object in a vacuum to calculate the scattered fields. This requires the first two and last two elements of each row to have $\varepsilon_r = 1 + j0$ and $\mu_r = 1 + j0$. These elements represent the space between the objects perimeter and the offset boundary contour. Since the plane wave solution is no longer valid for this geometry, a new problem with a known solution was needed.

Given a homogeneous dielectric circular cylinder of radius a and permittivity ε_r, , as shown in Figure 34 on page 54, it can be shown that [Ref. 8],

$$\psi_n(R, \phi) = A_n J_n(k_r R) \cos n\phi,$$

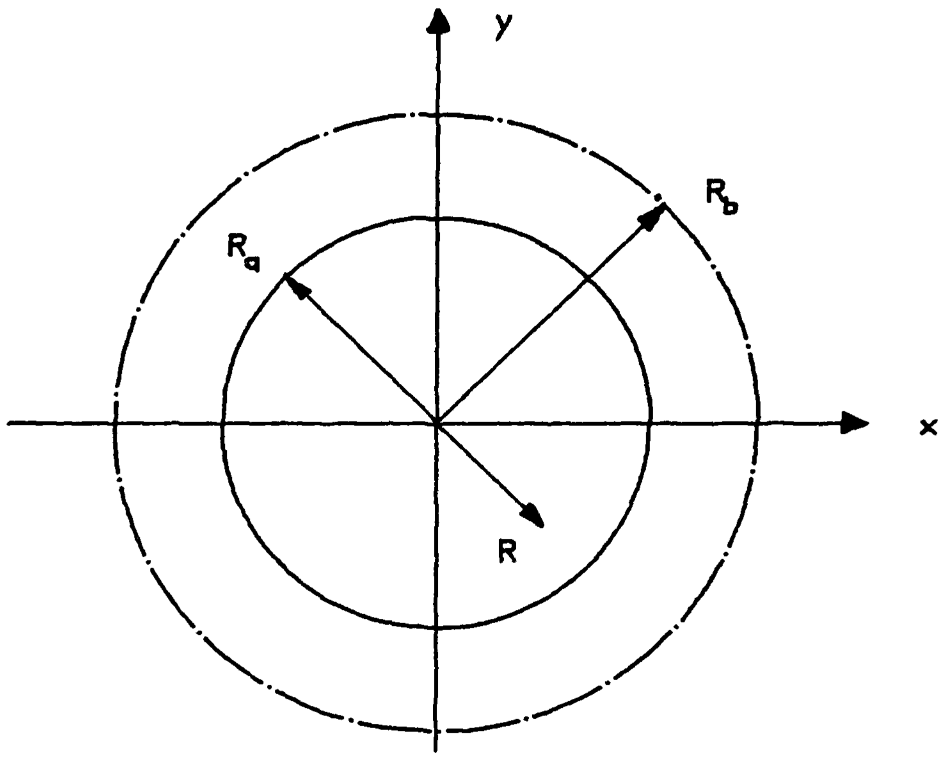

Figure 34. Cylindrical Mode Geometry

where,

$\psi_n(R, \phi)$ is the field value inside the dielectric material.

$$A_n = \frac{-j2D_n}{\pi a \nabla_n}$$

$$\nabla_n = \frac{J_N(R_b)\left[J_n(k_r R_a) H_n^{(2)\prime}(R_a) - k_r J'_n(k_r R_a) H_n^{(2)}(R_a)\right] -}{H_n^{(2)}(R_b)[J_n(k_r R_a) J'_n(R_a) - k_r J'_n(k_r R_a) J_n(R_a)]}$$

$J_n =$ Bessel Function of the 1^{st} kind of order n

and

$$k_r = \sqrt{\varepsilon_r}\,.$$

An additional formulation is required for the fields outside of the cylinder. A cylindrical wave (mode, n = 1) was applied to a homogeneous dielectric cylinder. The exact and finite element field values were calculated and compared. Figure 35 on page 55 , shows a typical result. Mesh resolution, offset distance and material permittivity were varied to determine convergence. The results were similar to the dielectric homogeneous plane wave case discussed earlier.

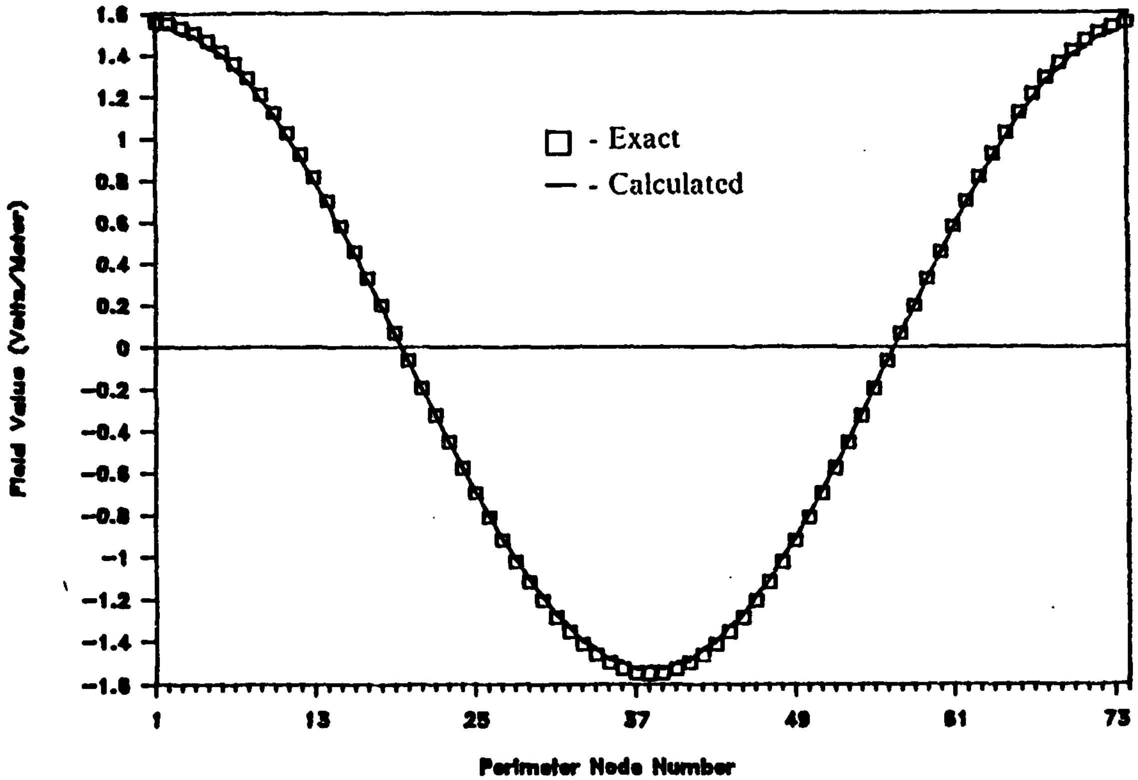

Figure 35. Typical Cylindrical Mode Boundary Value Problem Result

F. FINITE ELEMENT CONCLUSIONS

High accuracy solutions can be obtained by maintaining a mesh resolution of $< \frac{\lambda}{20}$ in the material. The offset distance should be kept at approximately the same magnitude as the mesh resolution, but may be as large as $\frac{\lambda_0}{20}$. Increased accuracy requires an increased mesh resolution. This increase in accuracy is at the expense of increased processor time.

V. FIELD FEEDBACK PROGRAM

A. INTRODUCTION

The Field Feedback Program is the feedback (T) operator in the F^3. This program utilizes the output of the finite element program and the input incident fields to evaluate a "near field" Green's function integral. This integral is called "near field" in that the integral will be performed within $\frac{\lambda_0}{20}$ of the object perimeter. As seen in Figure 36, three surface contours are defined. The boundary contour is where the incident field boundary condition is applied.

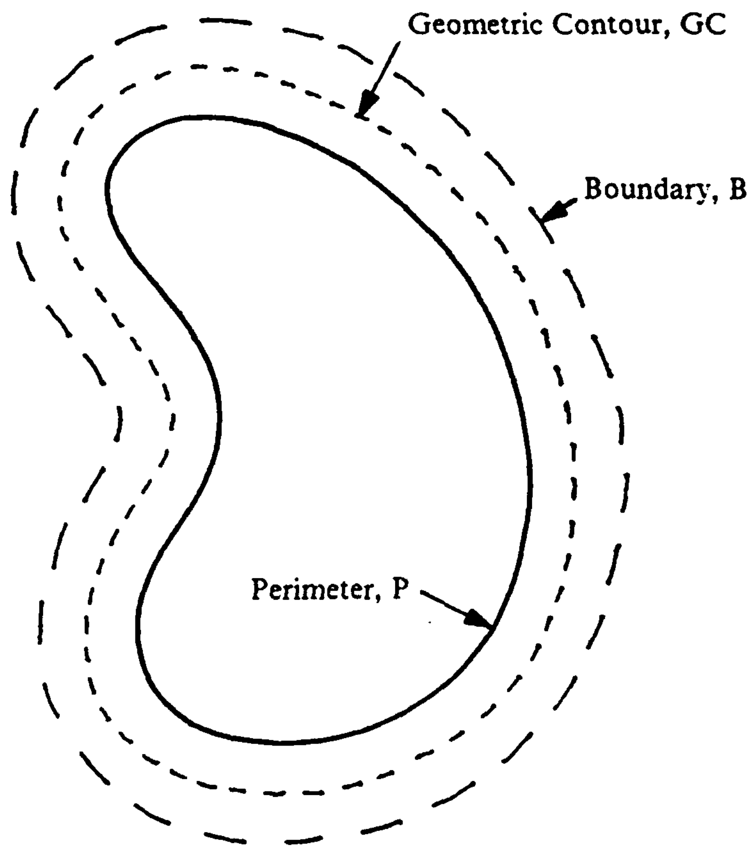

Figure 36. Contour Arrangement

The perimeter contour is where the finite element program solves for the field values on the object perimeter. The geometric contour is midway between the boundary and the perimeter and is the contour over which the Green's function contour integral is performed. This is necessary to allow the $\frac{\partial\psi}{\partial n}$ to be approximated by a finite difference technique. The ψ_{GC} is the average of the field values on the boundary and perimeter. Thus,

$$\frac{\partial\psi}{\partial n} = \frac{(\psi_{boundary} - \psi_{perimeter})}{\partial n}$$

and

$$\psi_{GC} = \frac{(\psi_{perimeter} + \psi_{boundary})}{2}.$$

The Field Feedback program is provided as Appendix E.

B. FIELD FEEDBACK PROGRAM

1. Input

This subroutine reads the finite element data stored in the F3.DAT file by the SAVE subroutine. All necessary nodal X, Y coordinates are calculated.

2. TMAT (T Matrix)

This subroutine loads a complex matrix called TMAT. Each element of the T matrix is the result of a Green's function contour integral. This integral is conducted on the GC contour. The m-th column of the T matrix is evaluated with a single unit valued basis function on the m-th boundary node. The equation,

$$T_{n,m} = \int_{GC} \left[G \frac{\partial \psi_{\hat{n}}}{\partial n} - \psi_n \frac{\partial G}{\partial n} \right] dGC,$$

may now be numerically evaluated at each of the n boundary nodes. This process is repeated for each row until the entire matrix is filled.

The numeric integration uses a rectangular mid-point approximation technique. For this linearized problem, this is equivalent to a trapezoidal integral technique. The integral path between any given two nodes is subdivided based on the distance to the first point. This subdivision maybe controlled by three input variable fields. A multiplicative scale factor and offset terms are available. To date, the utilization of the scale factor (set > 1.0) and the offset terms (set > 0) have not proved necessary. This may be necessary for more complicated geometric structures.

3. CNSOLV

This subroutine solves the equation,

$$C_n = [I - T_{matrix}]^{-1} \bullet \psi_{boundary},$$

where C_n is the total scattered fields on the boundary, I is the identity matrix and $\psi_{boundary}$ is the initially specified incident fields.

4. FFLD (Far Fields)

This subroutine calculates the scattered far field or bistatic radar cross section per unit length of the object. Any desired integer angular resolution for the calculation may be specified. The final results are stored in the FFPAT.DAT file.

C. FIELD FEEDBACK VALIDATION

Given a plane wave boundary conditions imposed on an imaginary cylinder in free space, a Green's function contour integral may be performed around the cylinder. This cylinder is imaginary in the sense that it does not actually exist. The cylinder is actually an artificial boundary that does not form a material interface. For points inside this cylinder, the resulting integral should equal the negative of the actual plane wave value at that point in freespace. For points outside the cylinder, the resulting integral should equal zero. Several test conditions were investigated to ensure that the numerical integration of the Green's function did arrive at the expected values. Maximum absolute errors for these tests were less than 7×10^{-7}. These test cases started with exact values of the field, ψ and the normal derivative, $\frac{\partial \psi}{\partial n}$. After initial validation, finite element calculated ψ and $\frac{\partial \psi}{\partial n}$ values were used. Errors increased by approximately a factor of 10. Numerous competing effects were observed with two dominant effects becoming evident.

1. Small Object Phenomenon

For very small objects, where only a few elements are needed to meet the $\frac{\lambda}{20}$ mesh resolution requirement, the normal derivative accuracy is crucial. Since the normal derivative is the calculated normal to the piecewise linear approximation of the objects perimeter, this fit is usually not adequate. To prevent this problem, additional perimeter/boundary nodes are needed. This is easily accomplished by increasing the mesh density, which is controlled by the mesh resolution (input field 7). For these very small objects, the RCS is almost uniform for the TM case and co-sinusoidal for the TE case. The utility of this calculation must, therefore, be questioned.

2. Offset Distance Phenomenon

The offset distance must not be made too small or the accuracy of the Green's function integral will suffer. For a circular cylinder, the optimal offset was between 0.03 λ and 0.035 λ. Note that these distances are always $< \frac{\lambda_0}{20}$.

VI. VALIDATION

A. INTRODUCTION

The difficulty in the total Field Feedback Formulation (F^3) program validation is in finding problems that have well established or accepted solutions. The total F^3 program is the combination of the mesh generation/finite element program and the field feedback program. If possible, a problem that has a closed form analytical solution is desired.

B. HOMOGENEOUS CIRCULAR CYLINDRICAL SCATTERING

This is the first test case for the F^3 program. A penetrable circular cylinder has a closed form analytic solution, so the final results could be verified against exact values. It can be shown that for the TM case (E - wave) the reflection coefficient is,

$$\Gamma_n^{TE} = \frac{\left[J_n(k_r R_a)J'_n(R_a) - \sqrt{\frac{\varepsilon_r}{\mu_r}}\, J'_n(k_r R_a)J_n(R_a)\right]}{\left[J_n(k_r R_a)H_n^{(2)'}(R_a) - \sqrt{\frac{\varepsilon_r}{\mu_r}}\, J'_n(k_r R_a)H_n^{(2)}(R_a)\right]}$$

and that for the TE case (H - wave) the refection coefficient is,

$$\Gamma_n^{TM} = \frac{\left[J_n(k_r R_a)J'_n(R_a) - \sqrt{\frac{\mu_r}{\varepsilon_r}}\, J'_n(k_r R_a)J_n(R_a)\right]}{\left[J_n(k_r R_a)H_n^{(2)'}(R_a) - \sqrt{\frac{\mu_r}{\varepsilon_r}}\, J'_n(k_r R_a)H_n^{(2)}(R_a)\right]}$$

where $k_r = \sqrt{\mu_r \varepsilon_r}$, $y_r = \sqrt{\frac{\varepsilon_r}{\mu_r}}$ and $z_r = \sqrt{\frac{\mu_r}{\varepsilon_r}}$. [Ref. 9] The scattered field for the TM case is,

$$E_Z^s(R,\phi) = -\,E^{\text{incident}}\left[\Gamma_0 H_0^{(2)}(R) + 2\sum_{n=1}^{\infty} j^{-n}\Gamma_n H_n^{(2)}(R)\cos n\phi\right].$$

Similarly, the scattered field for the TE case is,

$$H_Z^s(R,\phi) = -\,H^{\text{incident}}\left[\Gamma_0 H_0^{(2)}(R) + 2\sum_{n=1}^{\infty} j^{-n}\Gamma_n H_n^{(2)}(R)\cos n\phi\right].$$

In the far field region, as R approaches infinity, the scattering width may be defined as,

$$\sigma(\phi) = 2\pi \lim_{\rho \to \infty} \rho \frac{|\overline{H}^s|^2}{|\overline{H}^{\text{incident}}|^2}$$

and

$$\sigma(\phi) = \frac{4}{k_0} |\Gamma_0 + 2\sum_{n=1}^{\infty} \Gamma_n \cos n\phi|^2.$$

The source code, without the Bessel function routines, for these calculations is provided as Appendix F.

Three different radius dielectric cylinders were tested. Wavenumber normalized radii of 0.5, 1.0 and 2.0 with ε_r = 2.56 + j0 were selected. The TE and TM results of the 0.5, 1.0 and 2.0 radii problems are provided as Figure 37 on page 62, Figure 38 on page 63, and Figure 39 on page 64 . To validate the Chapter VI conclusion that the unit normal was the cause of the errors for very small circular cylinders, the ε_r was increased from 2.56 to 25.6. This value still meets the requirement for mesh resolution not to exceed $\frac{\lambda}{20}$. The TE and TM results are provided as Figure 40 on page 65 . Actual cross section values are indicated by the squares and the Field Feedback Formulation solutions are plotted as a single solid curve.

C. HOMOGENEOUS IRREGULAR OBJECTS

The results detailed in [Ref. 10] were next validated with the F^3 program. This object, as seen in Figure 41 on page 66 , is a dielectric shell with inner radius = .25 λ_0, outer radius = .30 λ_0 and ε_r = 4 + j0. The comparison of the [Ref. 10] data and the F^3 results are provided as Figure 42 on page 67.

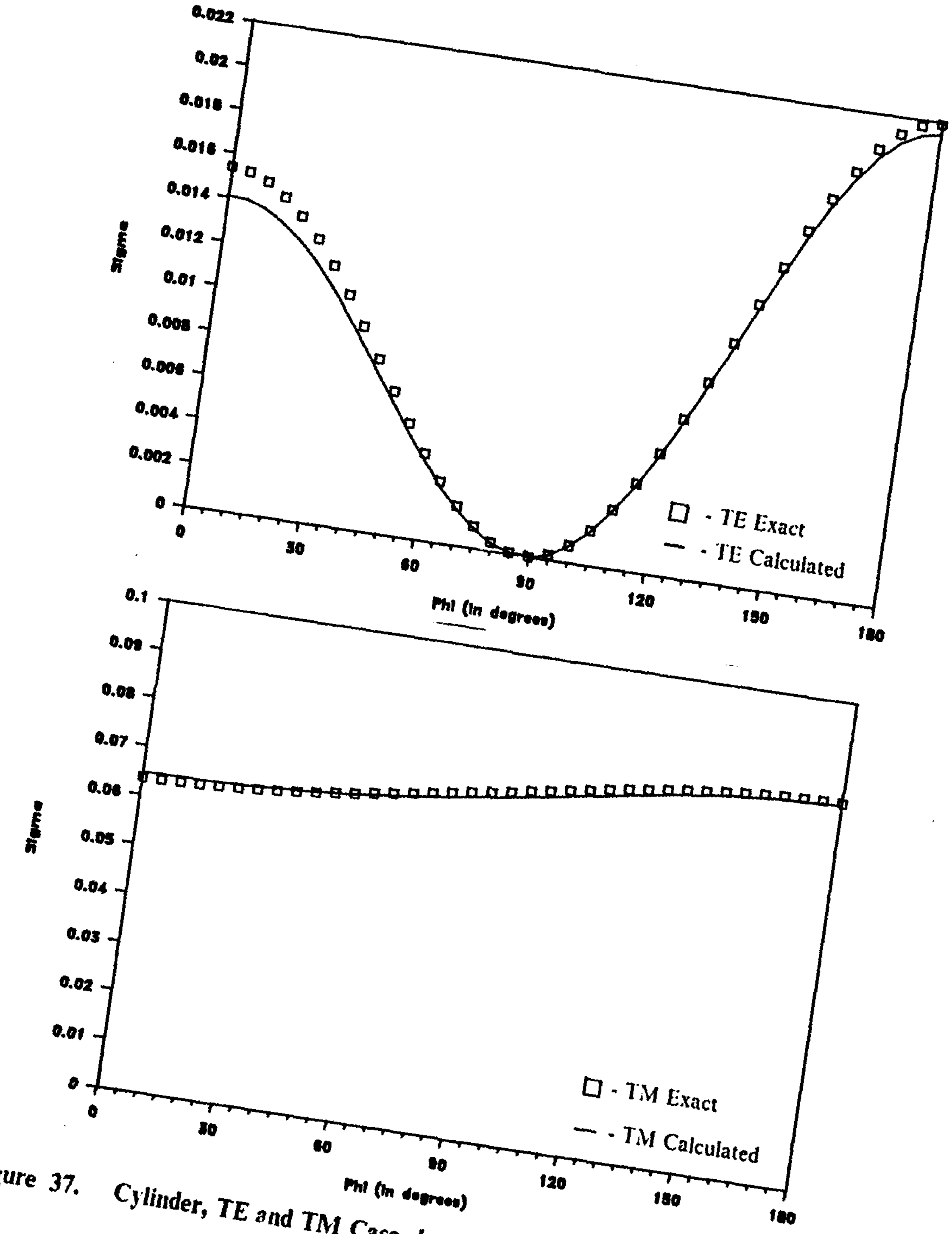

Figure 37. Cylinder, TE and TM Case, $k_0a = 0.5$, $\varepsilon_r = 2.56$

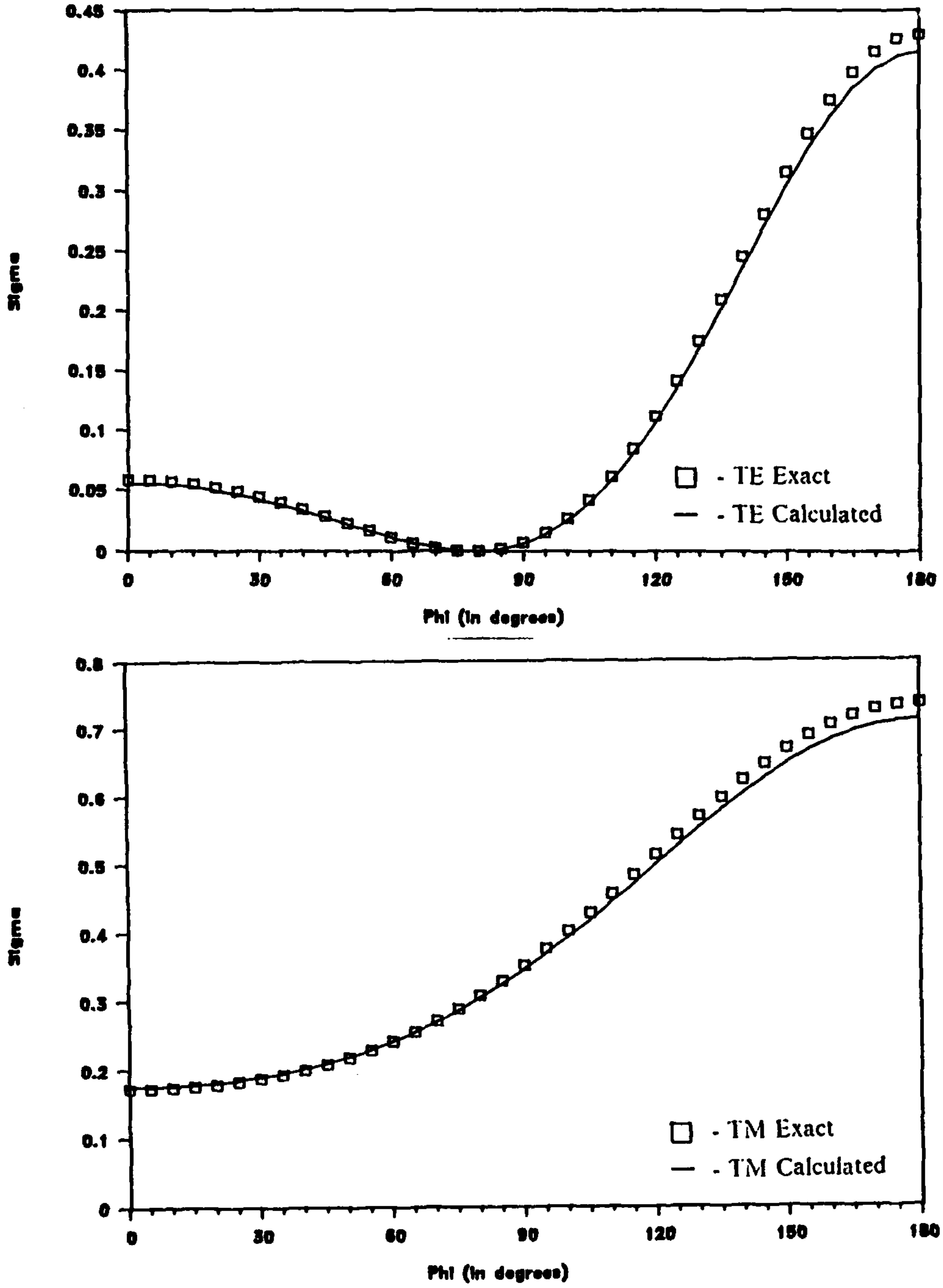

Figure 38. Cylinder, TE and TM Case, $k_0a = 1.0$, $\varepsilon_r = 2.56$

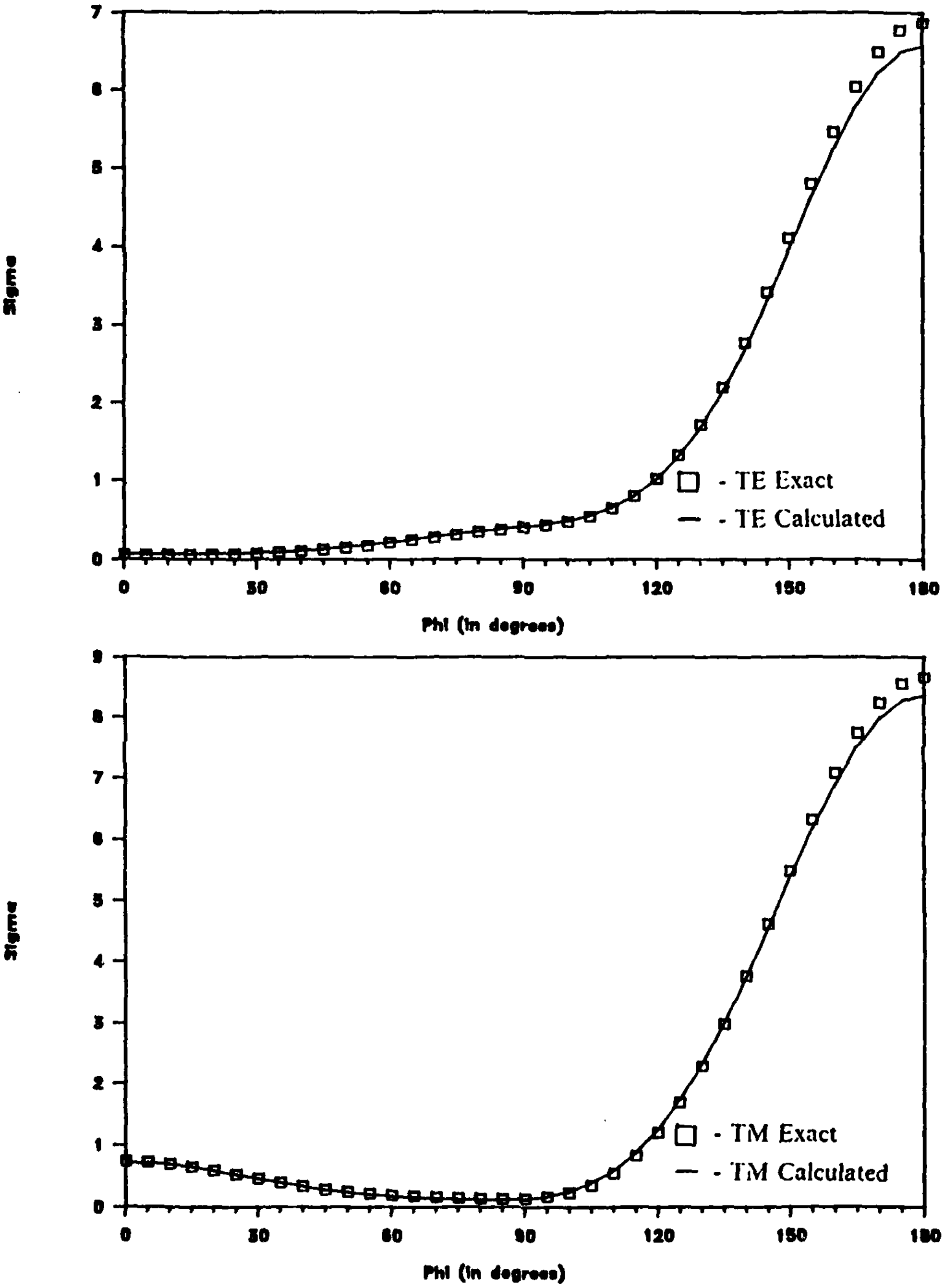

Figure 39. Cylinder, TE and TM Case, $k_0a = 2.0$, $\varepsilon_r = 2.56$

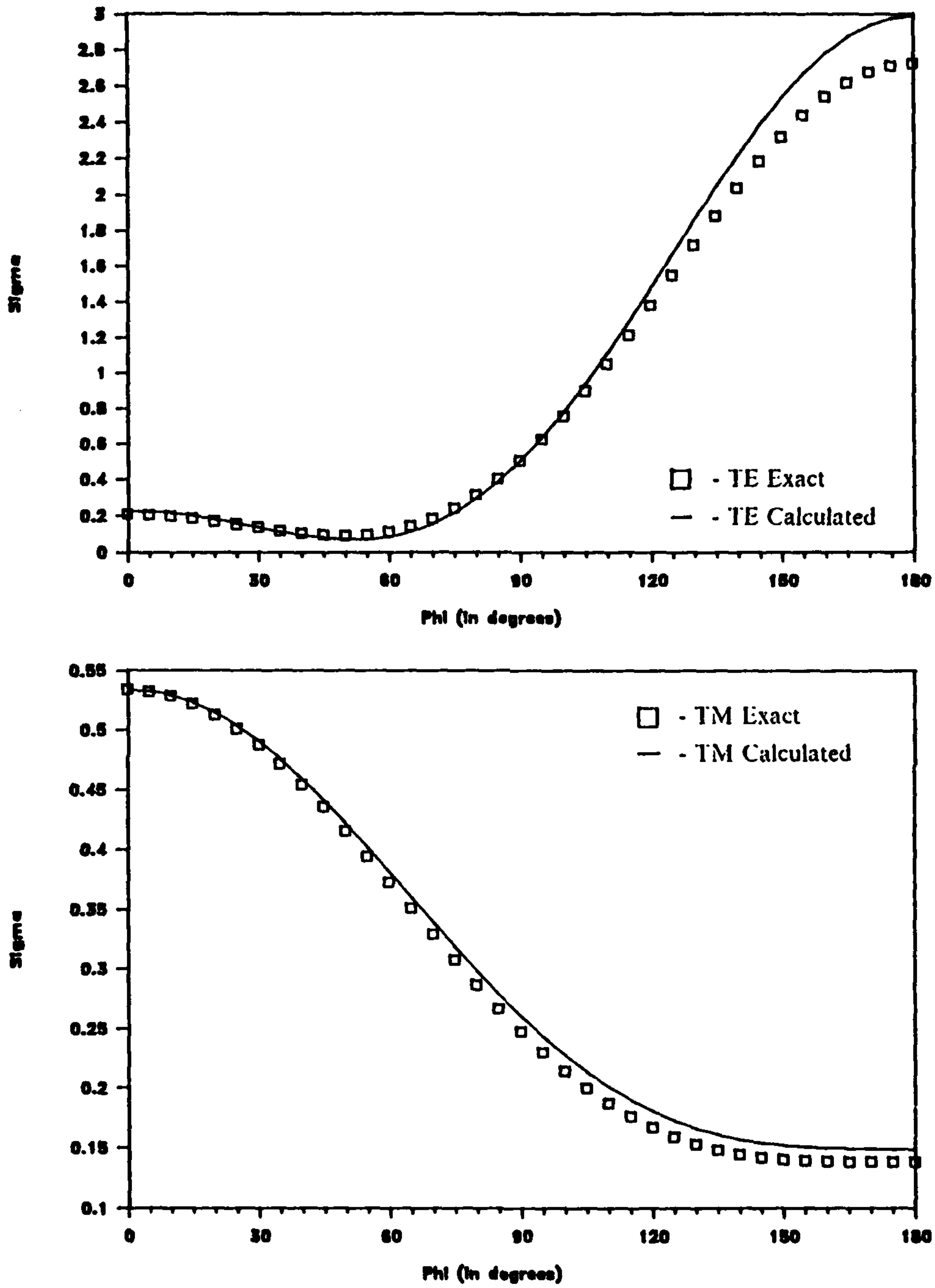

Figure 40. Cylinder, TE and TM Case, $k_0a = 0.5$, $\varepsilon_r = 25.6$

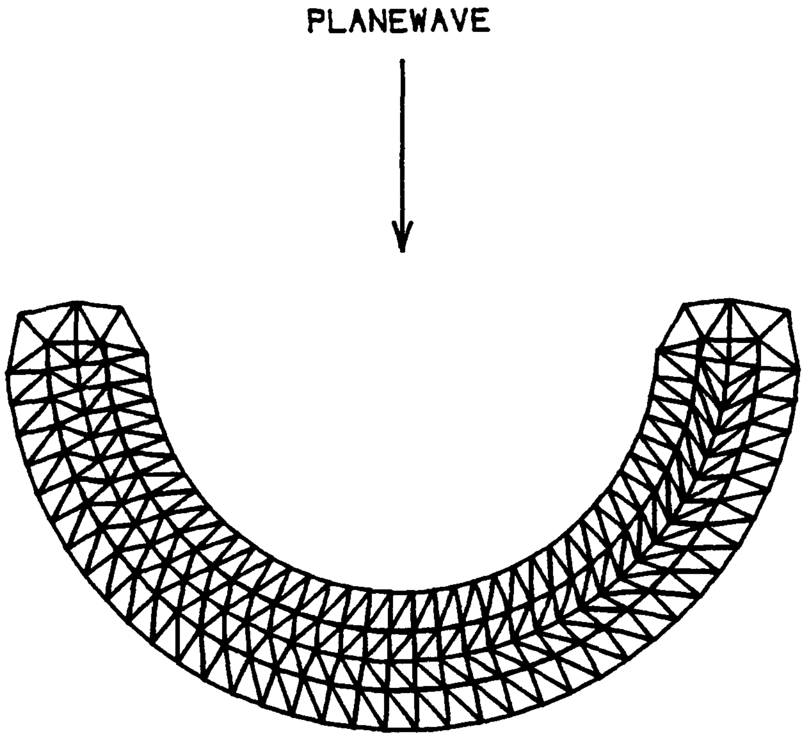

Figure 41. Dielectric Shell Mesh

The TE case shows excellent agreement. The TM case tends to diverge at the smaller values of ϕ.

D. AN INHOMOGENEOUS OBJECT

The results detailed in [Ref. 11] were next validated with the F^3 program. This object, as seen in Figure 43 on page 68, is a dielectric ring with inner radius = .25 λ_0, outer radius = .30 λ_0 and $\varepsilon_r = 4 + j0$. The exact solution to this problem is available. [Ref. 11]

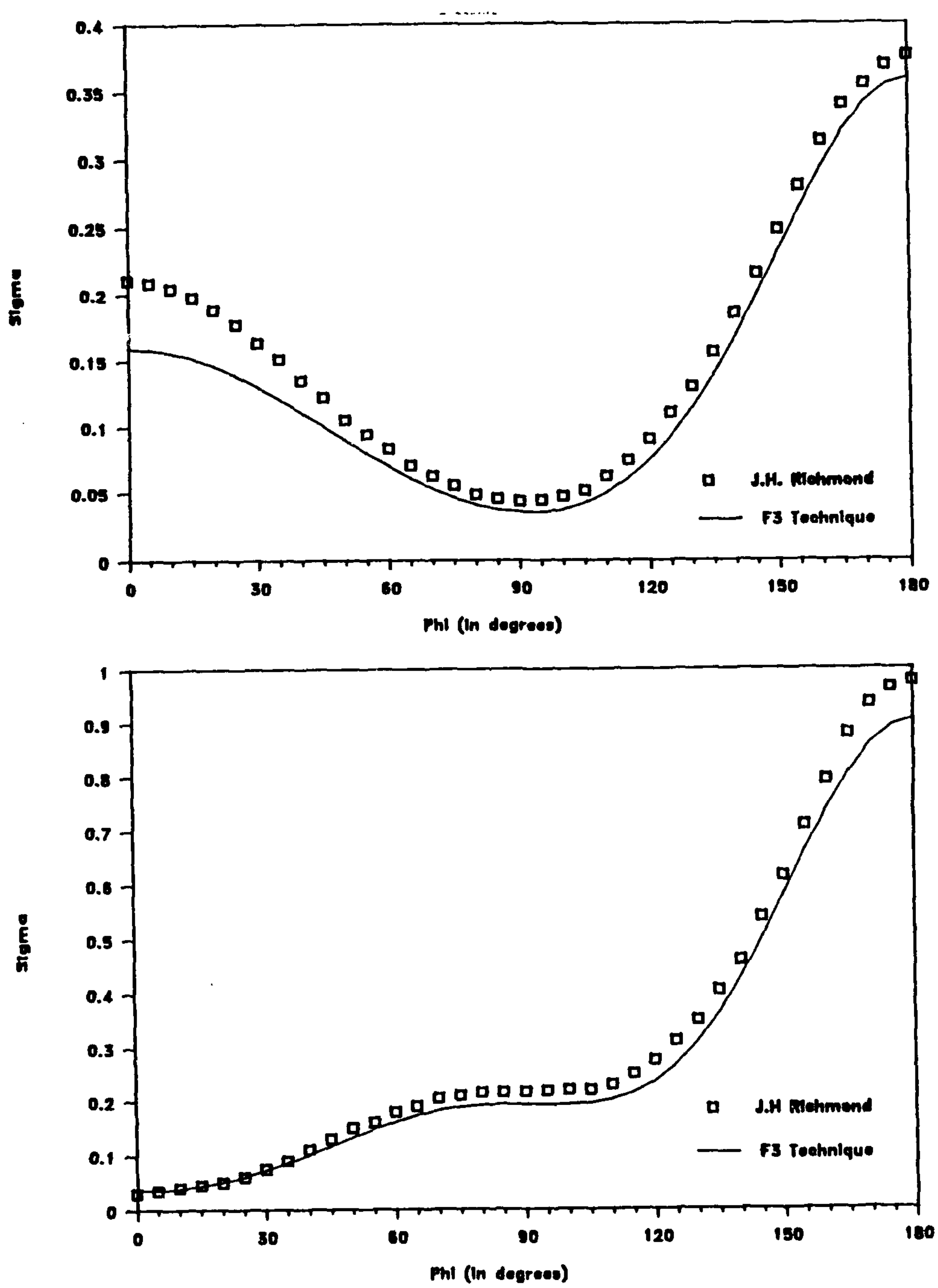

Figure 42. Dielectric Shell, TE and TM Case, $\varepsilon_r = 4 + j0$

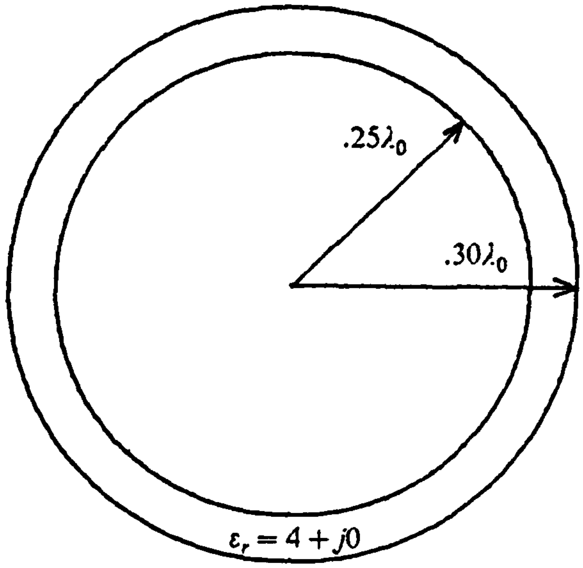

Figure 43. Dielectric Ring

The mesh generation program was not modified to conform to both the outer and inner material boundaries. The original mesh generation program design concept was to closely fit the mesh to the objects outer perimeter and then allow for material inhomogeneities to be accounted for by different material parameters being assigned to individual elements. A section of the generated mesh is shown in Figure 44 on page 69. Note the additional curved line showing the .25 λ_0 inner radius. This curve does not follow the established element boundaries. The effective ring is shown in Figure 45 on page 69. This resulted in the inner radius having an irregular pattern as elements near the material interface varied in material composition. This is similar to the granular noise problem characteristic of delta modulation communication channels. The TE and TM results are provided as Figure 46 on page 70.

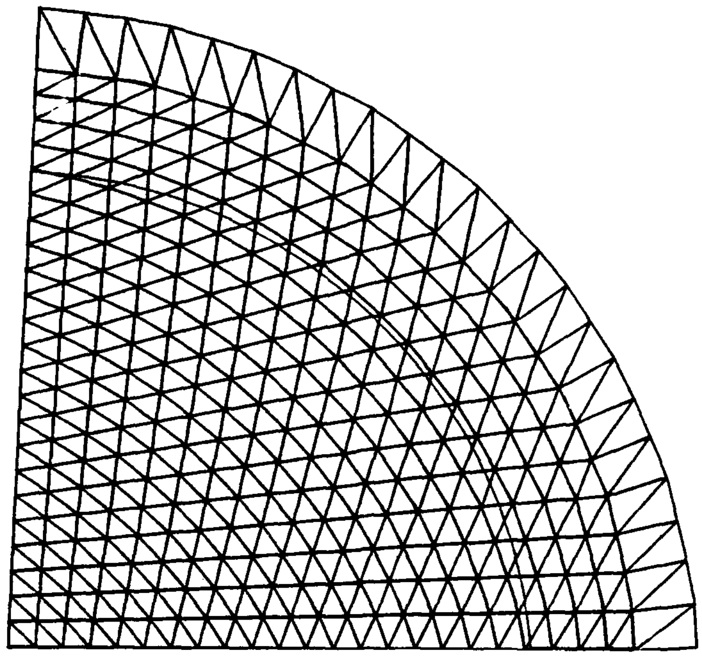

Figure 44. Partial Mesh with Inner Radius Curve

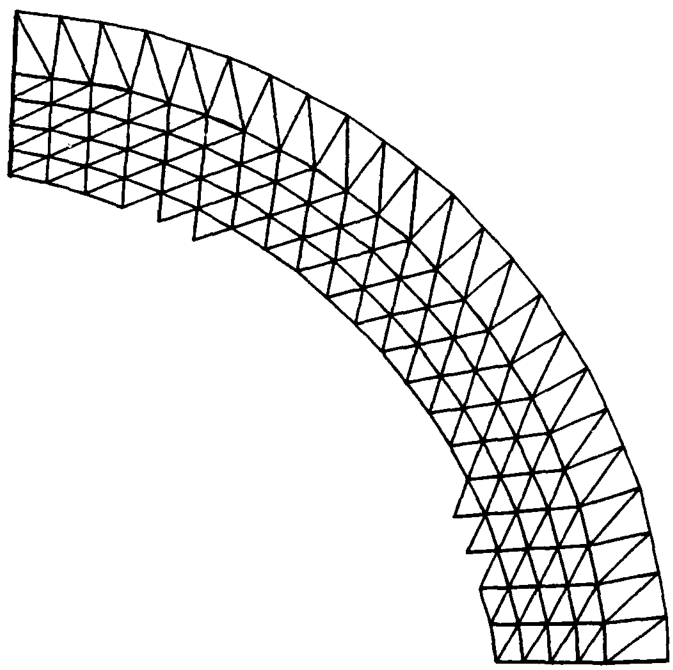

Figure 45. Effective Geometry for the Dielectric Mesh

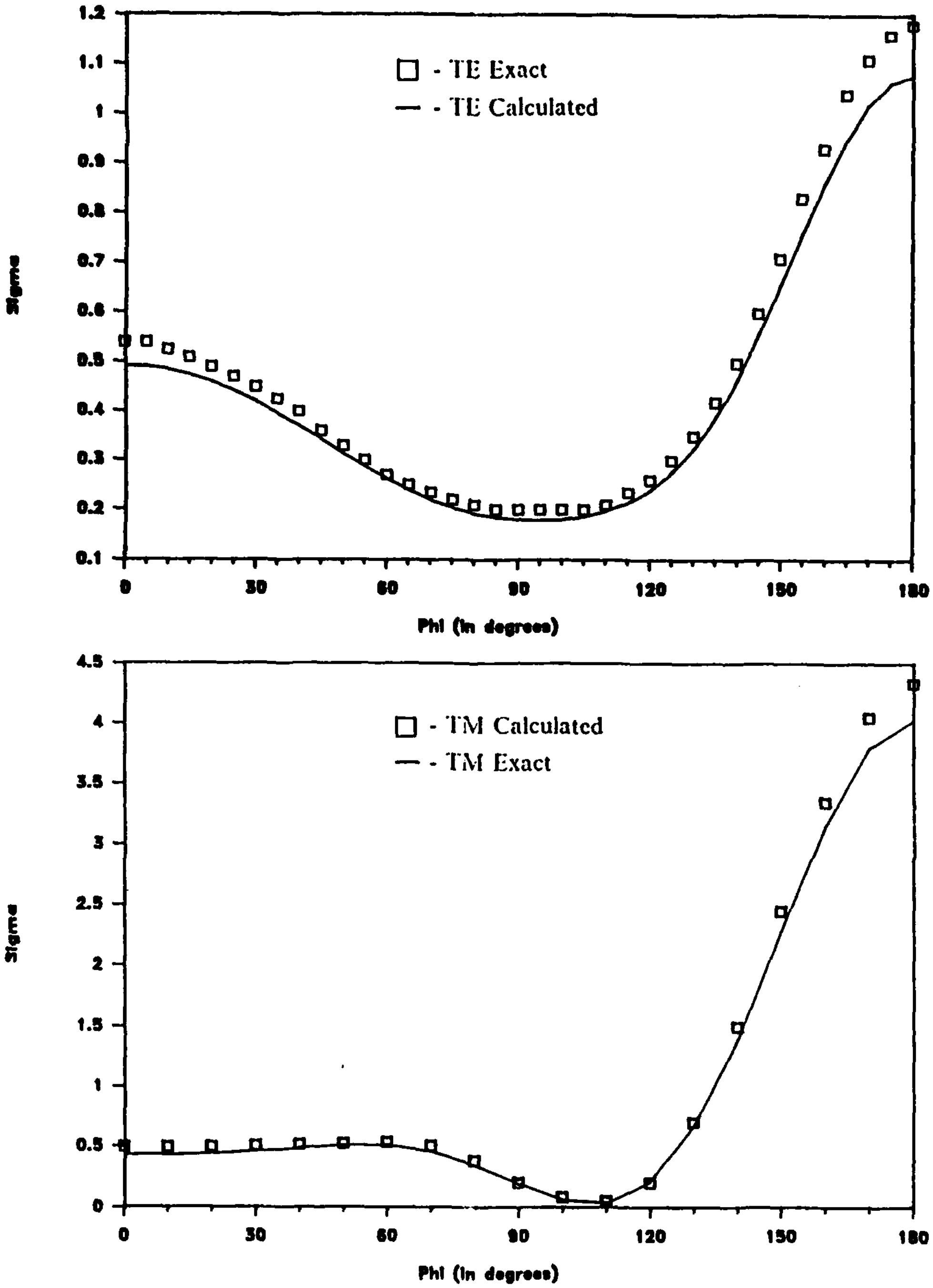

Figure 46. Dielectric Ring, TE and TM Case, $\varepsilon_r = 4 + j0$

VII. CONCLUSIONS

A. RESULTS

The Field Feedback Formulation proved to be an excellent tool for calculating the internal and scattered fields from the tested inhomogeneous asymmetric objects. The keys to satisfactory results are,

- Keep the maximum element dimension $\leq \frac{\lambda}{20}$,
- Maintain a mesh with a uniform distribution, and
- Maintain the offset distance near the mesh resolution in magnitude, but no greater than $\frac{\lambda_0}{20}$.

The techniques used proved to be capable of adapting to a wide variety of situations. These adaptations did not require program modifications or reprogramming. The numerous built-in features, as discussed in the input field description of Chapter III, allowed for this robustness.

B. RECOMMENDATIONS AND EXTENSIONS

With the basic program validated, future testing and development should,

- Emphasize large object validation against accepted results.
- Stress inhomogeneity and irregularity in all testing.
- Optimize the T matrix element calculation by improving the Green's function contour integral.
- Evaluate the usefulness of the maximum distance feature (Field 12). Beyond this maximum distance no contribution is made to the Green's function integral. It is not expected that this will be possible for objects less than a few wavelengths in maximum dimension.
- Modify the mesh generation program to allow for conformity to multiple interfaces (multi-layered objects without the granular noise problem).
- Modify all programs for Intel 80386 based Fortran compiler use, such as NDP FORTRAN by Microway. This will remove the 640 KB memory limitation imposed by the IBM DOS and allows for the solution to larger scattering problems.

APPENDIX A. INPUT DATA FILE EXAMPLE

```
 CIRCLE          - PLOT TITLE
R                - RECTANGULAR INPUT DATA
NI               - NO INTERMEDIATE DATA RECORDING
ND               - NO DISSPLA PROGRAM GENERATION
NU               - NON UNIFORM MATERIAL (MATERIAL INTERFACE PRESENT)
NM               - NO MESH GENERATION ONLY
0.019            - REQUESTED MESH RESOLUTION (IN WAVELENGTHS)
0.03             - CONTOUR DISTANCE (IN WAVELENGTHS)
1.0              - MULTIPLICATIVE GFI SCALE FACTOR
0                - DISTANCE < 1.0 BIAS TERM
0                - DISTANCE > 1.0 BIAS TERM
999.9            - MAX DIST BEYOND WHICH THERE IS NO CONTRIBUTION TO GFI
36               - NUMBER OF INPUT DATA POINTS
72               - NUMBER OF POINTS FOR SIGMA CALCULATION (IN THE CIRCLE)
1                - ELEMENT GENERATION METHOD
5                - START NODE
35               - STOP NODE
4.5              - X AXIS OFFSET
4.0              - Y AXIS OFFSET
0.3              - DESIRED DIMENSION (ORIGIN TO FIRST POINT ,WAVELENGTHS)
1.0              - REAL PART OF ALPHA
0.0              - IMAGINARY PART OF ALPHA
1.0              - REAL PART OF BETA
0.0              - IMAGINARY PART OF BETA
P                - PLANEWAVE GENERATOR ENABLED
1.0              - PLANEWAVE AMPLITUDE
3.00E+08         - INPUT FREQUENCY OF THE PLANEWAVE, FO
    0       .00000000     .50000000      - ANGLE, X, Y COORD.
   10       .08682409     .49240390
   20       .17101010     .46984630
   30       .25000000     .43301270
   40       .32139380     .38302220
   50       .38302220     .32139380
   60       .43301270     .25000000
   70       .46984630     .17101010
   80       .49240390     .08682407
   90       .50000000    -.00000002
  100       .49240390    -.08682411
  110       .46984630    -.17101010
  120       .43301270    -.25000000
  130       .38302220    -.32139380
  140       .32139380    -.38302220
  150       .25000000    -.43301270
  160       .17101000    -.46984630
  170       .08682404    -.49240390
  180      -.00000004    -.50000000
  190      -.08682413    -.49240390
  200      -.17101010    -.46984630
  210      -.25000000    -.43301270
  220      -.32139380    -.38302220
  230      -.38302220    -.32139380
```

240	-.43301270	-.25000000
250	-.46984630	-.17101000
260	-.49240390	-.08682403
270	-.50000000	.00000007
280	-.49240390	.08682416
290	-.46984630	.17101010
300	-.43301270	.25000010
310	-.38302220	.32139390
320	-.32139380	.38302230
330	-.24999990	.43301280
340	-.17101000	.46984630
350	-.08682401	.49240390

APPENDIX B. READ PROGRAM

```
C
      REAL A(0:2000), B(0:2000), MRES, DIST, XORIGIN, YORIGIN
      REAL C, D, E, F, DPER, DD, FREQ, E0, MAXD
      INTEGER I, J, METHOD, STARTND, STOPND, NRES, MODE, LBIAS, GBIAS
      CHARACTER*1 CHAR,CHAR1,CHAR2,CHAR3,CHAR4,CHAR5
      CHARACTER*13 NAME
C
      OPEN(UNIT = 1, FILE = 'D: FINALDWG.DAT')
      OPEN(UNIT = 2, FILE = 'C: MSFORT INPUT.DAT')
      J = 0
C
      WRITE(*,*) 'ENTER THE PLOT NAME OR LABEL (MAX OF 13 CHARACTERS)'
      READ(*,1005) NAME
      WRITE(*,*) 'ENTER THE COORDINATE SYSTEM IN USE. (R OR P)'
      READ(*,1000) CHAR
      WRITE(*,*) 'DO YOU WANT INTERMEDIATE VALUES ? (I)'
      READ(*,1000) CHAR1
      WRITE(*,*) 'DO YOU WANT A DISSPLA PROGRAM GENERATED ? (D)'
      READ(*,1000) CHAR2
      WRITE(*,*) 'UNIFORM SLAB (NO MATERIAL TO VACUUM INTERFACE ? (U)'
      READ(*,1000) CHAR3
      WRITE(*,*) 'MESH GENERATION ONLY ? (M)'
      READ(*,1000) CHAR4
      WRITE(*,*) 'ENTER THE MESH RESOLUTION. (0.4, ETC...)'
      READ(*,*) MRES
      WRITE(*,*) 'ENTER THE CONTOUR DISTANCE. (0.3, ETC...)'
      READ(*,*) DIST
      WRITE(*,*) 'ENTER THE GFI SCALE FACTOR.(1.0, ETC...)'
      READ(*,*) DPER
      WRITE(*,*) 'ENTER THE GFI ( < 1 BIAS TERM (0,1, ETC...)'
      READ(*,*) LBIAS
      WRITE(*,*) 'ENTER THE GFI ( > 1 BIAS TERM (0,1, ETC...)'
      READ(*,*) GBIAS
      WRITE(*,*) 'ENTER THE GFI MAX DISTANCE (999.0, ETC...)'
      READ(*,*) MAXD
      WRITE(*,*) 'ENTER THE NUMBER OF POINTS FOR SIGMA CALC.(36,ETC...)'
      READ(*,*) NRES
      WRITE(*,*) 'ENTER THE DRAWING METHOD. (1 - 6)'
      READ(*,*) METHOD
      WRITE(*,*) 'ENTER THE STARTING NODE NUMBER.(MUST BE > 0)'
      READ(*,*) STARTND
      WRITE(*,*) 'ENTER THE BISECTION STOPPING NODE NUMBER. '
      READ(*,*) STOPND
      WRITE(*,*) 'DESIRED DISTANCE FROM ORIGIN TO POINT 1 (IN WLs) '
      READ(*,*) DD
      WRITE(*,*) 'ENTER THE X AXIS ORIGIN.'
      READ(*,*) XORIGIN
      WRITE(*,*) 'ENTER THE Y AXIS ORIGIN.'
      READ(*,*) YORIGIN
      WRITE(*,*) 'ENTER THE REAL COMPONENT OF ALPHA.'
      READ(*,*) C
```

```
      WRITE(*,*) 'ENTER THE IMAGINARY COMPONENT OF ALPHA.'
      READ(*,*) D
      WRITE(*,*) 'ENTER THE REAL COMPONENT OF BETA.'
      READ(*,*) E
      WRITE(*,*) 'ENTER THE IMAGINARY COMPONENT OF BETA.'
      READ(*,*) F
      WRITE(*,*) 'PLANE WAVE (P) OR CYLINDRICAL MODE (C) '
      READ(*,1000) CHAR5
      WRITE(*,*) 'ENTER THE WAVE FREQUENCY (IN HERTZ)'
      READ(*,*) FREQ
      WRITE(*,*) 'ENTER THE WAVE AMPLITUDE '
      READ(*,*) E0
      IF((CHAR5.EQ.'C').OR.(CHAR5.EQ.'c')) THEN
           WRITE(*,*) 'ENTER THE MODE NUMBER '
           READ(*,*) MODE
      ENDIF
      DO 10, I = 0, 1999
           READ(1,*,ERR = 20) A(J), B(J)
           IF(A(J).GT.1000) THEN
                GOTO 5
           ELSEIF(B(J).GT.1000) THEN
                GOTO 5
           ELSE
                J = J + 1
           ENDIF
5     CONTINUE
10    CONTINUE
20    J = J - 1
      WRITE(2,1005) NAME
      WRITE(2,1000) CHAR
      WRITE(2,1000) CHAR1
      WRITE(2,1000) CHAR2
      WRITE(2,1000) CHAR3
      WRITE(2,1000) CHAR4
      WRITE(2,1010) MRES
      WRITE(2,1010) DIST
      WRITE(2,1010) DPER
      WRITE(2,1020) LBIAS
      WRITE(2,1020) GBIAS
      WRITE(2,1040) MAXD
      WRITE(2,1020) J
      WRITE(2,1020) NRES
      WRITE(2,1020) METHOD
      WRITE(2,1020) STARTND
      WRITE(2,1020) STOPND
      WRITE(2,1010) XORIGIN
      WRITE(2,1010) YORIGIN
      WRITE(2,1010) DD
      WRITE(2,1010) C
      WRITE(2,1010) D
      WRITE(2,1010) E
      WRITE(2,1010) F
      WRITE(2,1000) CHAR5
      WRITE(2,1010) E0
      IF((CHAR5.EQ.'C').OR.(CHAR5.EQ.'c')) THEN
           WRITE(2,1020) MODE
```

```
      ENDIF
      WRITE(2,*) FREQ
      DO 30, I = 1, J
           WRITE(2,1030) I, A(I), B(I)
30    CONTINUE
      WRITE(2,*) 'END OF FILE'
C
1000  FORMAT(A1)
1005  FORMAT(A13)
1010  FORMAT(F8.6)
1020  FORMAT(I3)
1030  FORMAT(1X,I3,7X,F12.8,5X,F12.8)
1040  FORMAT(F8.3)
C
      CLOSE(1)
      CLOSE(2)
      STOP
      END
C
```

APPENDIX C. MESH GENERATION/FINITE ELEMENT PROGRAM

```
C
C     FINITE ELEMENT MESH GENERATION PROGRAM
C     WRITTEN BY LT. T.B. WELCH
C
C     w/ PROGRAMMING IDEAS FROM PROF. M.A. MORGAN
C
C     COMPLETELY FILE (INPUT.DAT) DRIVEN
C
C     INPUT PARAMETERS:
C
C             CHARACTER (POLAR OR RECTANGULAR INPUT DATA - FLAG)
C             CHARACTER (INTERMEDIATE MATRICES WRITE - FLAG)
C             CHARACTER (DISSPLA PROGRAM GENERATION - FLAG)
C             CHARACTER (UNIFORM MATERIAL - FLAG)
C             CHARACTER (MESH GENERATION ONLY - FLAG)
C             MESH RESOLUTION
C             CONTOUR OFFSET
C             DESIRED SCALE FACTOR FOR GREEN'S FUNCTION INTEGRAL
C             BIAS (SHIFT) FOR GFI STEP WHEN < 1.0
C             BIAS (SHIFT) FOR GFI STEP WHEN > 1.0
C             MAXIMUM DISTANCE BEYOND WHICH THERE WILL BE NO
C                  CONTRIBUTION TO THE GREEN'S FUNCTION INTEGRAL
C             NUMBER OF POINTS
C             NUMBER OF POINTS FOR CROSS SECTION, (IN THE CIRCLE)
C             MESH GENERATION TECHNIQUE
C             START NODE
C             STOP NODE
C             X ORIGIN
C             Y ORIGIN
C             DESIRED DISTANCE (FROM ORIGIN TO FIRST POINT, WAVELENGTH)
C             ALPHA (INPUT AS A AND B THEN CONVERTED TO COMPLEX ALPHA)
C             BETA  (INPUT AS A AND B THEN CONVERTED TO COMPLEX BETA)
C             CHARACTER (PLANEWAVE GENERATOR - FLAG)
C                  AMPLITUDE
C                  FREQUENCY (IN HERTZ)
C             -- OR --
C                  AMPLITUDE
C                  CYLINDRICAL MODE NUMBER (GENERATOR ALWAYS DOES N=1)
C                  FREQUENCY (IN HERTZ)
C             -- OR --
C                  BOUNDARY CONDITIONS
C             DATA POINT PAIRS (X, Y OR RADIUS, THETA)
C             END OF FILE MARKER
C
C
      COMPLEX ALPHA,BETA,BCOND(100),LINE(50),ANS(100)
      COMPLEX A(50,50),B(50,50),C(50,50),P(50,100),SURBC(100)
      COMPLEX U(100,100),PSI(50,100),SVEC(50,100)
      REAL MRES,MESH(0:200,5),NDP(200,2),OFFSET,DPER,MRESW
      REAL MINAREA,MAXAREA,AREA,E0,XORIGIN,YORIGIN,K0,MAXD
      INTEGER NPNTS,PERND,NODES(200),MOR(200),BIND,NBMX,UNK,LBIAS,GBIAS
```

```
      INTEGER LND(0:200,3),NDL(200,4),NCT(200),MAXEL,IMX,MODE,NRES
      INTEGER METHOD,STARTND,STOPND,MINROW,MINEL,MAXROW,NABC(100,3)
      CHARACTER*1 CHAR, CHAR1, CHAR2, CHAR3, CHAR4, CHAR5
      EQUIVALENCE (PSI, P)
C
      COMMON/BLK1/MESH
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK4/LND,NDL,NCT
      COMMON/BLK5/NDP
      COMMON/BLK6/MOR
      COMMON/BLK7/MINAREA,MINROW,MINEL,MAXAREA,MAXROW,MAXEL,AREA
      COMMON/BLK8/A,B,C,P
      COMMON/BLK9/CHAR2, CHAR3, CHAR4
C
      OPEN (UNIT =  1, FILE = 'MATRIX.DAT',STATUS = 'UNKNOWN')
      OPEN (UNIT =  2, FILE = 'PLT.DAT',STATUS = 'UNKNOWN')
      OPEN (UNIT =  3, FILE = 'INPUT.DAT',STATUS = 'OLD')
      OPEN (UNIT =  4, FILE = 'TEXT.LBL',STATUS = 'UNKNOWN')
      OPEN (UNIT =  7, FILE = 'RS.DAT',STATUS = 'UNKNOWN',
     CACCESS='SEQUENTIAL',FORM='UNFORMATTED')
      OPEN (UNIT =  8, FILE = 'PSI.DAT',STATUS = 'UNKNOWN',
     CACCESS='SEQUENTIAL',FORM='FORMATTED')
      OPEN (UNIT =  9, FILE = 'ABCP.DAT',STATUS = 'UNKNOWN')
      OPEN (UNIT = 10, FILE = 'BCOND.DAT', STATUS = 'UNKNOWN')
      OPEN (UNIT = 11, FILE = 'FMATRIX.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 12, FILE = 'NABC.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 13, FILE = 'OUTPUT.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 19, FILE = 'ABRMAT.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 20, FILE = 'DISPLA.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 30, FILE = 'U.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 40, FILE = 'F3.DAT',STATUS='UNKNOWN')
C
      MINAREA = 999999.9
      MAXAREA = -999999.9
C
      CALL IO(NPNTS,MRES,METHOD,STARTND,STOPND,OFFSET,ALPHA,BETA,BCOND,
     CEO,CHAR1,MODE,XORIGIN,YORIGIN,CHAR5,DPER,KO,NRES,MRESW,LBIAS,GBIAS
     C,MAXD)
      CALL ROTATE(NPNTS,METHOD,STARTND,STOPND)
      CALL BOUND(MRES,NPNTS,METHOD,STOPND)
      CALL NORMAL
      CALL NODSET(METHOD,NABC,NBMX,UNK)
      PAUSE 'PLEASE PRESS ENTER TO CONTINUE, OR CTRL BREAK TO ABORT!'
      CALL LOADER(BCOND,OFFSET,ALPHA,BETA,NABC,IMX,NBMX,EO,SURBC,CHAR1,
     CLINE,MODE,XORIGIN,YORIGIN,SVEC,CHAR5)
      CALL SWEEP(IMX,NABC,SURBC,LINE,CHAR1,U,BCOND,PSI,ANS,CHAR5)
      CALL SAVE(BCOND,ANS,U,OFFSET,PERND,CHAR5,DPER,KO,XORIGIN,YORIGIN,
     CNRES,MRESW,LBIAS,GBIAS,MAXD)
C
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      CLOSE(7)
      CLOSE(8)
```

```
      CLOSE(9)
      CLOSE(10)
      CLOSE(11)
      CLOSE(12)
      CLOSE(13)
      CLOSE(19)
      CLOSE(20)
      CLOSE(30)
      CLOSE(40)
C
      STOP
      END
C
      SUBROUTINE IO(NPNTS,MRES,METHOD,STARTND,STOPND,DIST,ALPHA,BETA,
     CBCOND,E0,CHAR1,MODE,XORIGIN,YORIGIN,CHAR5,DPER,K0,NRES,MRESW,
     CLBIAS,GBIAS,MAXD)
C
C     THIS SUBROUTINE READS IN THE INPUT PARAMETERS
C     AND SURFACE DATA POINTS. THESE POINTS CAN BE
C     IN EITHER POLAR OR RECTANGULAR FORM.
C
      COMPLEX ALPHA,BETA,BCOND(100)
      REAL A,B,PI,XORIGIN,YORIGIN,E0,K0,F0,SFAC,DD
      REAL MESH(0:200,5),MRES,THETA,RADIUS,DIST,C,LAMBDA,MRESW
      REAL DISTW,DPER,MAXD
      INTEGER I,II,J,K,NPNTS,RES,METHOD,STARTND,STOPND,MODE,NRES,LBIAS
      INTEGER GBIAS
      CHARACTER*1 CHAR,CHAR1,CHAR2,CHAR3,CHAR4,CHAR5
      CHARACTER*12 NAME
C
      COMMON/BLK1/MESH
      COMMON/BLK9/CHAR2, CHAR3, CHAR4
C
      C = 2.997925E+08
      PI = 4.0*ATAN(1.0)
      READ(3,1070) NAME
      READ(3,1000) CHAR
      READ(3,1000) CHAR2
      READ(3,1000) CHAR3
      READ(3,1000) CHAR4
      READ(3,1000) CHAR5
      READ(3,*) MRESW
      READ(3,*) DISTW
      READ(3,*) DPER
      READ(3,*) LBIAS
      READ(3,*) GBIAS
      READ(3,*) MAXD
      READ(3,*) RES
      READ(3,*) NRES
      READ(3,*) METHOD
      READ(3,*) STARTND
      READ(3,*) STOPND
      READ(3,*) XORIGIN
      READ(3,*) YORIGIN
      READ(3,*) DD
      READ(3,*) A
```

```
      READ(3,*) B
      ALPHA = CMPLX(A,B)
      READ(3,*) A
      READ(3,*) B
      BETA = CMPLX(A,B)
      READ(3,1000) CHAR1
C
      IF(METHOD.EQ.1) THEN
           WRITE(6,*) 'METHOD 1 SELECTED - OBJECT BISECTION '
      ELSEIF(METHOD.EQ.2) THEN
           WRITE(6,*) 'METHOD 2 SELECTED - CONNECTED NODES '
      ELSEIF(METHOD.EQ.3) THEN
           WRITE(6,*) 'METHOD 3 SELECTED - SELECTED STOP NODE '
      ELSEIF(METHOD.EQ.4) THEN
           WRITE(6,*) 'METHOD 4 SELECTED - CONNECTED/SELECTED STOP NODE'
      ELSEIF(METHOD.EQ.5) THEN
           WRITE(6,*) 'METHOD 5 SELECTED - EQUALLY SPACED CONNECTD NODE'
      ELSE
           WRITE(6,*) 'METHOD 6 SELECTED - SELECTED START AND STOP NODE'
      ENDIF
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
           WRITE(6,*) 'INTERMEDIATE MATRIX FILE GENERATION   - ENABLED'
      ELSE
           WRITE(6,*) 'INTERMEDIATE MATRIX FILE GENERATION   - DISABLED'
      ENDIF
C
      IF((CHAR3.EQ.'D').OR.(CHAR3.EQ.'d')) THEN
           WRITE(6,*) 'DISSPLA FORTRAN  PROGRAM GENERATION   - ENABLED'
      ELSE
           WRITE(6,*) 'DISSPLA FORTRAN  PROGRAM GENERATION   - DISABLED'
      ENDIF
C
      IF((CHAR4.EQ.'U').OR.(CHAR4.EQ.'u')) THEN
           WRITE(6,*) 'UNIFORM MATERIAL SPECIFIED (NO INTERFACE) '
      ELSE
           WRITE(6,*) 'MATERIAL SPECIFIED WITH A VACUUM AROUND OBJECT'
      ENDIF
C
      IF((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m')) THEN
           WRITE(6,*) 'MESH GENERATION <<< ONLY >>>           - ENABLED'
      ELSE
           WRITE(6,*) 'MESH GENERATION AND FE PROGRAM         - ENABLED'
      ENDIF
C
      IF((CHAR1.EQ.'P').OR.(CHAR1.EQ.'p')) THEN
           READ(3,*) E0
           READ(3,*) F0
           LAMBDA = C/F0
           K0 = 2*PI/LAMBDA
           WRITE(6,*) 'PLANEWAVE BOUNDARY VALUE GENERATION   - ENABLED'
           WRITE(6,*) 'AMPLITUDE(E0) = ', E0,', WAVENUMBER(K0) = ',K0
           WRITE(6,*)'WAVELENGTH    = ',LAMBDA,', FREQUENCY(F0)  = ',F0
      ELSEIF((CHAR1.EQ.'C').OR.(CHAR1.EQ.'c')) THEN
           READ(3,*) E0
           READ(3,*) MODE
```

```
          READ(3,*) FO
          LAMBDA = C/FO
          KO = 2*PI*LAMBDA
          WRITE(6,*) 'CLYINRICAL BOUNDARY VALUE GENERATION  - ENABLED'
          WRITE(6,*) 'AMPLITUDE(EO) = ', EO,', MODE NUMBER = ', MODE
          WRITE(6,*)'WAVELENGTH    = ',LAMBDA,', FREQUENCY(FO)  = ',FO
      ELSE
          WRITE(6,*) 'ALL BOUNDARY VALUE GENERATION METHODS - DISABLED'
          DO 5, K = 1, RES
               READ(3,*) BCOND(K)
5         CONTINUE
      ENDIF
C
C     POLAR COORDINATE INPUT ROUTINE
C
      IF((CHAR.EQ.'P').OR.(CHAR.EQ.'p')) THEN
          READ(3,1020) THETA, RADIUS
          SFAC = 2*PI*DD/RADIUS
          MESH(0,4) = (RADIUS*SIN(THETA*PI/180.0))*SFAC + XORIGIN
          MESH(0,5) = (RADIUS*COS(THETA*PI/180.0))*SFAC + YORIGIN
          DO 10, J = 1, RES-1
               READ(3,1020) THETA, RADIUS
               MESH(J,4) = (RADIUS*SIN(THETA*PI/180.0))*SFAC + XORIGIN
               MESH(J,5) = (RADIUS*COS(THETA*PI/180.0))*SFAC + YORIGIN
10        CONTINUE
          WRITE(6,*) 'POLAR COORDINATE INPUT SELECTED '
C
C     RECTANGULAR COORDINATE INPUT ROUTINE
C
      ELSEIF((CHAR.EQ.'R').OR.(CHAR.EQ.'r')) THEN
               READ(3,1010) MESH(0,4), MESH(0,5)
               SFAC = 2*PI*DD/((MESH(0,4)**2+MESH(0,5)**2)**0.5)
               MESH(0,4) = MESH(0,4)*SFAC + XORIGIN
               MESH(0,5) = MESH(0,5)*SFAC + YORIGIN
          DO 20, J = 1, RES-1
               READ(3,1010) MESH(J,4), MESH(J,5)
               MESH(J,4) = MESH(J,4)*SFAC + XORIGIN
               MESH(J,5) = MESH(J,5)*SFAC + YORIGIN
20        CONTINUE
          WRITE(6,*) 'RECTANGULAR COORDINATE INPUT SELECTED '
      ELSE
          WRITE(6,*) 'INPUT DATA FILE COORDINATE SPECIFICATION ERROR'
      ENDIF
      WRITE(*,1100) DD, SFAC
      MESH(RES,4) = MESH(0,4)
      MESH(RES,5) = MESH(0,5)
      NPNTS = J
      WRITE(6,*) 'NODE AT 0 DEGREES (X/Y COORDINATES) = ', MESH(0,4),
     CMESH(0,5)
      WRITE(6,*) 'X, Y OFFSETS = ', XORIGIN, YORIGIN
      WRITE(6,1090) ALPHA, BETA
      MRES = MRESW*2*PI
      WRITE(6,*) 'MESH RESOLUTION      = ',MRESW,' WAVELENGTHS'
      DIST = DISTW*2*PI
      WRITE(6,*) 'CONTOUR DISTANCE     = ',DISTW,' WAVELENGTHS'
      WRITE(6,*) 'NUMBER OF DATA POINTS = ',RES
```

```
      IF((METHOD.EQ.3).OR.(METHOD.EQ.6)) THEN
           WRITE(6,*) 'START NODE              = ',STARTND
           WRITE(6,*) 'STOP  NODE              = ',STOPND
      ENDIF
C
      WRITE(4,*) '17'
      WRITE(4,*) '10            1             0             130
     C    55           1'
      WRITE(4,*) '2             1             0             365
     C    80           1'
      WRITE(4,*) '2             1             0             365
     C    90           2'
      WRITE(4,*) '2             1             0             365
     C    100          1'
      WRITE(4,*) '2             1             0             365
     C    110          2'
      WRITE(4,*) '2             1             0             365
     C    120          1'
      WRITE(4,*) '2             1             0             365
     C    130          2'
      WRITE(4,*) '2             1             0             365
     C    140          1'
      WRITE(4,*) '2             1             0             365
     C    150          2'
      WRITE(4,*) '2             1             0             365
     C    160          1'
      WRITE(4,*) '2             1             0             365
     C    170          2'
      WRITE(4,*) '2             1             0             365
     C    180          1'
      WRITE(4,*) '2             1             0             365
     C    190          2'
      WRITE(4,*) '2             1             0             365
     C    200          1'
      WRITE(4,*) '2             1             0             365
     C    210          2'
      WRITE(4,*) '2             1             0             365
     C    220          1'
      WRITE(4,*) '2             1             0             365
     C    230          2'
      WRITE(4,1070) NAME
      WRITE(4,*) 'L'
      WRITE(4,*) 'Mesh Resolution'
      WRITE(4,*) 'L'
      WRITE(4,1030) MRESW
      WRITE(4,*) 'L'
      WRITE(4,*) 'Contour Distance'
      WRITE(4,*) 'L'
      WRITE(4,1030) DISTW
      WRITE(4,*) 'L'
      WRITE(4,*) 'Number of Points'
      WRITE(4,*) 'L'
      WRITE(4,1040) RES
      WRITE(4,*) 'L'
      WRITE(4,*) 'Method'
      WRITE(4,*) 'L'
```

```
      WRITE(4,1040) Method
      WRITE(4,*) 'L'
      WRITE(4,*) 'X Origin'
      WRITE(4,*) 'L'
      WRITE(4,1050) XORIGIN
      WRITE(4,*) 'L'
      WRITE(4,*) 'Y Origin'
      WRITE(4,*) 'L'
      WRITE(4,1050) YORIGIN
      WRITE(4,*) 'L'
      WRITE(4,*) 'Alpha'
      WRITE(4,*) 'L'
      WRITE(4,1060) ALPHA
      WRITE(4,*) 'L'
      WRITE(4,*) 'Beta'
      WRITE(4,*) 'L'
      WRITE(4,1060) BETA
      WRITE(4,*) 'L'
      WRITE(4,*) '9'
C
      DO 30, II = 0, NPNTS-1
           WRITE(2,*) MESH(II,4), MESH(II,5)
30    CONTINUE
      WRITE(2,*) MESH(0,4), MESH(0,5)
      WRITE(2,*) '999992, 999991 '
      WRITE(2,*) '999990, 999990 '
C
1000  FORMAT(A1)
1010  FORMAT(10X,F12.8,4X,F12.8)
1020  FORMAT(3X,I3,5X,F12.8)
1030  FORMAT(F8.6)
1040  FORMAT(I4)
1050  FORMAT(1X,F8.6)
1060  FORMAT(1X,F8.6,' - J',F8.6)
1070  FORMAT(A12)
1080  FORMAT(/)
1090  FORMAT(1X,'ALPHA = ',F8.4,2X,'+ J ',F6.4,',    BETA = 'F8.4,2X,'+ J
     C ',F6.4)
1100  FORMAT(1X,'DESIRED DISTANCE = ',F8.5,',      SCALE FACTOR = ',F8.5)
      RETURN
      END
C
C
      SUBROUTINE ROTATE(NPNTS,METHOD,STARTND,STOPND)
C
C     THIS SUBROUTINE ROTATES THE SURFACE POINTS TO ALLOW FOR
C     A REARRANGING OF THE START NODE (FIRST DATA POINT).
C
      REAL MESH(0:200,5)
      INTEGER I, NPNTS, METHOD, STARTND, STOPND
      COMMON/BLK1/MESH
C
C
      IF(METHOD.EQ.6) THEN
           DO 10, I = 0, NPNTS-1
                MESH(I,1) = MESH(I,4)
```

```
              MESH(I,2) = MESH(I,5)
10         CONTINUE
C
           DO 20, I = STARTND-1, NPNTS-1
              MESH(I-(STARTND-1),4) = MESH(I,1)
              MESH(I-(STARTND-1),5) = MESH(I,2)
20         CONTINUE
C
           DO 30, I = 1, STARTND
              MESH(I+NPNTS-STARTND,4) = MESH(I-1,1)
              MESH(I+NPNTS-STARTND,5) = MESH(I-1,2)
30         CONTINUE
      ENDIF
      STOPND = STOPND - STARTND + 1
      IF(STOPND.GT.NPNTS) THEN
           STOPND = STOPND - NPNTS
      ELSEIF(STOPND.LT.0) THEN
           STOPND = NPNTS + STOPND +1
      ELSEIF(STOPND.EQ.0) THEN
           STOPND = 1
      ELSE
           STOPND = STOPND
      ENDIF
      RETURN
      END
C
C
      SUBROUTINE BOUND(MRES,NPNTS,METHOD,STOPND)
C
C     THIS SUBROUTINE REORDERS THE THE SURFACE POINTS
C     BASED ON THE DESIRED INPUT MESH RESOLU INN. THE
C     OBJECT IS ALSO BISECTED.
C
      REAL PERIM, DIST, MESH(0:200,5), TEMP, TEMP1, MRES, MRESN, MRESLW
      REAL MRESNL, MRESNR, TEMPR, TEMPL, DISTB, DZ, PI, MRESW, MRESRW
      INTEGER  I, PERND, BIND, NPNTS,STARTND, STOPND, METHOD, NEWND, J
      INTEGER M, L
      COMMON/BLK1/MESH
      COMMON/BLK2/PERND,BIND
C
      PI = 4.0*ATAN(1.0)
      PERIM = 0.0
C
      DO 10, I = 0, NPNTS-1
           DIST = SQRT((MESH(I+1,4)-MESH(I,4))**2+(MESH(I+1,5) -
     C     MESH(I,5))**2)
           PERIM = PERIM + DIST
           MESH(I+1,3) = PERIM
10    CONTINUE
      WRITE(6,*) 'PERIMETER LENGTH       = ', PERIM
      PERND = NINT(PERIM/MRES - AMOD(PERIM/MRES,2.0))
      BIND = (PERND - 2)/2
      WRITE(6,*) 'PERIMETER NODE #       = ', PERND
      MRESW = MRES/(2*PI)
      WRITE(6,*) 'YOUR REQUESTED MESH RESOLUTION OF ',MRESW
      IF((METHOD.EQ.3).OR.(METHOD.EQ.4).OR.(METHOD.EQ.5).OR.
```

```
     C(METHOD.EQ.6)) THEN
C
            MRESNL = (PERIM -MESH(STOPND-1,3))/FLOAT(PERND/2)
            MRESLW = MRESNL/(2*PI)
            MRESNR = MESH(STOPND-1,3)/FLOAT(PERND/2)
            MRESRW = MRESNR/(2*PI)
            WRITE(6,*) '. . .  HAS BEEN MODIFIED TO . . . '
            WRITE(6,*) 'LEFT  SIDE OF SEGMENT . . . ',MRESLW
            WRITE(6,*) 'RIGHT SIDE OF SEGMENT . . . ',MRESRW
      ELSE
            MRESN = PERIM/FLOAT(PERND)
            MRESW = MRESN/(2*PI)
            WRITE(6,*) '. . .  HAS BEEN MODIFIED TO . . . ',MRESW
      ENDIF
C
C     PERIMETER NODE INITIALIZATION (X,Y COORD)
C
      IF((METHOD.EQ.1).OR.(METHOD.EQ.2)) THEN
            DO 30, I = 0, PERND-1
                  TEMP = MRESN*I
                  J = 1
20                IF(TEMP.GT.MESH(J,3)) THEN
                        J = J + 1
                        GOTO 20
                  ENDIF
            TEMP1 = (TEMP - MESH(J-1,3))/(SQRT((MESH(J,4)-MESH(J-1,4))
     C      **2 + (MESH(J,5) - MESH(J-1,5))**2))
            MESH(I+1,1) = MESH(J-1,4) + TEMP1*(MESH(J,4)-MESH(J-1,4))
            MESH(I+1,2) = MESH(J-1,5) + TEMP1*(MESH(J,5)-MESH(J-1,5))
30          CONTINUE
C
      ELSE
            DO 60, I = 0, PERND-1
                  TEMPR = MRESNR*I
                  TEMPL = PERIM - MRESNL*(PERND - I)
                  IF(TEMPR.LE.MESH(STOPND-1,3)) THEN
                        J = 1
40                      IF(TEMPR.GT.MESH(J,3)) THEN
                              J = J + 1
                              GOTO 40
                        ENDIF
            TEMP1 = (TEMPR - MESH(J-1,3))/(SQRT((MESH(J,4)-MESH(J-1,4))
     C      **2 + (MESH(J,5) - MESH(J-1,5))**2))
            MESH(I+1,1) = MESH(J-1,4) + TEMP1*(MESH(J,4)-MESH(J-1,4))
            MESH(I+1,2) = MESH(J-1,5) + TEMP1*(MESH(J,5)-MESH(J-1,5))
                  ELSE
                        J = 1
50                      IF(TEMPL.GT.MESH(J,3)) THEN
                              J = J + 1
                              GOTO 50
                        ENDIF
            TEMP1 = (TEMPL - MESH(J-1,3))/(SQRT((MESH(J,4)-MESH(J-1,4))
     C      **2 + (MESH(J,5) - MESH(J-1,5))**2))
            MESH(I+1,1) = MESH(J-1,4) + TEMP1*(MESH(J,4)-MESH(J-1,4))
            MESH(I+1,2) = MESH(J-1,5) + TEMP1*(MESH(J,5)-MESH(J-1,5))
```

```
               ENDIF
60          CONTINUE
       ENDIF
C
C
C      BISECTION NODE INITIALIZATION (X,Y COORD)
C
       WRITE(6,*) 'BISECTION NODE #       = ',BIND
       IF((METHOD.EQ.1).OR.(METHOD.EQ.3)) THEN
            DO 70, I = 1, BIND
               TEMP1 = I/FLOAT(BIND + 1)
               MESH(I+PERND,1) = MESH(1,1) + TEMP1*(MESH(BIND+2,1) -
     CMESH(1,1))
               MESH(I+PERND,2) = MESH(1,2) + TEMP1*(MESH(BIND+2,2) -
     CMESH(1,2))
70          CONTINUE
C
       ELSE
            DO 80, I = 2, PERND+1
               MESH(PERND+I-1,1) = MESH(PERND+2-I,1) + 0.5*(MESH(I,1) -
     C         MESH(PERND+2-I,1))
               MESH(PERND+I-1,2) = MESH(PERND+2-I,2) + 0.5*(MESH(I,2) -
     C         MESH(PERND+2-I,2))
80          CONTINUE
C
C
       ENDIF
C
       IF((METHOD.EQ.5).OR.(METHOD.EQ.6)) THEN
            DO 90, M = 1, BIND
               MESH(M,4) = MESH(PERND+M,1)
               MESH(M,5) = MESH(PERND+M,2)
90          CONTINUE
            DISTB = 0.0
            MESH(0,3) = 0.0
            DO 100, L = 1, BIND+1
               IF(L.EQ.1) THEN
                    DIST = SQRT((MESH(1,1) - MESH(1,4))**2 +
     C(MESH(1,2) - MESH(1,5))**2)
               ELSEIF(L.EQ.BIND+1) THEN
                    DIST = SQRT((MESH(BIND,4) - MESH(BIND+2,1))
     C**2 + (MESH(BIND,5) - MESH(BIND+2,2))**2)
               ELSE
                    DIST = SQRT((MESH(L-1,4) - MESH(L,4))
     C**2 + (MESH(L-1,5) - MESH(L,5))**2)
               ENDIF
               DISTB = DISTB + DIST
               MESH(L,3) = DISTB
100         CONTINUE
            DZ = DISTB/(BIND + 1.0)
       WRITE(6,*) 'DZ SPACING             = ', DZ
C
            DO 120, I = 1, BIND
               TEMP = DZ*I
               J = 1
110            IF(TEMP.GT.MESH(J,3)) THEN
```

```
                  J = J + 1
                  GOTO 110
               ENDIF
               IF(J.EQ.1) THEN
            TEMP1 = TEMP/(SQRT((MESH(1,1) - MESH(1,4))**2 +
     C      (MESH(1,2) - MESH(1,5))**2))
            MESH(PERND+I,1) = MESH(1,1) + TEMP1*(MESH(1,4)
     C      - MESH(1,1))
            MESH(PERND+I,2) = MESH(1,2) + TEMP1*(MESH(1,5)
     C      - MESH(1,2))
               ELSE
            TEMP1 = (TEMP - MESH(J-1,3))/(SQRT((MESH(J,4) -
     C      MESH(J-1,4))**2 + (MESH(J,5) - MESH(J-1,5))**2))
            MESH(PERND+I,1) = MESH(J-1,4) + TEMP1*(MESH(J,4)
     C     - MESH(J-1,4))
            MESH(PERND+I,2) = MESH(J-1,5) + TEMP1*(MESH(J,5)
     C     - MESH(J-1,5))
               ENDIF
120         CONTINUE
         ENDIF
         RETURN
         END
C
C
         SUBROUTINE NORMAL
C
C        THIS ROUTINE COMPUTES THE X AND Y COMPONENTS
C        OF THE OUTWARD UNIT NORMAL AT EACH SURFACE POINT.
C
         REAL DR, DZ, DL, MESH(0:200,5)
         INTEGER I, PERND, BIND
         COMMON/BLK1/MESH
         COMMON/BLK2/PERND,BIND
         DO 10, I = 1, PERND
            IF(I.EQ.1) THEN
               DR = MESH(2,1) - MESH(PERND,1)
               DZ = MESH(2,2) - MESH(PERND,2)
               DL = SQRT(DR*DR+DZ*DZ)
               MESH(1,3)=-DZ/DL
               MESH(1,4)=DR/DL
            ELSEIF(I.EQ.PERND) THEN
               DR = MESH(1,1) - MESH(PERND-1,1)
               DZ = MESH(1,2) - MESH(PERND-1,2)
               DL = SQRT(DR*DR+DZ*DZ)
               MESH(PERND,3)=-DZ/DL
               MESH(PERND,4)=DR/DL
            ELSE
               DR = MESH(I+1,1) - MESH(I-1,1)
               DZ = MESH(I+1,2)- MESH(I-1,2)
               DL = SQRT(DR*DR+DZ*DZ)
               MESH(I,3)=-DZ/DL
               MESH(I,4)=DR/DL
            ENDIF
10       CONTINUE
         RETURN
```

```
      END
C
C
      SUBROUTINE NODSET(METHOD,NABC,NBMX,UNK)
C
C     THIS ROUTINE USES A TWO-SWEEP TECHNIQUE TO COMPUTE THE
C     NUMBER OF NODES ALONG EACH NODE ROW, I. A MESH
C     ORIENTATION ATTRIBUTE IS ALSO SET.
C
C     SET Z-AXIS SPACING AND ENDPOINT NODES
C
      REAL DZ, ZZ, MESH(0:200,5), D, DIST, DISTB
      INTEGER I, J, NODES(200), MOR(200), PERND, NOLD, NNEW, BIND
      INTEGER L, METHOD, NABC(100,3), NBMX, UNK
      CHARACTER*1 CHAR2, CHAR3, CHAR4
      COMMON/BLK1/MESH
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK6/MOR
      COMMON/BLK9/CHAR2,CHAR3,CHAR4
C
      UNK = 0
      NBMX = -99
      IF((METHOD.EQ.1).OR.(METHOD.EQ.3)) THEN
           DZ = (SQRT((MESH(1,2) - MESH(BIND+2,2))**2 +
     C     (MESH(1,1) - MESH(BIND+2,1))**2))/(BIND + 1.0)
      ELSE
           DISTB = 0.0
           DO 10, L = 1, BIND+1
                IF(L.EQ.1) THEN
                     DIST = SQRT((MESH(1,1) - MESH(PERND+1,1))**2 +
     C(MESH(1,2) - MESH(PERND+1,2))**2)
                ELSEIF(L.EQ.BIND+1) THEN
                     DIST = SQRT((MESH(PERND+BIND,1) - MESH(BIND+2,1))
     C**2 + (MESH(PERND+BIND,2) - MESH(BIND+2,2))**2)
                ELSE
                     DIST = SQRT((MESH(PERND+I-1,1) - MESH(PERND+I,1))
     C**2 + (MESH(PERND+I-1,2) - MESH(PERND+I,2))**2)
                ENDIF
                DISTB = DISTB + DIST
10         CONTINUE
           DZ = DISTB/(BIND + 1.0)
      ENDIF
      WRITE(6,*) 'BISECTION SPACING     = ', DZ
      NODES(1) = 2
      NODES(2) = 3
      NODES(BIND+2) = 2
      NODES(PERND+1) = 2
      NODES(PERND) = 3
C     PERFORMING FORWARD-SWEEP
      DO 20 I = 3, BIND+1
           NOLD = NODES(I-1)
           D = SQRT((MESH(I,1) - MESH(I+PERND-1,1))**2 +
     C     (MESH(I,2)-MESH(I+PERND-1,2))**2)
           NNEW = INT(0.1 + D/DZ) + 2
           NODES(I) = NNEW
```

```
          IF (NNEW.GT.NOLD+1) NODES(I) = NOLD + 1
          IF (NNEW.LT.NOLD-1) NODES(I) = NOLD - 1
          IF (NODES(I).LT.3) NODES(I) = 3
20    CONTINUE
C
      J = PERND + 2
      DO 30, I = PERND-1, BIND+3, -1
          NOLD = NODES(I+1)
          D = SQRT((MESH(I,1) - MESH(J,1))**2 +
     C    (MESH(I,2)-MESH(J,2))**2)
          NNEW = INT(0.1 + D/DZ) + 2
          NODES(I) = NNEW
          IF (NNEW.GT.NOLD+1) NODES(I) = NOLD + 1
          IF (NNEW.LT.NOLD-1) NODES(I) = NOLD - 1
          IF (NODES(I).LT.3) NODES(I) = 3
          J = J + 1
30    CONTINUE
C
C     BACK-SWEEP TO RESET LAST NODES IF NEEDED
C
      I = BIND + 2
40    I = I - 1
      IF (I.EQ.2) GO TO 50
      IF (NODES(I).LE.NODES(I+1)+1) GO TO 60
      NODES(I) = NODES(I+1) + 1
      GO TO 40
50    CONTINUE
      WRITE(6,*) ' PROGRAM ABORTED IN NODSET RIGHT SIDE BACKSWEEP '
      STOP
60    CONTINUE
C
      I = BIND + 2
70    I = I + 1
      IF (I.EQ.PERND) GO TO 80
      IF (NODES(I).LE.NODES(I-1)+1) GO TO 90
      NODES(I) = NODES(I-1) + 1
      GO TO 70
80    CONTINUE
      WRITE(6,*) ' PROGRAM ABORTED IN NODSET LEFT HALF BACKSWEEP '
      STOP
90    CONTINUE
C
C     FORWARD SWEEP TO LOAD MESH ORIENTATION ARRAY, MOR
C
      MOR(1) = 0
      MOR(BIND+2) = 0
      MOR(PERND+1) = 0
C
      DO 100, I=2, BIND+1
          IF(NODES(I+1).GT.NODES(I)) THEN
              MOR(I) = 0
          ELSEIF(NODES(I+1).LT.NODES(I)) THEN
              MOR(I) = 1
          ELSE
              IF(NODES(I+2).GT.NODES(I)) THEN
                  MOR(I) = 0
```

```
          ELSEIF(NODES(I+2).LT.NODES(I)) THEN
               MOR(I) = 1
          ELSE
               MOR(I) = MOR(I-1)
          ENDIF
     ENDIF
100   CONTINUE
C
      DO 110, I = PERND, BIND+3, -1
     IF(NODES(I-1).GT.NODES(I)) THEN
          MOR(I) = 0
     ELSEIF(NODES(I-1).LT.NODES(I)) THEN
          MOR(I) = 1
     ELSE
          IF(NODES(I-2).GT.NODES(I)) THEN
               MOR(I) = 0
          ELSEIF(NODES(I-2).LT.NODES(I)) THEN
               MOR(I) = 1
          ELSE
               MOR(I) = MOR(I+1)
          ENDIF
     ENDIF
110   CONTINUE
C     LOAD NABC ARRAY
      DO 120, I = 1, BIND+2
     IF(I.EQ.1) THEN
          NABC(1,1) = 0
          NABC(1,2) = 1
          NABC(1,3) = 3
     ELSEIF(I.LE.BIND) THEN
          NABC(I,1) = NABC(I-1,2)
          NABC(I,2) = NABC(I-1,3)
          NABC(I,3) = NODES(PERND-I+1) + NODES(I+1) - 3
     ELSEIF(I.EQ.BIND+1) THEN
          NABC(I,1) = NABC(I-1,2)
          NABC(I,2) = NABC(I-1,3)
          NABC(I,3) = 1
     ELSE
          NABC(I,1) = NABC(I-1,2)
          NABC(I,2) = NABC(I-1,3)
          NABC(I,3) = 0
     ENDIF
120   CONTINUE
C
      DO 130, I = 1, BIND+2
     IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
          WRITE(12,*) I,NABC(I,1),NABC(I,2),NABC(I,3)
     ENDIF
     UNK = UNK + NABC(I,2)
     IF(NABC(I,2).GE.NBMX) THEN
          NBMX = NABC(I,2)
     ENDIF
130   CONTINUE
C
      WRITE(*,*) 'MAXIMUM UNKNOWN WIDTH = ',NBMX
      WRITE(*,*) 'TOTAL # OF UNKNOWNS   = ',UNK
```

```
      WRITE(*,*) ' '
      RETURN
      END
C
C
C
      SUBROUTINE SORTER(I,LEL,LAND)
C
C     THIS SUBROUTINE GENERATES A MESH ROW FOR THE INPUT ROW I.
C     LOADING OF THE LOCAL NODE-ELEMENT CONNECTION MATRICES LND AND
C     NDL FOR ELEMENTS BETWEEN I AND I+1 NODE ROWS. REFERENCE IS TO
C     THE LEFT SIDE OF THE Ith ROW OR VECTOR.
C
      INTEGER NODES(200), MOR(200), LND(0:200,3), NDL(200,4), NCT(200)
      INTEGER I, PERND, BIND, LEL, LAND, J, K, LL, JJ, NN, KK, N, N1, N2
      INTEGER NDMX, NDS1L, NDS2L, NDS1R, NDS2R, LMX
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK4/LND,NDL,NCT
      COMMON/BLK6/MOR
      IF(I.EQ.BIND+2) THEN
           WRITE(6,*) 'ERRORED OUT IN SUBROUTINE SORTER, YOU ATTEMPTED'
           WRITE(6,*) '      TO CALL SORTER WITH I = BIND + 2!'
           WRITE(6,*) '            THIS ROW HAS NO ELEMENTS'
           RETURN
      ENDIF
      DO 20, J = 0, 200
           DO 10, K = 1, 3
                LND(J,K) = 0
10         CONTINUE
20    CONTINUE
      NDMX = 200
      NDS1L = NODES(PERND+2-I)
      NDS2L = NODES(PERND+1-I)
      NDS1R = NODES(I)
      NDS2R = NODES(I+1)
      LMX = NDS1L + NDS1R + NDS2L + NDS2R - 2
C
C
C     LEFTSIDE OF THE BISECTION SEGMENT
C
C     TOP ROW
C
      IF(I.EQ.1) THEN
           LND(1,1) = 4
           LND(1,2) = 1
           LND(1,3) = 3
           LND(2,1) = 1
           LND(2,2) = 4
           LND(2,3) = 2
           LND(3,1) = 5
           LND(3,2) = 2
           LND(3,3) = 4
           LND(4,1) = 5
           LND(4,2) = 2
           LND(4,3) = 6
```

```
            LND(5,1) = 1
            LND(5,2) = 6
            LND(5,3) = 2
            LND(6,1) = 6
            LND(6,2) = 1
            LND(6,3) = 7
            LEL = 6
            LAND = 7
C
C     BOTTOM ROW
C
      ELSEIF(I.EQ.BIND+1) THEN
            LND(1,1) = 2
            LND(1,2) = 6
            LND(1,3) = 1
            LND(2,1) = 6
            LND(2,2) = 2
            LND(2,3) = 7
            LND(3,1) = 3
            LND(3,2) = 7
            LND(3,3) = 2
            LND(4,1) = 3
            LND(4,2) = 7
            LND(4,3) = 4
            LND(5,1) = 6
            LND(5,2) = 4
            LND(5,3) = 7
            LND(6,1) = 4
            LND(6,2) = 6
            LND(6,3) = 5
            LEL = 6
            LAND = 7
C
C     EQUAL NODE NUMBERS
C
      ELSEIF(NDS1L.EQ.NDS2L) THEN
C     FOR MOR = 0 (LH ORIENTATION)
            IF(MOR(PERND+2-I).EQ.0) THEN
                  LND(1,1) = 1
                  LND(1,2) = NDS1L + NDS1R
                  LND(1,3) = 2
                  LND(2,1) = NDS1L + NDS1R + 1
                  LND(2,2) = 2
                  LND(2,3) = NDS1L + NDS1R
                  DO 40, N = 1, NDS1L-2
                        N1 = 2*N + 1
                        N2 = N1 + 1
                        DO 30, K = 1, 3
                              LND(N1,K) = LND(1,K) + N
                              LND(N2,K) = LND(2,K) + N
30                      CONTINUE
40                CONTINUE
C     FOR MOR = 1 (RH ORIENTATION)
            ELSE
                  LND(1,1) = NDS1L + NDS1R
                  LND(1,2) = 1
```

```
                    LND(1,3) = NDS1L + NDS1R + 1
                    LND(2,1) = 2
                    LND(2,2) = NDS1L + NDS1R + 1
                    LND(2,3) = 1
                    DO 60, N = 1, NDS1L-2
                         N1 = 2*N + 1
                         N2 = N1 + 1
                         DO 50, K = 1, 3
                              LND(N1,K) = LND(1,K) + N
                              LND(N2,K) = LND(2,K) + N
50                       CONTINUE
60                  CONTINUE
               ENDIF
C
C     LEFTHAND MESH ORIENTATION
C
      ELSEIF(MOR(PERND+2-I).EQ.0) THEN
               LND(1,1) = NDS1L + NDS1R + 1
               LND(1,2) = 1
               LND(1,3) = NDS1L + NDS1R
               LND(0,1) = 0
               LND(0,2) = NDS1L + NDS1R
               LND(0,3) = 1
      DO 80, N = 1, NDS1L-1
               N1 = 2*N
               N2 = N1 + 1
               DO 70, K = 1, 3
                    LND(N1,K) = LND(0,K) + N
                    LND(N2,K) = LND(1,K) + N
70             CONTINUE
80    CONTINUE
C
C     RIGHTHAND MESH ORIENTATION
C
      ELSE
               LND(1,1) = 2
               LND(1,2) = NDS1L + NDS1R
               LND(1,3) = 1
               LND(0,1) = NDS1L + NDS1R - 1
               LND(0,2) = 1
               LND(0,3) = NDS1L + NDS1R
      DO 100, N = 1, NDS1L-2
               N1 = 2*N
               N2 = N1 + 1
               DO 90, K = 1, 3
                    LND(N1,K) = LND(0,K) + N
                    LND(N2,K) = LND(1,K) + N
90             CONTINUE
100   CONTINUE
      ENDIF
C
      IF((I.EQ.1).OR.(I.EQ.BIND+1)) THEN
               GOTO 230
      ENDIF
C
```

```
      LEL = N2
C
C
C     RIGHTSIDE OF THE BISECTION SEGMENT
C
C
C     EQUAL NODE NUMBERS
C
      IF(NDS1R.EQ.NDS2R) THEN
C     MOR = 0 (LH ORIENTATION)
           IF(MOR(I).EQ.0) THEN
                LND(LEL+1,1) = NDS1L + NDS1R + NDS2L - 1
                LND(LEL+1,2) = NDS1L
                LND(LEL+1,3) = NDS1L + NDS1R + NDS2L
                LND(LEL+2,1) = NDS1L + 1
                LND(LEL+2,2) = NDS1L + NDS1R + NDS2L
                LND(LEL+2,3) = NDS1L
                DO 120, N = 1, NDS1R-2
                     N1 = 2*N + 1 + LEL
                     N2 = N1 + 1
                          DO 110, K = 1, 3
                               LND(N1,K) = LND(LEL+1,K) + N
                               LND(N2,K) = LND(LEL+2,K) + N
110                       CONTINUE
120                  CONTINUE
           LEL = N2
           LAND = LMX
C     MOR = 1 (RH ORIENTATION)
           ELSE
                LND(LEL+1,1) = NDS1L
                LND(LEL+1,2) = NDS1L + NDS1R + NDS2L - 1
                LND(LEL+1,3) = NDS1L + 1
                LND(LEL+2,1) = NDS1L + NDS1R + NDS2L
                LND(LEL+2,2) = NDS1L + 1
                LND(LEL+2,3) = NDS1L + NDS1R + NDS2L - 1
                DO 140, N = 1, NDS1R-2
                     N1 = 2*N + 1 + LEL
                     N2 = N1 + 1
                          DO 130, K = 1, 3
                               LND(N1,K) = LND(LEL+1,K) + N
                               LND(N2,K) = LND(LEL+2,K) + N
130                       CONTINUE
140                  CONTINUE
           ENDIF
           LEL = N2
           LAND = LMX
C
C     LEFT HAND MESH ORIENTATION
C
      ELSEIF(MOR(I).EQ.0) THEN
           LND(LEL+1,1) = NDS1L + NDS1R + NDS2L - 1
           LND(LEL+1,2) = NDS1L
           LND(LEL+1,3) = NDS1L + NDS1R + NDS2L
           LND(LEL+2,1) = NDS1L + 1
           LND(LEL+2,2) = NDS1L + NDS1R + NDS2L
```

```
          LND(LEL+2,3) = NDS1L
C
          DO 160, N = 1, NDS1R-1
               N1 = 2*N + LEL + 1
               DO 150, K = 1, 3
                    LND(N1,K) = LND(LEL+1,K) + N
150            CONTINUE
160       CONTINUE
C
          DO 180, N = 1, NDS1R-2
               N2 = 2*N + LEL + 2
               DO 170, K = 1, 3
                    LND(N2,K) = LND(LEL+2,K) + N
170            CONTINUE
180       CONTINUE
C
          LEL = N1
          LAND = LMX
C
C     RIGHT HAND MESH ORIENTATION
C
      ELSE
          LND(LEL+1,1) = NDS1L
          LND(LEL+1,2) = NDS1L + NDS1R + NDS2L - 1
          LND(LEL+1,3) = NDS1L + 1
          LND(LEL+2,1) = NDS1L + NDS1R + NDS2L
          LND(LEL+2,2) = NDS1L + 1
          LND(LEL+2,3) = NDS1L + NDS1R + NDS2L - 1
C
          DO 200, N = 1, NDS1R-2
               N1 = 2*N + LEL + 1
               DO 190, K = 1, 3
                    LND(N1,K) = LND(LEL+1,K) + N
190            CONTINUE
200       CONTINUE
C
          DO 220, N = 1, NDS1R-3
               N2 = 2*N + LEL + 2
               DO 210, K = 1, 3
                    LND(N2,K) = LND(LEL+2,K) + N
210            CONTINUE
220       CONTINUE
C
          LEL = N1
          LAND = LMX
C
C
      ENDIF
C
C
C
C     ZEROING NDL AND NCT PRIOR TO FILL
C
230   DO 250, N = 1, NDMX
          NCT(N) = 0
          DO 240, K = 1, 4
```

```
                NDL(N,K) = 0
240         CONTINUE
250   CONTINUE
C
C     SCANNING LND ARRAY TO LOAD NDL
C
      DO 270 LL = 1, LEL
          DO 260 JJ = 1, 3
                NN = LND(LL,JJ)
                NCT(NN) = NCT(NN) + 1
                KK = NCT(NN)
                NDL(NN,KK) = LL
260         CONTINUE
270   CONTINUE
C
      END
C
C
      SUBROUTINE FINDER(I,LAND,DIST)
C
C     THIS SUBROUTINE DETERMINES THE (X,Y) COORDINATES OF EACH
C     NODE IN THE CALLING Ith ROW OR VECTOR MESH.
C
      INTEGER I, J, LAND, PERND, BIND, NODES(200)
      REAL MESH(0:200,5), NDP(200,2), DIST
      COMMON/BLK1/MESH
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK5/NDP
      IF(I.EQ.1) THEN
          NDP(1,1) = MESH(1,1)+MESH(1,3)*DIST
          NDP(1,2) = MESH(1,2)+MESH(1,4)*DIST
          NDP(2,1) = MESH(1,1)
          NDP(2,2) = MESH(1,2)
          NDP(3,1) = MESH(PERND,1)+MESH(PERND,3)*DIST
          NDP(3,2) = MESH(PERND,2)+MESH(PERND,4)*DIST
          NDP(4,1) = MESH(PERND,1)
          NDP(4,2) = MESH(PERND,2)
          NDP(5,1) = MESH(PERND+1,1)
          NDP(5,2) = MESH(PERND+1,2)
          NDP(6,1) = MESH(2,1)
          NDP(6,2) = MESH(2,2)
          NDP(7,1) = MESH(2,1)+MESH(2,3)*DIST
          NDP(7,2) = MESH(2,2)+MESH(2,4)*DIST
      ELSEIF(I.EQ.BIND+1) THEN
          NDP(1,1) = MESH(BIND+3,1)+MESH(BIND+3,3)*DIST
          NDP(1,2) = MESH(BIND+3,2)+MESH(BIND+3,4)*DIST
          NDP(2,1) = MESH(BIND+3,1)
          NDP(2,2) = MESH(BIND+3,2)
          NDP(3,1) = MESH(PERND+BIND,1)
          NDP(3,2) = MESH(PERND+BIND,2)
          NDP(4,1) = MESH(BIND+1,1)
          NDP(4,2) = MESH(BIND+1,2)
          NDP(5,1) = MESH(BIND+1,1)+MESH(BIND+1,3)*DIST
          NDP(5,2) = MESH(BIND+1,2)+MESH(BIND+1,4)*DIST
          NDP(6,1) = MESH(BIND+2,1)+MESH(BIND+2,3)*DIST
```

```
          NDP(6,2) = MESH(BIND+2,2)+MESH(BIND+2,4)*DIST
          NDP(7,1) = MESH(BIND+2,1)
          NDP(7,2) = MESH(BIND+2,2)
      ELSEIF(I.EQ.BIND+2) THEN
          WRITE(6,*) ' ERRORED OUT IN SUBROUTINE FINDER, YOU ATTEMPTED'
          WRITE(6,*)  '      TO CALL FINDER WITH I = BIND + 2!'
          WRITE(6,*) '   THIS ROW HAS NO ELEMENTS AND NO COORDINATES'
      ELSE
C         NODE 1
          NDP(1,1) = MESH(PERND-I+2,1)+MESH(PERND-I+2,3)*DIST
          NDP(1,2) = MESH(PERND-I+2,2)+MESH(PERND-I+2,4)*DIST
C         NODE 2 TO THE BISECTION SEGMENT
          DO 10, J = 2, NODES(PERND-I+2)
              NDP(J,1)=MESH(PERND-I+2,1)-(J-2)*(MESH(PERND-I+2,1)-
     C        MESH(PERND+I-1,1))/(NODES(PERND-I+2)-2)
              NDP(J,2)=MESH(PERND-I+2,2)-(J-2)*(MESH(PERND-I+2,2)-
     C        MESH(PERND+I-1,2))/(NODES(PERND-I+2)-2)
10        CONTINUE
C         BISECTION SEGMENT TO THE RIGHTSIDE SURFACE
          DO 20, J = 3, NODES(I)
              NDP(NODES(PERND-I+2)+J-2,1)=MESH(PERND+I-1,1)+(J-2)*
     C        (MESH(I,1)-MESH(PERND+I-1,1))/(NODES(I)-2)
              NDP(NODES(PERND-I+2)+J-2,2)=MESH(PERND+I-1,2)+(J-2)*
     C        (MESH(I,2)-MESH(PERND+I-1,2))/(NODES(I)-2)
20        CONTINUE
C         Ith ROWS LAST NODE
          NDP(NODES(PERND-I+2)+NODES(I)-1,1)=MESH(I,1)+MESH(I,3)*DIST
          NDP(NODES(PERND-I+2)+NODES(I)-1,2)=MESH(I,2)+MESH(I,4)*DIST
C         I+1th ROWS FIRST NODE
          NDP(NODES(PERND-I+2)+NODES(I),1)=MESH(PERND-I+1,1)+MESH
     C    (PERND-I+1,3)*DIST
          NDP(NODES(PERND-I+2)+NODES(I),2)=MESH(PERND-I+1,2)+MESH
     C    (PERND-I+1,4)*DIST
C         I+1TH ROW (NODE 2) TO THE BISECTION SEGMENT
          DO 30, J = 2, NODES(PERND-I+1)
              NDP(J+NODES(PERND-I+2)+NODES(I)-1,1)=MESH(PERND-I+1,
     C        1)-(J-2)*(MESH(PERND-I+1,1)-MESH(PERND+I,1))/
     C        (NODES(PERND-I+1)-2)
              NDP(J+NODES(PERND-I+2)+NODES(I)-1,2)=MESH(PERND-I+1,
     C        2)-(J-2)*(MESH(PERND-I+1,2)-MESH(PERND+I,2))/
     C        (NODES(PERND-I+1)-2)
30        CONTINUE
C         I+1th ROW BISECTION SEGMENT TO THE RIGHTSIDE SURFACE
          DO 40, J = 3, NODES(I+1)
              NDP(LAND-NODES(I+1)+J-1,1)=MESH(PERND+I,1)+(J-2)*
     C        (MESH(I+1,1)-MESH(PERND+I,1))/(NODES(I+1)-2)
              NDP(LAND-NODES(I+1)+J-1,2)=MESH(PERND+I,2)+(J-2)*
     C        (MESH(I+1,2)-MESH(PERND+I,2))/(NODES(I+1)-2)
40        CONTINUE
C         LAST NODE
          NDP(LAND,1) = MESH(I+1,1)+MESH(I+1,3)*DIST
          NDP(LAND,2) = MESH(I+1,2)+MESH(I+1,4)*DIST
C
      ENDIF
      RETURN
```

```
      END
C
C
      SUBROUTINE VARINT(J,F,ALPHA,BETA,AREA,LND)
C
C     GENERATING VARIATIONAL FINITE ELEMENT AREA INTEGRATIONS OF THE
C     LINEAR BASIS FUNCTION LAGRANGIAN FOR THE HELMHOLTZ EQUATION.
C     THESE ARE RETURNED IN F(3,3).  X(3) AND Y(3) ARE THE WAVENUMBER
C     NORMALIZED CARTESIAN COORDINATES OF THE TRIANGLE VERTICES.
C     X = Ko*x, Y = Ko*y     ALPHA AND BETA ARE COMPLEX
C     MATERIAL PARAMETERS WITHIN THE ELEMENT.
C
C       FOR TM INCIDENCE:  ALPHA = 1/ur;  BETA = er
C       FOR TE INCIDENCE:  ALPHA = 1/er;  BETA = ur
C
C     OUTPUTS :    F(3,3) - FINITE ELEMENT AREA INTEGRATION
C                  AREA   - AREA OFF A TRIANGLE
C
      COMPLEX ALPHA, BETA, B12, F(3,3)
      REAL NDP(200,2), X(3), Y(3), T(3,3), AREA, DET
      INTEGER L, K, J, LND(0:200,3)
      COMMON/BLK5/NDP
C
      X(1) = NDP(LND(J,1),1)
      X(2) = NDP(LND(J,2),1)
      X(3) = NDP(LND(J,3),1)
      Y(1) = NDP(LND(J,1),2)
      Y(2) = NDP(LND(J,2),2)
      Y(3) = NDP(LND(J,3),2)
      DET  = X(2)*Y(3) + X(3)*Y(1) + X(1)*Y(2) - X(3)*Y(2) -
     CX(1)*Y(3) - X(2)*Y(1)
      AREA = ABS(0.5*DET)
      B12 = BETA/12.
      T(1,1) = (Y(2) - Y(3))/DET
      T(1,2) = (Y(3) - Y(1))/DET
      T(1,3) = (Y(1) - Y(2))/DET
      T(2,1) = (X(3) - X(2))/DET
      T(2,2) = (X(1) - X(3))/DET
      T(2,3) = (X(2) - X(1))/DET
      T(3,1) = (X(2)*Y(3) - X(3)*Y(2))/DET
      T(3,2) = (X(3)*Y(1) - X(1)*Y(3))/DET
      T(3,3) = (X(1)*Y(2) - X(2)*Y(1))/DET
      DO 10, K = 1, 3
           DO 10, L = 1, 3
                F(K,L) = ALPHA*(T(1,K)*T(1,L) + T(2,K)*T(2,L)) - B12
                IF(K.EQ.L) F(K,L) = F(K,L) - B12
10    F(K,L) = AREA*F(K,L)
C
      RETURN
      END
C
C
      SUBROUTINE LOADER(BCOND,OFFSET,ALPHA,BETA,NABC,IMX,NBMX,EO,SURBC,
     CCHAR1,LINE,MODE,XORIGIN,YORIGIN,SVEC,CHAR5)
C
      COMPLEX A(50,50),B(50,50),C(50,50),P(50,100)
```

```
      COMPLEX F(3,3),FROW(100,3,3),BCOND(100),LINE(50)
      COMPLEX ALPHA, BETA, DETERM, SURBC(100), SVEC(50,100)
      REAL OFFSET,MINAREA,MAXAREA,AREA,RATIO,E0,K0,XORIGIN,YORIGIN
      REAL UBCOND
      INTEGER I,J,JD,K,L,NDTOP,NDBOT,NDTOT,NOD,KND,ND(3),LEL,LAND
      INTEGER LND(0:?00,3), NDL(200,4), NCT(200), PERND, BIND, JJ
      INTEGER NODES(200), MINROW, MINEL, MAXROW, MAXEL, TCALL, NL
      INTEGER N,M,NBMX,NABC(100,3),INORM,IMX,MODE
      CHARACTER*1 CHAR1, CHAR2, CHAR3, CHAR4, CHAR5
C
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK4/LND,NDL,NCT
      COMMON/BLK7/MINAREA,MINROW,MINEL,MAXAREA,MAXROW,MAXEL,AREA
      COMMON/BLK8/A,B,C,P
      COMMON/BLK9/CHAR2, CHAR3, CHAR4
C
      UBCOND = 1.0
      INORM = 0
      IMX - BIND + 2
      I = 1
      TCALL = BIND + 1
C
      IF((CHAR3.EQ.'D').OR.(CHAR3.EQ.'d')) THEN
           WRITE(20,1030)
           WRITE(20,1040)
           WRITE(20,1050)
           WRITE(20,1060)
           WRITE(20,1070)
      ENDIF
           WRITE(*,1000) I,TCALL
C
      CALL SORTER(I,LEL,LAND)
      CALL FINDER(I,LAND,OFFSET)
      IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
      CALL BNDC(I,BCOND,E0,SURBC,CHAR1,ALPHA,BETA,LINE,MODE,XORIGIN,
     CYORIGIN,CHAR4)
      ENDIF
      CALL FILL(I,LEL,FROW,ALPHA,BETA)
      IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
      CALL ZERO
      ENDIF
C     ESTABLISH THE NUMBER OF NODES IN THE I = 1 AND I = 2 ROWS
      NDTOP = 2
      NDBOT = 5
      NDTOT = 7
C
C     B, C & P FOR I = 1 (TOP)
C
      J = 1
      DO 70, JD = 1, NCT(2)
           L = NDL(2,JD)
           DO 50, K = 1, 3
                ND(K) = LND(L,K)
                IF(ND(K).EQ.2) KND = K
50         CONTINUE
```

```
          DO 60, K = 1, 3
               NL = ND(K)
C     BOUNDARY CONDITION
               IF(NL.EQ.1) THEN
                    P(J,1) = -UBCOND*FROW(L,KND,K) + P(J,1)
C     UPPER ROW
               ELSEIF(NL.EQ.2) THEN
                    B(J,1) = FROW(L,KND,K) + B(J,1)
C     LOWER ROW
               ELSEIF((NL.GT.3).AND.(NL.LT.7)) THEN
                    C(J,NL-3) = FROW(L,KND,K) + C(J,NL-3)
C     ERROR
               ELSE
                    WRITE(*,*) NL,'ERROR - INDEX EQUALS 3 OR 7'
               ENDIF
60        CONTINUE
70    CONTINUE
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
          WRITE(1,*) I
          DO 72, M = 1, NABC(I,2)
               WRITE(1,1020) (REAL(B(M,N)), N = 1, NABC(I,2))
72        CONTINUE
               WRITE(1,*) ' '
          DO 73, M = 1, NABC(I,2)
               WRITE(1,1020) (REAL(C(M,N)), N = 1, NABC(I,3))
73        CONTINUE
               WRITE(1,*) ' '
          DO 74, M = 1, NABC(I,2)
               WRITE(1,1020) (REAL(P(M,N)), N = 1, PERND)
74        CONTINUE
C
          WRITE(1,*) ' '
          WRITE(1,*) ' '
          WRITE(1,*) ' '
      ENDIF
C
      IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
      CALL MARCH(I,IMX,NBMX,NABC(I,1),NABC(I,2),NABC(I,3),INORM,SVEC)
      CALL ZERO
      ENDIF
C     BEGIN LOOPING
      DO 900, I = 1, BIND
C
          DO 400, J = 1, NDBOT-2
               NOD = J + NDTOP + 1
               DO 300, JD = 1, NCT(NOD)
                    L = NDL(NOD,JD)
                    DO 100, K = 1, 3
                         ND(K) = LND(L,K)
                         IF(ND(K).EQ.NOD) KND = K
100                 CONTINUE
                    DO 200, K = 1, 3
                         NL = ND(K)
C     BOUNDARY CONDITION
      IF((I.EQ.1).AND.(NL.EQ.1)) THEN
```

```
          P(J,NL) = -UBCOND*FROW(L,KND,K) + P(J,NL)
      ELSEIF(NL.EQ.1) THEN
          P(J,PERND-I+2)=-UBCOND*FROW(L,KND,K)+P(J,PERND-I+2)
C     SPECIAL CASE FOR NODE #2 AND I = 1
      ELSEIF((I.EQ.1).AND.(NL.EQ.2))THEN
          A(J,1) = FROW(L,KND,K) + A(J,1)
      ELSEIF(NL.EQ.NDTOP) THEN
          P(J,I) = -UBCOND*FROW(L,KND,K) + P(J,I)
      ELSEIF(NL.EQ.NDTOP+1) THEN
          P(J,PERND-I+1)=-UBCOND*FROW(L,KND,K)+P(J,PERND-I+1)
      ELSEIF(NL.EQ.NDTOT) THEN
          P(J,I+1) = -UBCOND*FROW(L,KND,K) + P(J,I+1)
C     UPPER ROW
      ELSEIF(NL.LT.NDTOP) THEN
          A(J,NL-1) = FROW(L,KND,K) + A(J,NL-1)
C     LOWER ROW
      ELSEIF(NL.LT.NDTOT) THEN
          B(J,NL-NDTOP-1) = FROW(L,KND,K) + B(J,NL-NDTOP-1)
C     ERROR
      ELSE
          WRITE(*,*) NL,'ERROR - DUE TO INDEX GREATER THAN NODE TOTAL'
      ENDIF
C
200                           CONTINUE
300                       CONTINUE
400                   CONTINUE
C
C     INDEX FOR NEXT ROW AND COMPLETE  B, C, P FILLS
C
          JJ = I + 1
          WRITE(6,1010) JJ,TCALL
          CALL SORTER(JJ,LEL,LAND)
          CALL FINDER(JJ,LAND,OFFSET)
          IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
          CALL BNDC(JJ,BCOND,E0,SURBC,CHAR1,ALPHA,BETA,LINE,MODE,
    C     XORIGIN,YORIGIN,CHAR4)
          ENDIF
          CALL FILL(JJ,LEL,FROW,ALPHA,BETA)
C     ESTABLISH THE NUMBER OF NODES IN THE I+1th AND I+2th ROWS
          IF(JJ.NE.(BIND+1)) THEN
              NDTOP = NODES(PERND+2-JJ) + NODES(JJ) - 1
              NDBOT = NODES(PERND+1-JJ) + NODES(JJ+1) - 1
              NDTOT = NDBOT + NDTOP
          ELSE
              NDTOP = 5
              NDBOT = 2
              NDTOT = 7
          ENDIF
C
          DO 800, J = 1, NDTOP-2
              NOD = J + 1
              DO 700, JD = 1, NCT(NOD)
                  L = NDL(NOD,JD)
                  DO 500, K = 1, 3
                      ND(K) = LND(L,K)
```

```
                        IF(ND(K).EQ.NOD) KND = K
500                CONTINUE
                   DO 600, K = 1, 3
                        NL = ND(K)
C     BOUNDARY CONDITION
      IF(NL.EQ.1) THEN
        P(J,PERND-JJ+2)=-UBCOND*FROW(L,KND,K)+P(J,PERND-JJ+2)
      ELSEIF(NL.EQ.NDTOP) THEN
        P(J,JJ)    -UBCOND*FROW(L,KND,K) + P(J,JJ)
      ELSEIF(NL.EQ.NDTOP+1) THEN
        P(J,PERND-JJ+1)=-UBCOND*FROW(L,KND,K)+P(J,PERND-JJ+1)
      ELSEIF((JJ.EQ.(BIND+1)).AND.(NL.EQ.NDTOT)) THEN
        C(J,1) = FROW(L,KND,K) + C(J,1)
      ELSEIF(NL.EQ.NDTOT) THEN
        P(J,JJ+1) = -UBCOND*FROW(L,KND,K) + P(J,JJ+1)
C     UPPER ROW
      ELSEIF(NL.LT.NDTOP) THEN
        B(J,NL-1) = FROW(L,KND,K) + B(J,NL-1)
C     LOWER ROW
      ELSEIF(NL.LT.NDTOT) THEN
        C(J,NL-NDTOP-1) = FROW(L,KND,K) + C(J,NL-NDTOP-1)
C     ERROR
      ELSE
        WRITE(*,*) NL,'ERROR - DUE TO INDEX GREATER THAN NODE TOTAL'
      ENDIF
C
600                CONTINUE
700           CONTINUE
800       CONTINUE
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
          WRITE(1,*) JJ
          DO 91, M = 1, NABC(JJ,2)
              WRITE(1,1020) (REAL(A(M,N)), N = 1, NABC(JJ,1))
91        CONTINUE
              WRITE(1,*) ' '
          DO 92, M = 1, NABC(JJ,2)
              WRITE(1,1020) (REAL(B(M,N)), N = 1, NABC(JJ,2))
92        CONTINUE
              WRITE(1,*) ' '
          DO 93, M = 1, NABC(JJ,2)
              WRITE(1,1020) (REAL(C(M,N)), N = 1, NABC(JJ,3))
93        CONTINUE
              WRITE(1,*) ' '
          DO 94, M = 1, NABC(JJ,2)
              WRITE(1,1020) (REAL(P(M,N)), N = 1, PERND)
94        CONTINUE
C
          WRITE(1,*) ' '
          WRITE(1,*) ' '
          WRITE(1,*) ' '
      ENDIF
C
      IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
      CALL MARCH(JJ,IMX,NBMX,NABC(JJ,1),NABC(JJ,2),NABC(JJ,3),INORM,
     CSVEC)
```

```
      CALL ZERO
      ENDIF
C
900   CONTINUE
C
C     LOAD A, B & P FOR I = BIND + 2 (BOTTOM)
C
      J = 1
      DO 30, JD = 1, NCT(7)
           L = NDL(7,JD)
           DO 10, K = 1, 3
                ND(K) = LND(L,K)
                IF(ND(K).EQ.7) KND = K
10                   CONTINUE
                     DO 20, K = 1, 3
                          NL = ND(K)
C     BOUNDARY CONDITION
      IF(NL.EQ.6) THEN
           P(J,JJ+1) = -UBCOND*FROW(L,KND,K) + P(J,JJ+1)
C     UPPER ROW
      ELSEIF((NL.GT.1).AND.(NL.LT.5)) THEN
           A(J,NL-1) = FROW(L,KND,K) + A(J,NL-1)
C     LOWER ROW
      ELSEIF(NL.EQ.7) THEN
           B(J,1) = FROW(L,KND,K) + B(J,1)
C     ERROR
      ELSE
           WRITE(*,*) NL,'ERROR - INDEX EQUAL TO 1 OR 5'
      ENDIF
C
20         CONTINUE
30    CONTINUE
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
           WRITE(1,*) TCALL+1
           DO 61, M = 1, NABC(TCALL+1,2)
               WRITE(1,1020) (REAL(A(M,N)), N = 1, NABC(TCALL+1,1))
61         CONTINUE
               WRITE(1,*) ' '
           DO 62, M = 1, NABC(TCALL+1,2)
               WRITE(1,1020) (REAL(B(M,N)), N = 1, NABC(TCALL+1,2))
62         CONTINUE
               WRITE(1,*) ' '
           DO 64, M = 1, NABC(TCALL+1,2)
               WRITE(1,1020) (REAL(P(M,N)), N = 1, PERND)
64         CONTINUE
C
           WRITE(1,*) ' '
           WRITE(1,*) ' '
           WRITE(1,*) ' '
      ENDIF
C
      WRITE(6,*) '    FINAL INVERSION'
      IF(.NOT.((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m'))) THEN
      CALL MARCH(JJ+1,IMX,NBMX,NABC(JJ+1,1),NABC(JJ+1,2),NABC(JJ+1,3),
     CINORM,SVEC)
```

```
      ENDIF
C
      WRITE(2,*) ' END  END '
      WRITE(6,*) ' '
      WRITE(6,*) 'MINIMUM AREA = ',MINAREA
      WRITE(6,*) 'AT ROW ',MINROW,'     AND ELEMENT NUMBER ',MINEL
      WRITE(6,*) 'MAXIMUM AREA = ',MAXAREA
      WRITE(6,*) 'AT ROW ',MAXROW,'     AND ELEMENT NUMBER ',MAXEL
      RATIO = MAXAREA/MINAREA
      WRITE(6,*) 'AREA RATIO    = ',RATIO
      IF(RATIO.GT.2.5) THEN
           WRITE(6,*) 'YOU SHOULD CONSIDER ABORTING THIS RUN AND '
           WRITE(6,*) 'LOOKING AT THE MESH IN CURVE DIGITIZER '
           WRITE(6,*) 'A BETTER METHOD MAY BE AVAILABLE '
      ENDIF
      IF((CHAR3.EQ.'D').OR.(CHAR3.EQ.'d')) THEN
           WRITE(20,1080)
           WRITE(20,1090)
           WRITE(20,1100)
      ENDIF
C
1000  FORMAT( I6,' OUT OF',I6,' CALLS ')
1010  FORMAT( I6,' OUT OF',I6,' CALLS ')
1020  FORMAT( 20(E14.8,1X,E14.8,1X))
1030  FORMAT( 6X,'CALL COMPRS')
1040  FORMAT( 6X,'CALL NOBRDR')
1050  FORMAT( 6X,'CALL PAGE(8.0,10.0)')
1060  FORMAT( 6X,'CALL AREA2D(5.0,7.0)')
1070  FORMAT( 6X,'CALL FRAME')
1080  FORMAT( 6X,'CALL DONEPL')
1090  FORMAT( 6X,'STOP')
1100  FORMAT( 6X,'END')
C
      RETURN
      END
C

C
      SUBROUTINE FILL(I,LEL,FROW,ALPHA,BETA)
C
      COMPLEX F(3,3), FROW(100,3,3), ALPHA, BETA, A, B
      REAL AREA, MINAREA, MAXAREA
      REAL NDP(200,2)
      INTEGER I, J, K, L, LEL, LND(0:200,3), NDL(200,4), NCT(200)
      INTEGER MINROW, MINEL, MAXROW, MAXEL, M
      CHARACTER*1 CHAR2, CHAR3, CHAR4
      COMMON/BLK4/LND,NDL,NCT
      COMMON/BLK5/NDP
      COMMON/BLK7/MINAREA,MINROW,MINEL,MAXAREA,MAXROW,MAXEL,AREA
      COMMON/BLK9/CHAR2, CHAR3, CHAR4
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
           WRITE(11,*) 'ROW NUMBER = ',I
      ENDIF
      DO 30, J = 1, LEL
           IF((CHAR4.EQ.'U').OR.(CHAR4.EQ.'u')) THEN
```

```
                A = ALPHA
                B = BETA
           ELSE
                IF((J.EQ.1).OR.(J.EQ.2).OR.(J.EQ.LEL-1).OR.(J.EQ.LEL))
C               THEN
                     A = (1.0,0.0)
                     B = (1.0,0.0)
                ELSE
                     A = ALPHA
                     B = BETA
                ENDIF
           ENDIF
           CALL VARINT(J,F,A,B,AREA,LND)
           IF((J.GT.2).AND.(J.LT.LEL-1)) THEN
                IF(AREA.LT.MINAREA) THEN
                     MINAREA = AREA
                     MINROW = I
                     MINEL = J
                ELSEIF(AREA.GT.MAXAREA) THEN
                     MAXAREA = AREA
                     MAXROW = I
                     MAXEL = J
                ENDIF
           ENDIF
           DO 20, K = 1, 3
                WRITE(2,*) NDP(LND(J,K),1), NDP(LND(J,K),2)
                DO 10, L = 1, 3
                     FROW(J,K,L) = F(K,L)
10              CONTINUE
                IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
                     WRITE(11,1000) J,K,(REAL(F(K,L)), L = 1,3)
                ENDIF
20         CONTINUE
           IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
                WRITE(11,*) ' '
           ENDIF
           WRITE(2,*) NDP(LND(J,1),1), NDP(LND(J,1),2)
           WRITE(2,*) ' 999990  999990 '
C     DISSPLA PROGRAM GENERATION
           IF((CHAR3.EQ.'D').OR.(CHAR3.EQ.'d')) THEN
                WRITE(20,1010) NDP(LND(J,1),1), NDP(LND(J,1),2)
                WRITE(20,1020) NDP(LND(J,2),1), NDP(LND(J,2),2)
                WRITE(20,1020) NDP(LND(J,3),1), NDP(LND(J,3),2)
                WRITE(20,1020) NDP(LND(J,1),1), NDP(LND(J,1),2)
           ENDIF
C
30    CONTINUE
C
1000  FORMAT(1X,2(I3,2X),3X,3(F8.5,2X))
1010  FORMAT(6X,'CALL STRTPT(',F8.5,',',F8.5,')')
1020  FORMAT(6X,'CALL CONNPT(',F8.5,',',F8.5,')')
C
      RETURN
      END
C
```

```
      SUBROUTINE ZERO
C     ZERO A, B, C, AND P MATRICES
      COMPLEX A(50,50), B(50,50), C(50,50), P(50,100)
      INTEGER J, K, L
C
      COMMON/BLK8/A,B,C,P
C
      DO 50, J = 1, 50
           DO 40, K = 1, 50
                A(J,K) = CMPLX(0.0,0.0)
                B(J,K) = CMPLX(0.0,0.0)
                C(J,K) = CMPLX(0.0,0.0)
40         CONTINUE
           DO 60, L = 1, 100
                P(J,L) = CMPLX(0.0,0.0)
60         CONTINUE

50    CONTINUE
      RETURN
      END
C
C
      SUBROUTINE BNDC(I,BCOND,E0,SURBC,CHAR1,ALPHA,BETA,LINE,MODE,
     CXORIGIN,YORIGIN,CHAR4)
C
C     BOUNDARY CONDITION FILL FOR A PLANE WAVE
C     E0 = FIELD STRENGTH
C     K0 = WAVE NUMBER (2*PI/WAVELENGTH)
C
C     BOUNDARY CONDITION FILL FOR A CYLINDRICAL BOUNDARY CONDITION
C     E0 = FIELD STRENGTH
C     MODE = MODE NUMBER FOR CYLINDRICAL BOUNDARY CONDITIONS
C
      COMPLEX SURBC(100), VALUE(50), BCOND(100), ALPHA, BETA, LINE(50)
      COMPLEX K, RAK
      REAL NDP(200,2), E0, KR, KI, XORIGIN, YORIGIN, RA, RB
      INTEGER I, J, NDTOP, NDBOT, NDTOT, NODES(200), PERND, BIND, MODE
      CHARACTER*1 CHAR1, CHAR4
C
      COMMON/BLK2/PERND,BIND
      COMMON/BLK3/NODES
      COMMON/BLK5/NDP
C
      IF((CHAR1.EQ.'P').OR.(CHAR1.EQ.'p')) THEN
           IF((CHAR4.EQ.'U').OR.(CHAR4.EQ.'u')) THEN
                KR = REAL(CSQRT(BETA))
                KI = ABS(AIMAG(CSQRT(BETA)))
           ELSE
                KR = 1.0
                KI = 0.0
           ENDIF
C
C     CHECK ON WHETHER WE ARE USING ER = ALPHA OR BETA
C
      IF(I.EQ.1) THEN
           BCOND(1) = E0*EXP(KI*NDP(1,2))*CMPLX(COS(KR*NDP(1,2)),
```

```
     C      SIN(KR*NDP(1,2)))
            BCOND(PERND) = E0*EXP(KI*NDP(3,2))*CMPLX(COS(KR*NDP(3,2)),
     C      SIN(KR*NDP(3,2)))
            BCOND(2) = E0*EXP(KI*NDP(7,2))*CMPLX(COS(KR*NDP(7,2)),
     C      SIN(KR*NDP(7,2)))
            VALUE(1) = E0*EXP(KI*NDP(2,2))*CMPLX(COS(KR*NDP(2,2)),
     C      SIN(KR*NDP(2,2)))
            WRITE(10,*) BCOND(1)
            WRITE(10,1000) VALUE(1)
            WRITE(10,*) ' '
            SURBC(1) = VALUE(1)
       ELSEIF(I.LE.BIND) THEN
            NDTOP = NODES(I) + NODES(PERND-I+2) - 1
            NDBOT = NODES(I+1) + NODES(PERND-I+1) - 1
            NDTOT = NDTOP + NDBOT
            BCOND(PERND-I+1) = E0*EXP(KI*NDP(NDTOP+1,2))*CMPLX(COS(KR*
     C      NDP(NDTOP+1,2)),SIN(KR*NDP(NDTOP+1,2)))
            BCOND(I+1) = E0*EXP(KI*NDP(NDTOT,2))*CMPLX(COS(KR*NDP(NDTOT,
     C      2)),SIN(KR*NDP(NDTOT,2)))
            WRITE(10,*) BCOND(PERND-I+2)
            DO 10, J = 2, NDTOP-1
                 VALUE(J) = E0*EXP(KI*NDP(J,2))*CMPLX(COS(KR*NDP(J,2)),
     C           SIN(KR*NDP(J,2)))
                 IF(J.EQ.2) THEN
                      SURBC(PERND-I+2) = VALUE(2)
                 ELSEIF(J.EQ.(NDTOP-1)) THEN
                      SURBC(I) = VALUE(NDTOP-1)
                 ENDIF
10          CONTINUE
            LINE(I) = VALUE((NDTOP+1)/2)
            WRITE(10,1000) (VALUE(J), J = 2, NDTOP-1)
            WRITE(10,*) BCOND(I)
            WRITE(10,*) ' '
C
       ELSEIF(I.EQ.BIND+1) THEN
            BCOND(I+1) = E0*EXP(KI*NDP(6,2))*CMPLX(COS(KR*NDP(6,2)),
     C      SIN(KR*NDP(6,2)))
            VALUE(2) = E0*EXP(KI*NDP(2,2))*CMPLX(COS(KR*NDP(2,2)),
     C      SIN(KR*NDP(2,2)))
            VALUE(3) = E0*EXP(KI*NDP(3,2))*CMPLX(COS(KR*NDP(3,2)),
     C      SIN(KR*NDP(3,2)))
            VALUE(4) = E0*EXP(KI*NDP(4,2))*CMPLX(COS(KR*NDP(4,2)),
     C      SIN(KR*NDP(4,2)))
            WRITE(10,*) BCOND(I+2)
            WRITE(10,1000) VALUE(2), VALUE(3), VALUE(4)
            SURBC(BIND+3) = VALUE(2)
            SURBC(BIND+1) = VALUE(4)
            LINE(I) = VALUE(3)
            WRITE(10,*) BCOND(I)
            WRITE(10,*) ' '
            VALUE(2) = E0*EXP(KI*NDP(7,2))*CMPLX(COS(KR*NDP(7,2)),
     C      SIN(KR*NDP(7,2)))
            WRITE(10,1000) VALUE(2)
            WRITE(10,*) BCOND(I+1)
            SURBC(BIND+2) = VALUE(2)
C
```

```
      ENDIF
C
      ELSEIF((CHAR1.EQ.'C').OR.(CHAR1.EQ.'c')) THEN
C
      IF(I.EQ.1) THEN
           RA = SQRT((NDP(2,1)-XORIGIN)**2+(NDP(2,2)-YORIGIN)**2)
           RB = SQRT((NDP(1,1)-XORIGIN)**2+(NDP(1,2)-YORIGIN)**2)
           WRITE(*,*) BETA
           K = CSQRT(BETA)
           RAK = RA*K
           CALL CYLBC(RA,RB,RAK,NDP(1,1),NDP(1,2),XORIGIN,YORIGIN,EO,
     C     BCOND(1),SURBC(1),K)
           CALL CYLBC(RA,RB,RAK,NDP(3,1),NDP(3,2),XORIGIN,YORIGIN,EO,
     C     BCOND(PERND),SURBC(PERND),K)
           CALL CYLBC(RA,RB,RAK,NDP(7,1),NDP(7,2),XORIGIN,YORIGIN,EO,
     C     BCOND(2),SURBC(2),K)
           WRITE(10,*) BCOND(1), SURBC(1)
           WRITE(10,*) ' '
      ELSEIF(I.LE.BIND) THEN
           NDTOP = NODES(I) + NODES(PERND-I+2) - 1
           NDBOT = NODES(I+1) + NODES(PERND-I+1) - 1
           NDTOT = NDTOP + NDBOT
           CALL CYLBC(RA,RB,RAK,NDP(NDTOP+1,1),NDP(NDTOP+1,2),XORIGIN,
     C     YORIGIN,EO,BCOND(PERND-I+1),SURBC(PERND-I+1),K)
           CALL CYLBC(RA,RB,RAK,NDP(NDTOT,1),NDP(NDTOT,2),XORIGIN,
     C     YORIGIN,EO,BCOND(I+1),SURBC(I+1),K)
           WRITE(10,*) BCOND(PERND-I+2), SURBC(PERND-I+2)
           WRITE(10,*) BCOND(I), SURBC(I)
           WRITE(10,*) ' '
      ELSEIF(I.EQ.BIND+1) THEN
           WRITE(10,*) BCOND(I+2), SURBC(I+2)
           WRITE(10,*) BCOND(I), SURBC(I)
           WRITE(10,*) ' '
           CALL CYLBC(RA,RB,RAK,NDP(6,1),NDP(6,2),XORIGIN,
     C     YORIGIN,EO,BCOND(I+1),SURBC(I+1),K)
           WRITE(10,*) BCOND(I+1), SURBC(I+1)
           WRITE(10,*) ' '
      ENDIF
C
      ELSE
           RETURN
      ENDIF
C
1000  FORMAT(1X,50(E14.8,2X,E14.8,2X))
C
      RETURN
      END
C
C
      SUBROUTINE CYLBC(RA,RB,RAK,X,Y,XORIGIN,YORIGIN,EO,BC,PSI,K)
C
      COMPLEX SQRTM1, JORB, J1RB, JORAK, J1RAK, HORA, H1RA, HORB, H1RB
      COMPLEX DELTAN, AN, PSI, JORA, J1RA, K, RAK, BC
      REAL PI, XORIGIN, YORIGIN, X, Y, RA, RB, EO, PHI
C
      PI = 4.0*ATAN(1.0)
```

```
      SQRTM1 = CMPLX(0.0,1.0)
      PHI = ATAN2(X - XORIGIN, Y - YORIGIN)
      CALL BES1(CMPLX(RA,0.0),J0RA,J1RA)
      CALL BES1(CMPLX(RB,0.0),J0RB,J1RB)
      CALL BES1(RAK,J0RAK,J1RAK)
      CALL HAN1(CMPLX(RA,0.0),H0RA,H1RA)
      CALL HAN1(CMPLX(RB,0.0),H0RB,H1RB)
C
      DELTAN = J1RB*(J1RAK*(H0RA-H1RA/RA)-K*(J0RAK-J1RAK/RAK)*H1RA)-
     CH1RB*(J1RAK*(J0RA-J1RA/RA)-K*(J0RAK-J1RAK/RAK)*J1RA)
      AN = -2.0*SQRTM1/(PI*RA*DELTAN)
      BC = E0*COS(PHI)
      PSI = AN*J1RAK*BC
C
      RETURN
      END
C
C
      SUBROUTINE BES1(Z,J0,J1)
C
C       Computing Bessel Functions for n = 0, 1 with
C       Complex Argument Z.  Direct Power Series Method for
C       CABS(Z) .LE. 6  and Hankel's Asymptotic Formula for
C       CABS(Z) .GT. 6.
C       Written 11/5/87 by M.A. Morgan
C
      INTEGER M,M2
      REAL C(34),DM,F(34),G0,P(34),Pi,P2
      COMPLEX Z,Z2,Z3,Z4,J0,J1,AM,CL,P0,P1,Q0,Q1,C0,C1,S0,S1
      Pi=3.1415927
      P2=2.0/PI
      IF(CABS(Z).LE.6.0) THEN
C
C   Utilizing the Direct Power Series Method
C
            G0= 1.781072
            Z2=0.5*Z
            CL=CLOG(G0*Z2)
C
C       Computing F(m) = m !  and P(m) = 1 + 1/2 + 1/3 + ....+ 1/m
C
            F(1)=1.0
            P(1)=1.0
            DO 11 M=2,34
                  F(M)=M*F(M-1)
                  P(M)=P(M-1)+1.0/M
11          CONTINUE
C
C       Computing Power Series Coefficients
C
            DM=-1.0
            DO 22 M=1,34
                  C(M)=DM/(F(M)*F(M))
                  DM=-DM
22          CONTINUE
C
```

```
C       Computing J0 and J1
C
            J0=(1.,0.)
            J1=(0.,0.)
            M=0
33          M=M+1
            M2=2*M
            AM=C(M)*(Z2**M2)
            J0=J0+AM
            J1=J1-M*AM
            IF((CABS(AM).GT.1.0E-10).AND.(M.LT.34)) GO TO 33
            J1=J1/Z2
            return
      ELSE
C
C   Hankel' Asymptotic Formula (Abram. & Stegun p. 364)
C
            Z2=Z*Z
            Z3=Z*Z2
            Z4=Z*Z3
            P0=1.0-.0703125/Z2+.1121521/Z4
            Q0=-.125/Z+.0732422/Z3
            P1=1.0+.1171875/Z2-.1441956/Z4
            Q1=.375/Z-.10253906/Z3
            C0=CCOS(Z-.25*PI)
            S0=CSIN(Z-.25*PI)
            C1=CCOS(Z-.75*PI)
            S1=CSIN(Z-.75*PI)
            AM=CSQRT(P2/Z)
            J0=AM*(P0*C0-Q0*S0)
            J1=AM*(P1*C1-Q1*S1)
      ENDIF
      RETURN
      END

        SUBROUTINE HAN1(Z,H0,H1)
C
C       Computing Hankel Functions for n = 0, 1 with
C       Complex Argument, Z.   Direct Power Series Method for
C       CABS(Z) .LE. 5  and Hankel's Asymptotic Formula for
C       CABS(Z) .GT. 5.   Written 11/6/87 by M.A. Morgan
C
      INTEGER M,M2
      REAL C(34),DM,F(34),G0,P(34),Pi,P2
      COMPLEX Z,Z2,Z3,Z4,J0,J1,Y0,Y1,AM,CL,P0,P1,Q0,Q1
      COMPLEX E0,E1,X0,X1,H0,H1,j
      PI=3.1415927
      P2=2.0/PI
      j=(0.,1.)
      IF(CABS(Z).LE.5.0) THEN
C
C           Direct Power Series Method
C
            G0= 1.78072
            Z2=0.5*Z
```

```
      CL=CLOG(G0*Z2)
C
C   Computing F(m) = m !  and P(m) = 1 + 1/2 + 1/3 + ....+ 1/m
C
      F(1)=1.0
      P(1)=1.0
      DO 11 M=2,34
            F(M)=M*F(M-1)
            P(M)=P(M-1)+1.0/M
11    CONTINUE
C
C   Computing Power Series Coefficients
C
      DM=-1.0
      DO 22 M=1,34
            C(M)=DM/(F(M)*F(M))
            DM=-DM
22    CONTINUE
C
C   Computing J0 and J1
C
      J0=(1.,0.)
      J1=(0.,0.)
      M=0
33    M=M+1
      M2=2*M
      AM=C(M)*(Z2**M2)
      J0=J0+AM
      J1=J1-M*AM
      IF((CABS(AM).GT.1.0E-10).AND.(M.LT.34)) GO TO 33
      J1=J1/Z2
C
C   Computing Y0 and Y1
C
      M=0
      Y0=CL*J0
      Y1=Z2*CL*J1-0.5*J0
44    M=M+1
      M2=2*M
      AM=C(M)*P(M)*(Z2**M2)
      Y0=Y0-AM
      Y1=Y1+M*AM
      IF((CABS(AM).GT.1.0E-10).AND.(M.LT.34)) GO TO 44
      Y0=P2*Y0
      Y1=P2*Y1/Z2
      H0=J0-j*Y0
      H1=J1-j*Y1
      RETURN
    ELSE
C
C   Hankel' Asymptotic Formula (Abram. & Stegun p. 364
C
      Z2=Z*Z
      Z3=Z*Z2
      Z4=Z*Z3
      P0=1.0-.0703125/Z2+.1121521/Z4
```

```
         Q0=-.125/Z+.0732422/Z3
         P1=1.0+.1171875/Z2-.1441956/Z4
         Q1=.375/Z-.10253906/Z3
         X0=(Z-.25*PI)
         X1=(Z-.75*PI)
         E0=CEXP(-j*X0)
         E1=CEXP(-j*X1)
         AM=CSQRT(P2/Z)
         H0=AM*(P0-j*Q0)*E0
         H1=AM*(P1-j*Q1)*E1
      ENDIF
      RETURN
      END
      SUBROUTINE MARCH(I,IMX,NBMX,NA,NB,NC,INORM,SVEC)
C     THIS ROUTINE PERFORMS THE RICCATI TRANSFORM FIRST
C     SWEEP, GENERATING AND STORING ON DISK 1 RMAT
C     AND SVEC (FOR EACH MODE) AT EACH FORWARD STEP.
C
      COMPLEX RMAT(50,50),SVEC(50,100)
      COMPLEX A(50,50),B(50,50),C(50,50),P(50,100)
      COMPLEX D(50),SUM,DET
      REAL COND,DMAG
      INTEGER I,J,K,L,NA,NB,NC,PERND,BIND,IMX,INORM
      INTEGER NBMX
      CHARACTER*1 CHAR2, CHAR3, CHAR4
C
      COMMON/BLK2/PERND,BIND
      COMMON/BLK8/A,B,C,P
      COMMON/BLK9/CHAR2, CHAR3, CHAR4
C
C     LOADING THEN INVERTING (B+A*RMAT)
C     USING MINIMUM MEMORY SINGLE MATRIX TECHNIQUE
C     DATE 1/29/80 FOR THIS CHANGE
C
C     SKIPPING FIRST A*R (WHEN A = 0, ZERO R MATRIX)
      IF(I.EQ.1) THEN
         DO 10, J = 1, NB
            DO 10, K = 1, NB
               RMAT(J,K) = (0.,0.)
10       CONTINUE
C     RMAT = A*R
      ELSE
C
      IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
         WRITE(19,*) ' OLD R MATRIX'
         DO 11, J = 1, 5
            WRITE(19,1000) (REAL(RMAT(J,K)), K = 1, 5)
11       CONTINUE
         WRITE(19,*) ' '
         WRITE(19,*) ' A MATRIX'
         DO 12, J = 1, 5
            WRITE(19,1000) (REAL(A(J,K)), K = 1, 5)
12       CONTINUE
         WRITE(19,*) ' '
      ENDIF
C
```

```
        DO 30, K = 1, NB
             DO 20, J = 1, NB
                  D(J) = (0.,0.)
                  DO 20, L = 1, NA
                       D(J) = D(J) + A(J,L)*RMAT(L,K)
20           CONTINUE
             DO 30, J = 1, NB
                  RMAT(J,K) = D(J)
30      CONTINUE
C
   IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
        WRITE(19,*) ' NEW R MATRIX'
        DO 13, J = 1, 5
             WRITE(19,1000) (REAL(RMAT(J,K)), K = 1, 5)
13      CONTINUE
        WRITE(19,*) ' '
   ENDIF
C
   ENDIF
C
   IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
        WRITE(19,*) ' B MATRIX'
        DO 14, J = 1, 5
             WRITE(19,1000) (REAL(B(J,K)), K = 1, 5)
14      CONTINUE
        WRITE(19,*) ' '
   ENDIF
C
C  RMAT = B + RMAT
   DO 40, J = 1, NB
        DO 40, K = 1, NB
             RMAT(J,K) = RMAT(J,K) + B(J,K)
40 CONTINUE
C
   IF((CHAR2.EQ.'I').OR.(CHAR2.EQ.'i')) THEN
        WRITE(19,*) ' NEWEST R MATRIX'
        DO 15, J = 1, NB
             WRITE(9,1000) (REAL(RMAT(J,K)), K = 1, NB)
             WRITE(19,1000) (REAL(RMAT(J,K)), K = 1, NB)
15      CONTINUE
        WRITE(9,*) ' '
        WRITE(19,*) ' '
   ENDIF
C  INVERTING THE MATRIX (B + A*R)
   CALL CSMINV(RMAT,NBMX,NB,DET,COND,INORM)
C
   DMAG = CABS(DET)
   WRITE(*,*) I,NBMX,NB,DMAG,COND
C
C  COMPUTING THE NEW S-VECTORS
C
C  SKIPPING FIRST A*S (A = 0)
        IF (I.EQ.1) THEN
             CONTINUE
        ELSE
             DO 70, K = 1, PERND
```

```
                  DO 60, J = 1, NB
                       D(J) = (0.0,0.0)
                       DO 60, L = 1, NA
                            D(J) = D(J) + A(J,L)*SVEC(L,K)
60                CONTINUE
                  DO 70, J = 1, NB
                       P(J,K) = P(J,K) - D(J)
70             CONTINUE
          ENDIF
C     FINAL SVEC MULTIPLICATION
               DO 100, K = 1, PERND
                  DO 90, J = 1, NB
                       D(J) = (0.0,0.0)
                       DO 90, L = 1, NB
                            D(J) = D(J) + RMAT(J,L)*P(L,K)
90                CONTINUE
                  DO 100, J = 1, NB
                       SVEC(J,K) = D(J)
100            CONTINUE
C
C     STORING I+1 SVEC ON DISK 7
      DO 110 J = 1, NB
           WRITE(7) (SVEC(J,K), K = 1, PERND)
110   CONTINUE
C
C     FINAL RMAT MULTIPLICATION
      IF(I.EQ.IMX) RETURN
      DO 130, J = 1, NB
           DO 120, K = 1, NC
                D(K) = (0.0,0.0)
                DO 120, L = 1, NB
                     D(K) = D(K) - RMAT(J,L)*C(L,K)
120        CONTINUE
           DO 130, K = 1, NC
                RMAT(J,K) = D(K)
130   CONTINUE
C
C     STORING I+1 RMAT ON DISK 7
      DO 140, J = 1, NB
      WRITE(7) (RMAT(J,K), K = 1, NC)
140   CONTINUE
C
1000  FORMAT(10(E9.3,1X))
C
      RETURN
      END
C
C
      SUBROUTINE SWEEP(IMX,NABC,SURBC,LINE,CHAR1,U,BCOND,PSI,ANS,CHAR5)
C     THIS ROUTINE PERFORMS THE RICCATI TRANSFORM BACKSWEEP
C     FROM I=IMX TO I=1, RECALLING RMAT AND
C     SVEC FROM DISK 7 AT EACH BACKSTEP TO FORM
C     THE NODE VECTORS PSI, THEN STORING THESE ON
C     DISK 8 FOR EACH APPLIED DIRICHLET B.C.
      COMPLEX RMAT(50,100),PSI(50,100),SVEC(50,100),D(100),TEMP
      COMPLEX SURBC(100),ANS(100),LINE(50),ANSB(50),U(100,100)
```

```
      COMPLEX BCOND(100)
      REAL ERRORP,ERRORD,TERRN,TERRD,ATSERR
      INTEGER I,J,K,L,NABC(100,3),IMX,PERND,BIND
      CHARACTER*1 CHAR1, CHAR5
C
      COMMON/BLK2/PERND,BIND
C
      IF((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m')) THEN
           RETURN
      ENDIF
C
C     INITIAL DISK READ AT IMX (R = 0, NOT WRITTEN, => S IS READ FIRST)
      WRITE(*,*) ' '
      DO 90, I = IMX, 1, -1
           WRITE(*,1050) (IMX-I+1), IMX
           IF(I.EQ.IMX) THEN
                DO 10, J = NABC(I,2), 1, -1
                     BACKSPACE 7
                     READ(7) (PSI(J,K), K = 1, PERND)
                     BACKSPACE 7
10              CONTINUE
                DO 20, J = 1, NABC(I,2)
                     DO 15, K = 1, PERND
                          U(IMX,K) = PSI(J,K)
15                   CONTINUE
20              CONTINUE
C          SUBSEQUENT DISK READS
           ELSE
C               READ R MATRIX
                DO 30, J = NABC(I,2), 1, -1
                     BACKSPACE 7
                     READ(7) (RMAT(J,K), K = 1, NABC(I,3))
                     BACKSPACE 7
30              CONTINUE
C               READ S VECTOR
                DO 40, J = NABC(I,2), 1, -1
                     BACKSPACE 7
                     READ(7) (SVEC(J,K), K = 1, PERND)
                     BACKSPACE 7
40              CONTINUE
C               MULTIPLY RMAT = RMAT*PSI
                DO 60, J = 1, NABC(I,2)
                     DO 50, K = 1, PERND
                          D(K) = (0.0,0.0)
                          DO 50, L = 1, NABC(I,3)
                               D(K) = D(K) + RMAT(J,L)*PSI(L,K)
50                   CONTINUE
                     DO 60, K = 1, PERND
                          RMAT(J,K) = D(K)
60              CONTINUE
C               PSI = RMAT + SVEC
                DO 70, J = 1, NABC(I,2)
                     DO 70, K = 1, PERND
                          PSI(J,K) = RMAT(J,K) + SVEC(J,K)
70              CONTINUE
                ANSB(I) = (0.0,0.0)
```

```
                DO 80, J = 1, NABC(I,2)
                     DO 75, K = 1, PERND
                          IF((J.EQ.NABC(I,2)).OR.(I.EQ.1)) THEN
                               U(I,K) = PSI(J,K)
                          ELSEIF(J.EQ.1) THEN
                               U(2*IMX-I,K) = PSI(J,K)
                          ELSEIF(J.EQ.(NABC(I,2)+1)/2) THEN
                               ANSB(I) = ANSB(I) + PSI(J,K)*BCOND(K)
                          ENDIF
75                   CONTINUE
80              CONTINUE
          ENDIF
90    CONTINUE
      WRITE(8,*) ' '
C
      DO 98, J = 1, PERND
           ANS(J) = (0.0,0.0)
           DO 94, L = 1, PERND
                ANS(J) = ANS(J) + U(J,L)*BCOND(L)
94         CONTINUE
98    CONTINUE
C
      TERRN = 0.0
      TERRD = 0.0
      DO 100, I = 1, PERND
           ERRORP = (CABS(ANS(I) - SURBC(I)))**2
           ERRORD = (CABS(SURBC(I)))**2
           TERRN = TERRN + ERRORP
           TERRD = TERRD + ERRORD
           WRITE(13,1020) I, BCOND(I), SURBC(I), ANS(I), ERRORP
100   CONTINUE
      WRITE(13,*) ' '
      ATSERR = (TERRN/TERRD)**0.5
      WRITE(13,1030) ATSERR
      WRITE(*,*) 'RMS ERROR (FOR THE PERIMETER)     = ',ATSERR
      WRITE(13,*) ' '
C
      IF((CHAR1.EQ.'C').OR.(CHAR1.EQ.'c')) THEN
           RETURN
      ELSE
           TERRN = 0.0
           TERRD = 0.0
           DO 110, I = 2, BIND+1
                ERRORP = (CABS(ANSB(I) - LINE(I)))**2
                ERRORD = (CABS(LINE(I)))**2
                TERRN = TERRN + ERRORP
                TERRD = TERRD + ERRORD
                WRITE(13,1025) (I-1), LINE(I), ANSB(I), ERRORP
110        CONTINUE
           WRITE(13,*) ' '
           ATSERR = (TERRN/TERRD)**0.5
           WRITE(13,1040) ATSERR
           WRITE(*,*) 'RMS ERRROR (FOR BISECTION SEGMENT) = ',ATSERR
      ENDIF
C
      DO 120, I = 1, PERND
```

```
      WRITE(30,1060) (U(I,J), J = 1, PERND)
120   CONTINUE
C
1020  FORMAT(1X,I3,2X,3(E14.8,1X,E14.8,3X),F10.6)
1025  FORMAT(1X,I3,2X,4(E14.8,2X),F10.6)
1030  FORMAT(1X,' RMS ERROR (FOR THE PERIMETER)     = ',F12.6)
1040  FORMAT(1X,' RMS ERROR (FOR BISECTION SEGMENT) = ',F12.6)
1050  FORMAT(1X,I3,' TOTAL BACKSWEEP ROWS OUT OF ',I3,' COMPLETED')
1060  FORMAT(1X,100(E8.2,E8.2,2X))
C
      RETURN
      END
C
CC
C
      SUBROUTINE CSMINV(A,NDIM,N,DETERM,COND,INORM)
C
C     INORM - FLAG TO NORMALIZE COLUMNS AND ROWS OF MATRIX A
C     MATRIX NORMALIZATION BY M.A. MORGAN
C     APRIL 24,1978
C
C     A      - MATRIX TO INVERT              - INPUT/OUTPUT
C     NDIM   -                               - INPUT
C     N      -                               - INPUT
C     DETERM - DETERMINATE OF A              - OUTPUT
C     COND   - CONDITION NUMBER OF A         - OUTPUT
C     INORM  - INTEGER NORMALIZATION FLAG    - INPUT
C
C
      COMPLEX A(50,50),PIVOT(50),AMAX,T,SWAP,DETERM,U
      INTEGER I,J,K,L,IPIVOT(50),INDEX(50,2),IROW,ICOLUM,L1,JROW
      INTEGER JCOLUM,N,INORM
      REAL TEMP,ALPHA(50),COL(50),ROW(50),AJK,SUMAXA,SUMROW,SUMAXI
C
      IF(NDIM.GT.50) THEN
           WRITE(*,*) ' ERROR IN INVERTION CALL... DIMENSION > 50 '
           STOP
      ENDIF
      IF(N.GT.NDIM) THEN
           WRITE(*,*) ' ERROR IN INVERTION CALL... N > MAX DIM. '
           STOP
      ENDIF
      IF(INORM.NE.1) GO TO 7
      DO 3 K = 1, N
           COL(K) = 0.0
           DO 1 J = 1,N
                AJK = CABS(A(J,K))
                IF(AJK.GT.COL(K)) COL(K) = AJK
1          CONTINUE
           DO 2 J = 1, N
2          A(J,K) = A(J,K)/COL(K)
3     CONTINUE
C     ROW NORMALIZING
      DO 6 J = 1, N
           ROW(J) = 0.0
           DO 4 K = 1, N
```

```
                AJK = CABS(A(J,K))
                IF(AJK.GT.ROW(J)) ROW(J) = AJK
4           CONTINUE
            DO 5 K = 1, N
5               A(J,K) = A(J,K)/ROW(J)
6       CONTINUE
7       CONTINUE
        DETERM = CMPLX(1.0,0.0)
        SUMAXA = 0.0
        DO 20 J = 1, N
            ALPHA(J) = 0.0
            SUMROW = 0.0
            DO 10 I = 1, N
                ALPHA(J) = ALPHA(J) + A(J,I)* CONJG(A(J,I))
10          SUMROW = SUMROW + CABS(A(J,I))
            ALPHA(J) = SQRT(ALPHA(J))
            IF(SUMROW.GT.SUMAXA) SUMAXA = SUMROW
20      IPIVOT(J) = 0
        DO 600 I = 1, N
C
C
            AMAX = CMPLX(0.0,0.0)
            DO 105 J = 1, N
                IF (IPIVOT(J)-1) 60, 105, 60
60              DO 100 K = 1, N
                    IF (IPIVOT(K)-1) 80, 100, 740
80                  TEMP = AMAX*CONJG(AMAX) - A(J,K)*CONJG(A(J,K))
                    IF(TEMP)85,85,100
85                      IROW = J
                        ICOLUM = K
                        AMAX = A(J,K)
100             CONTINUE
105         CONTINUE
            IPIVOT(ICOLUM) = IPIVOT(ICOLUM) + 1
C
C
            IF (IROW-ICOLUM) 140, 260, 140
140             DETERM = -DETERM
                DO 200 L = 1, N
                    SWAP = A(IROW,L)
                    A(IROW,L) = A(ICOLUM,L)
200             A(ICOLUM,L) = SWAP
                SWAP = ALPHA(IROW)
                ALPHA(IROW) = ALPHA(ICOLUM)
                ALPHA(ICOLUM) = SWAP
260             INDEX(I,1) = IROW
                INDEX(I,2) = ICOLUM
                PIVOT(I) = A(ICOLUM,ICOLUM)
                U = PIVOT(I)
                DETERM = DETERM*U
                DETERM = DETERM/ALPHA(ICOLUM)
                TEMP = PIVOT(I)*CONJG(PIVOT(I))
                IF(TEMP)330,720,330
C
C
330             A(ICOLUM,ICOLUM) = CMPLX(1.0,0.0)
```

```
               DO 350 L = 1, N
                    U = PIVOT(I)
350            A(ICOLUM,L) = A(ICOLUM,L)/U
C
C
380            DO 550 L1 = 1, N
                    IF(L1-ICOLUM) 400, 550, 400
400                 T = A(L1,ICOLUM)
                    A(L1,ICOLUM) = CMPLX(0.0,0.0)
                    DO 450 L = 1, N
                         U = A(ICOLUM,L)
450                 A(L1,L) = A(L1,L) - U*T
550       CONTINUE
600  CONTINUE
C
C
620  DO 710 I = 1, N
          L = N + 1 - I
          IF (INDEX(L,1) - INDEX(L,2)) 630, 710, 630
630            JROW = INDEX(L,1)
               JCOLUM = INDEX(L,2)
               DO 705 K = 1, N
                    SWAP = A(K,JROW)
                    A(K,JROW) = A(K,JCOLUM)
                    A(K,JCOLUM) = SWAP
705            CONTINUE
710  CONTINUE
     SUMAXI = 0.0
     DO 910 I = 1, N
          SUMROW = 0.0
          DO 900 J = 1, N
900       SUMROW = SUMROW + CABS(A(I,J))
          IF(SUMROW.GT.SUMAXI) SUMAXI = SUMROW
910  CONTINUE
     COND = SUMAXA*SUMAXI
     IF(INORM.NE.1) GO TO 955
          DO 950 K = 1, N
               DO 950 J = 1, N
950       A(J,K) = A(J,K)/(ROW(K)*COL(J))
955  CONTINUE
     RETURN
720  WRITE(*,730)
730  FORMAT( ' MATRIX IS SINGULAR')
740  RETURN
     END

     SUBROUTINE SAVE(BCOND,ANS,U,OFFSET,PERND,CHAR5,DPER,K,XORG,YORG,
    CNRES,MRES,LBIAS,GBIAS,MAXD)
C
C    THIS SUBROUTINE SAVES THE ESSENCE OF THE FINITE ELEMENT PROBLEM
C    TO A DATA FILE CALLED " F3.DAT ".  tHIS DATA IS NECESSARY TO
C    SOLVE THE FIELD FEEDBACK FORMULATION, (F3).
C
     COMPLEX BCOND(100),ANS(100),U(100,100)
     REAL OFFSET,MESH(0:200,5),DPER,K,XORG,YORG,MRES,MAXD
     INTEGER PERND,I,J,NRES,LBIAS,GBIAS
```

```
      CHARACTER*1 CHAR5
C
      COMMON/BLK1/MESH
C
      IF((CHAR5.EQ.'M').OR.(CHAR5.EQ.'m')) THEN
           RETURN
      ENDIF
C
      WRITE(40,*) PERND
      WRITE(40,*) OFFSET
      WRITE(40,*) DPER
      WRITE(40,*) K
      WRITE(40,*) XORG
      WRITE(40,*) YORG
      WRITE(40,*) NRES
      WRITE(40,*) MRES
      WRITE(40,*) LBIAS
      WRITE(40,*) GBIAS
      WRITE(40,*) MAXD
C
      DO 10, I = 1, 4
           DO 10, J = 1, PERND
           WRITE(40,*) MESH(J,I)
10    CONTINUE
C
      DO 20, J = 1, PERND
           WRITE(40,*) BCOND(J)
20    CONTINUE
C
      DO 30, J = 1, PERND
           WRITE(40,*) ANS(J)
30    CONTINUE
C
      DO 40, I = 1, PERND
           DO 40, J = 1, PERND
                WRITE(40,*) U(I,J)
40    CONTINUE
C
      RETURN
      END
C
C
```

APPENDIX D. VARINT CONVERGENCE PROGRAM

```
C
C       TEST OF VARINT CONVERGENCE
C
      COMPLEX F(3,3), ALPHA, BETA, EXACT, CENTER, LEFT, RIGHT, TOP
      COMPLEX BOTTOM, SUM, CALC
      REAL X(3), Y(3), D, AREA, KR, PI, ERROR
      INTEGER I
      OPEN (UNIT = 1, FILE = 'C: MSFORT TEST.DAT', STATUS = 'UNKNOWN')
      ALPHA = (1.0,0.0)
      BETA = (1.0,0.0)
      PI = 4.0*ATAN(1.0)
C
      DO 5, I = 1, 100
           D = 2.0*PI*FLOAT(I)/100.0
           KR = REAL(CSQRT(BETA))
           X(1) = 0.0
           Y(1) = 0.0
           X(2) = D
           Y(2) = 0.0
           X(3) = 0.0
           Y(3) = D
C
           CALL VARINT(X,Y,F,ALPHA,BETA,AREA)
C
           EXACT = CMPLX(COS(KR*0.0),SIN(KR*0.0))
           CENTER = 4.0*F(1,1)
           RIGHT = -2.0*F(1,2)*CMPLX(COS(KR*Y(2)),SIN(KR*Y(2)))
           TOP = -2.0*F(1,2)*CMPLX(COS(KR*Y(3)),SIN(KR*Y(3)))
           LEFT = -2.0*F(1,2)*CMPLX(COS(KR*Y(2)),SIN(KR*Y(2)))
           BOTTOM = -2.0*F(1,2)*CMPLX(COS(-KR*Y(3)),SIN(-KR*Y(3)))
           SUM = TOP + BOTTOM + LEFT + RIGHT
           CALC = SUM/CENTER
           ERROR = (CALC-EXACT)/EXACT
           WRITE(1,1000) I,D,EXACT,CALC,ERROR
5     CONTINUE
C
      CLOSE(1)
C
1000  FORMAT(1X,I3,1X,F8.5,1X,2(F8.5,1X,F8.5,1X),E13.6)
C
      STOP
      END
C
C
      SUBROUTINE VARINT(X,Y,F,ALPHA,BETA,AREA)
C
C     GENERATING VARIATIONAL FINITE ELEMENT AREA INTEGRATIONS OF THE
C     LINEAR BASIS FUNCTION LAGRANGIAN FOR THE HELMHOLTZ EQUATION.
C     THESE ARE RETURNED IN F(3,3). X(3) AND Y(3) ARE THE WAVENUMBER
C     NORMALIZED CARTESIAN COORDINATES OF THE TRIANGLE VERTICES.
C     X = Ko*x, Y = Ko*y    ALPHA AND BETA ARE COMPLEX
```

```
C     MATERIAL PARAMETERS WITHIN THE ELEMENT.
C
C       FOR TM INCIDENCE: ALPHA = 1/ur; BETA = er
C       FOR TE INCIDENCE: ALPHA = 1/er; BETA = ur
C
C     OUTPUTS :    F(3,3) - FINITE ELEMENT AREA INTEGRATION
C                  AREA   - AREA OF A TRIANGLE
C
      COMPLEX ALPHA, BETA, B12, F(3,3)
      REAL X(3), Y(3), T(3,3), AREA, DET
      INTEGER L, K
C
      DET  = ABS(X(2)*Y(3) + X(3)*Y(1) + X(1)*Y(2) - X(3)*Y(2) -
     CX(1)*Y(3) - X(2)*Y(1))
      AREA = ABS(0.5*DET)
      B12 = BETA/12.
      T(1,1) = (Y(2) - Y(3))/DET
      T(1,2) = (Y(3) - Y(1))/DET
      T(1,3) = (Y(1) - Y(2))/DET
      T(2,1) = (X(3) - X(2))/DET
      T(2,2) = (X(1) - X(3))/DET
      T(2,3) = (X(2) - X(1))/DET
      T(3,1) = (X(2)*Y(3) - X(3)*Y(2))/DET
      T(3,2) = (X(3)*Y(1) - X(1)*Y(3))/DET
      T(3,3) = (X(1)*Y(2) - X(2)*Y(1))/DET
      DO 10, K = 1, 3
           DO 10 L = 1, 3
                F(K,L) = ALPHA*(T(1,K)*T(1,L) + T(2,K)*T(2,L)) - B12
                IF(K.EQ.L) F(K,L) = F(K,L) - B12
10    F(K,L) = AREA*F(K,L)
C
      RETURN
      END
C
C
```

APPENDIX E. FIELD FEEDBACK PROGRAM

```
C
C     FIELD FEEDBACK FORMULATION PROGRAM
C     WRITTEN BY T.B. WELCH
C
C     w/ PROGRAMMING IDEAS FROM PROF M.A. MORGAN
C
C     BCOND      - BOUNDARY CONDITIONS
C     OFFSET     - OFFSET IN WAVELENGTHS (PERIMETER TO BOUNDARY)
C     ANS        - CALCULATED PSI VALUES ON PERIMETER
C     DPER       - DESIRED PERCENT ERROR SCALE FACTOR FOR GREEN'S
C                  FUNCTION INTEGRAL PATCH STEPPING
C     LBIAS      - BIAS THAT IS ADDED TO THE GFI STEP FOR < 1.0
C     GBIAS      - BIAS THAT IS ADDED TO THE GFI STEP FOR > 1.0
C     MAXD       - MAXIMUM DISTANCE BEYOND WHICH NO CONTRIBUTION IS MADE
C                  TO THE GFI
C     K          - WAVENUMBER
C     XORG       - X ORIGIN
C     YORG       - Y ORIGIN
C     NRES       - NUMBER OF EVENLY SPACED POINTS DESIRED FOR THE FAR
C                  FIELD CALCULATIONS (360/NRES = ANGULAR RESOLUTION)
C     U          - MATRIX THAT RELATES EACH BOUNDARY NODE VALUE TO THE
C                  UNKNOWN PERIMETER NODE VALUE.  MULTIPLY U BY A DRIVING
C                  VECTOR (ON BOUNDARY) TO FIND PERIMETER VALUES.
C     PERND      - NUMBER OF PERIMETER NODES
C     MESH       - GEOMETRY ARRAY CONTAINING:
C                     1 - X POSITION OF PERIMETER NODES
C                     2 - Y POSITION OF PERIMETER NODES
C                     3 - X UNIT NORMAL OF PERIMETER NODES
C                     4 - Y UNIT NORMAL OF PERIMETER NODES
C                     5 - X POSITION OF GEOMETRIC CONTOUR NODES
C                     6 - Y POSITION OF GEOMETRIC CONTOUR NODES
C                     7 - X POSITION OF BOUNDARY NODES
C                     8 - Y POSITION OF BOUNDARY NODES
C     T          - MATRIX THAT RELATES PERIMETER VALUES BACK OUT
C                  TO THE BOUNDARY VIA A GREEN'S FUNCTION INTEGRAL
C     CNVEC      - VECTOR OF SCATTERED FIELD BACK ONTO THE BOUNDARY
C     MRES       - MESH RESOLUTION
C
C
      COMPLEX BCOND(100),ANS(100),U(100,100),T(100,100),CNVEC(100)
      REAL OFFSET,MESH(100,8),DPER,K,XORG,YORG,MRES,MAXD
      INTEGER PERND,I,J,NRES,LBIAS,GBIAS
C
      OPEN (UNIT = 40, FILE = 'C: MSFORT F3.DAT',STATUS='UNKNOWN')
      OPEN (UNIT = 50, FILE = 'C: MSFORT FFPAT.DAT',STATUS='UNKNOWN')
C
      CALL INPUT(BCOND,ANS,U,OFFSET,PERND,MESH,DPER,K,NRES,XORG,YORG,
     CMRES,LBIAS,GBIAS,MAXD)
      CALL TMAT(U,PERND,MESH,T,OFFSET,DPER,BCOND,MRES,LBIAS,GBIAS,MAXD)
      CALL CNSOLV(T,BCOND,CNVEC,PERND)
```

```
      CALL FFLD(CNVEC,PERND,MESH,U,OFFSET,K,NRES,XORG,YORG)
C
      CLOSE(40)
      CLOSE(50)
C
      STOP
      END
C
C
      SUBROUTINE INPUT(BCOND,ANS,U,OFFSET,PERND,MESH,DPER,K,NRES,XORG,
     CYORG,MRES,LBIAS,GBIAS,MAXD)
C
C     THIS SUBROUTINE READS THE FINITE ELEMENT PROBLEM DATA FROM
C     THE DATA FILE CALLED  " F3.DAT ".  THIS DATA IS NECESSARY TO
C     SOLVE THE FIELD FEEDBACK FORMULATION, (F3).
C
      COMPLEX BCOND(100),ANS(100),U(100,100)
      REAL OFFSET,MESH(100,8),DPER,K,XORG,YORG,MRES,MAXD
      INTEGER PERND,I,J,NRES,LBIAS,GBIAS
C
      WRITE(*,*) ' READING INPUT DATA '
C
      READ(40,*) PERND
      READ(40,*) OFFSET
      READ(40,*) DPER
      READ(40,*) K
      READ(40,*) XORG
      READ(40,*) YORG
      READ(40,*) NRES
      READ(40,*) MRES
      READ(40,*) LBIAS
      READ(40,*) GBIAS
      READ(40,*) MAXD
C
      WRITE(6,*) ' NUMBER OF PERIMETER NODES           = ',PERND
      WRITE(6,*) ' BOUNDARY CONTOUR OFFSET             = ',OFFSET
      WRITE(6,*) ' DESIRED GFI SCALE FACTOR            = ',DPER
      WRITE(6,*) ' GFI STEP BIAS FOR < 1.0             = ',LBIAS
      WRITE(6,*) ' GFI STEP BIAS FOR > 1.0             = ',GBIAS
      WRITE(6,*) ' MAX DIST > NO CONTRIBUTION TO GFI  = ',MAXD
      WRITE(6,*) ' WAVENUMBER                          = ',K
      WRITE(6,*) ' X ORIGIN                            = ',XORG
      WRITE(6,*) ' Y ORIGIN                            = ',YORG
      WRITE(6,*) ' NUMBER OF NODES FOR SIGMA           = ',NRES
      WRITE(6,*) ' REQUESTED MESH RESOLUTION           = ',MRES
C
      DO 10, I = 1, 4
           DO 10, J = 1, PERND
                READ(40,*) MESH(J,I)
10    CONTINUE
C
      DO 20, J = 1, PERND
           READ(40,*) BCOND(J)
20    CONTINUE
C
      DO 30, J = 1, PERND
```

```
          READ(40,*) ANS(J)
30    CONTINUE
C
      DO 40, I = 1, PERND
          DO 40, J = 1, PERND
              READ(40,*) U(I,J)
40    CONTINUE
C
      DO 50, I = 1, PERND
          MESH(I,5) = MESH(I,1) + MESH(I,3)*OFFSET/2.0
          MESH(I,6) = MESH(I,2) + MESH(I,4)*OFFSET/2.0
          MESH(I,7) = MESH(I,1) + MESH(I,3)*OFFSET
          MESH(I,8) = MESH(I,2) + MESH(I,4)*OFFSET
50    CONTINUE
C
      RETURN
      END
C
C
      SUBROUTINE TMAT(U,PERND,MESH,T,OFFSET,DPER,BCOND,MRES,LBIAS,GBIAS,
     CMAXD)
C
C     THIS SUBROUTINE CALCULATES THE GREEN'S FUNCTION INTEGRAL
C     (FOR A SINGLE BASIS FUNCTION BOUNDARY CONDITION) GEOMETRIC
C     PERIMETER WITH RESPECT TO EACH OF THE OFFSET BOUNDARY NODES.
C     THE INTEGRATION IS REPEATED UNTIL EACH BOUNDARY CONDITION HAS BEEN
C     INDIVIDUALLY APPLIED AND INTEGRATED WITH RESPECT TO EACH OFFSET
C     BOUNDARY NODE.  THESE VALUES ARE RETURNED IN THE " T " MATRIX
C     FOR USE IN THE FIELD FEEDBACK FORMULATION.  THE MATRIX IS
C     ORGANIZED, T m,n.
C
      COMPLEX U(100,100),T(100,100),PVEC(100),PDVEC(100)
      COMPLEX J,H0RP1,H0RP2,H1RP1,SUM,TEMP(100),BCOND(100)
      COMPLEX H1RP2,PSI,PSIRP,INTEGRAL,DPSIC
      REAL RMRP,NORM11,NORM12,DOT,DPER,DISTM,MAXD
      REAL OFFSET,MESH(100,8),DIST,R,DZ,DL,MRES
      INTEGER I,M,N,NN,STEP,FN,SN,PERND,MM,STEPMX,STEPMN,LBIAS,GBIAS
      OPEN (UNIT =  2, FILE = 'C: MSFORT TMAT.DAT',STATUS = 'UNKNOWN')
C
      J = (0.0,1.0)
      STEPMX = INT(DPER*(-48.0*OFFSET + 17.2) + LBIAS)
      STEPMN = INT(DPER + GBIAS)
C
      WRITE(*,*) ' LOADING T MATRIX '
      WRITE(*,*) ' MAXIMUM STEP = ',STEPMX,', MINIMUM STEP = ',STEPMN
      WRITE(*,*) ' MAXIMUM DISTANCE FOR ANY CONTRIBUTION = ',MAXD
      DO 40, M = 1, PERND
          DO 5, I = 1, PERND
              IF(M.EQ.I) THEN
                  PVEC(I) = (1.0 + U(I,M))/2.0
                  PDVEC(I) = (1.0 - U(I,M))/OFFSET
              ELSE
                 PVEC(I) = U(I,M)/2.0
                 PDVEC(I) =  -U(I,M)/OFFSET
```

```
                ENDIF
5           CONTINUE
C
            DO 30, N = 1, PERND
                WRITE(*,1000) M, N, PERND
                SUM = (0.0,0.0)
                DO 20, NN = 1, PERND
                    FN = NN
                    IF(NN.EQ.PERND) THEN
                        SN = 1
                    ELSE
                        SN = NN + 1
                    ENDIF
                    DIST = SQRT((MESH(FN,5)-MESH(SN,5))**2+(MESH(FN,6)
    C               -MESH(SN,6))**2)
                    INTEGRAL = (0.0,0.0)
C
        DISTM = SQRT((MESH(N,7)-MESH(FN,5))**2+(MESH(N,8)-MESH(FN,6))**2)
C
        R = MESH(SN,5) - MESH(FN,5)
        DZ = MESH(SN,6) - MESH(FN,6)
        DL = SQRT(R**2 + DZ**2)
        NORM11 = -DZ/DL
        NORM12 = R/DL
C
        IF(DISTM.GT.MAXD) THEN
            GOTO 20
        ELSEIF(DISTM.LE.1.0) THEN
            STEP = STEPMX
        ELSE
            STEP = STEPMN
        ENDIF
        IF(STEP.LT.1) STEP = 1
C
        DO 10, I = 1, STEP+1
            IF(I.EQ.1) THEN
                RMRP = SQRT((MESH(N,7)-(MESH(FN,5)+0.25*(MESH(SN,5)-
    C           MESH(FN,5))/FLOAT(STEP)))**2+(MESH(N,8)-(MESH(FN,6)+
    C           0.25*(MESH(SN,6)-MESH(FN,6))/FLOAT(STEP)))**2)
                DPSIC = PDVEC(FN) + 0.25*(PDVEC(SN)-PDVEC(FN))/STEP
                PSIRP = PVEC(FN) + 0.25*(PVEC(SN)-PVEC(FN))/STEP
                DOT = (NORM11*(MESH(N,7) - (MESH(FN,5)+0.25*(MESH(SN,5)-
    C           MESH(FN,5))/STEP))+(NORM12*(MESH(N,8)-(MESH(FN,6)+0.25*
    C           (MESH(SN,6)-MESH(FN,6))/STEP))))/RMRP
            ELSEIF(I.EQ.STEP+1) THEN
                RMRP = SQRT((MESH(N,7)-(MESH(FN,5)+(FLOAT(STEP)-0.25)*(
    C           MESH(SN,5)-MESH(FN,5))/FLOAT(STEP)))**2+(MESH(N,8)-(MESH
    C           (FN,6)+(FLOAT(STEP)-0.25)*(MESH(SN,6)-MESH(FN,6))/FLOAT
    C           (STEP)))**2)
                DPSIC=PDVEC(FN)+(FLOAT(STEP)-0.25)*(PDVEC(SN)-PDVEC(FN))
    C           /(STEP)
                PSIRP = PVEC(FN)+(FLOAT(STEP)-0.25)*(PVEC(SN)-PVEC(FN))/
    C           (STEP)
                DOT = (NORM11*(MESH(N,7)-(MESH(FN,5)+(FLOAT(STEP)-0.25)*
    C           (MESH(SN,5)-MESH(FN,5))/STEP))+(NORM12*(MESH(N,8)-(MESH
    C           (FN,6)+(FLOAT(STEP)-0.25)*(MESH(SN,6)-MESH(FN,6))/STEP)
```

```
C           )))/RMRP
        ELSE
            RMRP = SQRT((MESH(N,7)-(MESH(FN,5)+FLOAT(I-1)*(MESH(SN,
C           5)-MESH(FN,5))/FLOAT(STEP)))**2+(MESH(N,8)-(MESH(FN,6)+
C           FLOAT(I-1)*(MESH(SN,6)-MESH(FN,6))/FLOAT(STEP)))**2)
            DPSIC=PDVEC(FN)+(I-1)*(PDVEC(SN)-PDVEC(FN))/(STEP)
            PSIRP = PVEC(FN)+(I-1)*(PVEC(SN)-PVEC(FN))/(STEP)
            DOT = (NORM11*(MESH(N,7)-(MESH(FN,5)+(I-1)*(MESH(SN,5)-
C           MESH(FN,5))/STEP))+(NORM12*(MESH(N,8)-(MESH(FN,6)+(I-1)*
C           (MESH(SN,6)-MESH(FN,6))/STEP))))/RMRP
        ENDIF
        CALL HAN1(CMPLX(RMRP,0.0),HORP1,H1RP1)
        IF((I.EQ.1).OR.(I.EQ.STEP+1)) THEN
            PSI = (J/4.0)*(HORP1*DPSIC - PSIRP*DOT*H1RP1)/2
        ELSE
            PSI = (J/4.0)*(HORP1*DPSIC - PSIRP*DOT*H1RP1)
        ENDIF
        INTEGRAL = INTEGRAL + PSI*DIST/STEP
C
10                  CONTINUE
                    SUM = SUM + INTEGRAL
20              CONTINUE
                T(N,M) = SUM
30          CONTINUE
40      CONTINUE
C
        DO 55, I = 1, PERND
            TEMP(I) = (0.0,0.0)
            DO 50, M = 1, PERND
                TEMP(I) = TEMP(I) + T(I,M)*BCOND(M)
50          CONTINUE
            WRITE(2,*) (T(I,MM), MM = 1, PERND)
55      CONTINUE
C
        DO 60, I = 1, PERND
            WRITE(2,*) I, TEMP(I)
60      CONTINUE
C
1000    FORMAT(1X,'COLUMN ',I3,', ROW ',I3,' OUT OF ',I3)
C
        CLOSE(2)
C
        RETURN
        END
C
C
        SUBROUTINE CNSOLV(T,BCOND,CNVEC,PERND)
C
C       THIS SUBROUTINE CALCULATES THE C VECTOR BY SOLVING:
C
C                  -1
C       Cn = [I - T]   * BOUNDARY CONDITIONS (INCIDENT FIELDS)
C
        COMPLEX BCOND(100),T(100,100),TEMP(100),CNVEC(100), DETERM
        REAL COND, DMAG
```

```
      INTEGER PERND,I,J,K,L,INORM,NMAX
C
      INORM = 0
      NMAX = 100
C
      DO 10, I = 1, PERND
           DO 10, J = 1, PERND
                IF(I.EQ.J) THEN
                     T(I,J) = 1 - T(I,J)
                ELSE
                     T(I,J) = -T(I,J)
                ENDIF
10    CONTINUE
      WRITE(*,*) 'INVERTING THE [I - T] MATRIX
C
      CALL CSMINV(T,NMAX,PERND,DETERM,COND,INORM)
C
      DMAG = CABS(DETERM)
      WRITE(*,*) ' DETERMINANT      = ', DMAG
      WRITE(*,*) ' CONDITION NUMBER = ', COND
      WRITE(*,*) ' MULTIPLING MATRICES TO FORM THE SCATTERED FIELDS '
C
      DO 30, I = 1, PERND
           CNVEC(I) = (0.0,0.0)
           DO 30, J = 1, PERND
                CNVEC(I) = CNVEC(I) + T(I,J)*BCOND(J)
30    CONTINUE
C
      RETURN
      END
C
C
      SUBROUTINE FFLD(CNVEC,PERND,MESH,U,OFFSET,K,NRES,XORG,YORG)
C
C     THIS SUBROUTINE CALCULATES THE FAR FIELDS DUE TO THE OFFSET
C     BOUNDARY SCATTERED FIELDS AND THE PERIMETER SCATTERED FIELDS.
C     ADDITIONAL GREEN'S FUNCTION INTEGRALS ARE ACCOMPLISHED.
C
      COMPLEX CNVEC(100),U(100,100),J,PSISP(100),TEMP,PSI,DPSI
      COMPLEX DPSISP(100),INTEGRAL,DPSIM(100,100),PSIM(100,100)
      REAL MESH(100,8),OFFSET,K,DOT,DOT1,PI,ARES,XORG,YORG,DIST,SIGMA
      REAL R,DZ,DL,NORM11,NORM12
      INTEGER PERND,I,M,N,L,NRES,FN,SN,STEP,II
C
      WRITE(*,*) ' CALCULATING THE SCATTERED FIELDS'
      J = (0.0,1.0)
      PI = 4.0*ATAN(1.0)
      ARES = 2.0*PI/FLOAT(NRES)
      STEP = 5
C
      DO 5, M = 1, PERND
           DO 5, I = 1, PERND
                IF(M.EQ.I) THEN
                     PSIM(I,M) = (1.0 + U(I,M))/2.0
                     DPSIM(I,M) = (1.0 - U(I,M))/OFFSET
                ELSE
```

```
                    PSIM(I,M) = U(I,M)/2.0
                    DPSIM(I,M) =  -U(I,M)/OFFSET
                ENDIF
5       CONTINUE
C
        DO 10, I = 1, PERND
            PSISP(I) = (0.0,0.0)
            DPSISP(I) = (0.0,0.0)
            DO 10, L = 1, PERND
                PSISP(I) = PSISP(I) + PSIM(I,L)*CNVEC(L)
                DPSISP(I) = DPSISP(I) + DPSIM(I,L)*CNVEC(L)
10          CONTINUE
C
        DO 30, I = 0, NRES-1
            WRITE(6,1000) I+1, NRES
            INTEGRAL = (0.0,0.0)
            DO 20, N = 1, PERND
                FN = N
                IF(N.EQ.PERND) THEN
                    SN = 1
                ELSE
                    SN = N + 1
                ENDIF
                R = MESH(SN,5) - MESH(FN,5)
                DZ = MESH(SN,6) - MESH(FN,6)
                DL = SQRT(R**2 + DZ**2)
                NORM11 = -DZ/DL
                NORM12 = R/DL
                DO 15, II = 1, STEP+1
                  DIST = SQRT((MESH(FN,5)-MESH(SN,5))**2+(MESH(FN,6)-
    C             MESH(SN,6))**2)
                  IF(II.EQ.1) THEN
                    PSI = PSISP(FN)+0.25*(PSISP(SN)-PSISP(FN))/
    C               FLOAT(STEP)
                    DPSI = DPSISP(FN)+0.25*(DPSISP(SN)-DPSISP
    C               (FN))/FLOAT(STEP)
                    DOT = NORM11*SIN(I*ARES) + NORM12*COS(I*ARES)
                    DOT1 = (MESH(FN,5)+0.25*(MESH(SN,5)-MESH(FN,
    C               5))/FLOAT(STEP)-XORG)*SIN(I*ARES) + (MESH(FN,6)+
    C               0.25*(MESH(SN,6)-MESH(FN,6))/FLOAT(STEP)-
    C               YORG)*COS(I*ARES)
                  ELSEIF(II.EQ.STEP+1) THEN
                    PSI = PSISP(FN)+(FLOAT(STEP)-0.25)*(PSISP(SN)-
    C               PSISP(FN))/FLOAT(STEP)
                    DPSI = DPSISP(FN)+(FLOAT(STEP)-0.25)*(DPSISP(SN)-
    C               DPSISP(FN))/FLOAT(STEP)
                    DOT = NORM11*SIN(I*ARES) + NORM12*COS(I*ARES)
                    DOT1 = (MESH(FN,5)+(FLOAT(STEP)-0.25)*(MESH(SN,5)-
    C               MESH(FN,5))/FLOAT(STEP)-XORG)*SIN(I*ARES) + (MESH(
    C               FN,6)+(FLOAT(STEP)-0.25)*(MESH(SN,6)-MESH(FN,6))/
    C               FLOAT(STEP)-YORG)*COS(I*ARES)
                  ELSE
                    PSI = PSISP(FN)+FLOAT(II-1)*(PSISP(SN)-PSISP(FN))/
    C               FLOAT(STEP)
                    DPSI = DPSISP(FN)+FLOAT(II-1)*(DPSISP(SN)-DPSISP
    C               (FN))/FLOAT(STEP)
```

```
                         DOT = NORM11*SIN(I*ARES) + NORM12*COS(I*ARES)
                         DOT1 = (MESH(FN,5)+FLOAT(II-1)*(MESH(SN,5)-MESH(FN,
     C                   5))/FLOAT(STEP)-XORG)*SIN(I*ARES) + (MESH(FN,6)+
     C                   FLOAT(II-1)*(MESH(SN,6)-MESH(FN,6))/FLOAT(STEP)-
     C                   YORG)*COS(I*ARES)
                      ENDIF
                         IF((II.EQ.1).OR.(II.EQ.STEP+1)) THEN
                              TEMP=(J*DPSI+DOT*PSI)*(EXP(J*DOT1))*DIST/(2.0*
     C                        STEP)
                         ELSE
                              TEMP=(J*DPSI+DOT*PSI)*(EXP(J*DOT1))*DIST/
     C                        (STEP)
                         ENDIF
                         INTEGRAL = INTEGRAL + TEMP
15                  CONTINUE
20             CONTINUE
               SIGMA = ((CABS(INTEGRAL))**2.0)/(4.0*K)
               WRITE(50,1010) I+1, (I*ARES*180.0/PI), SIGMA
30        CONTINUE
C
          WRITE(6,*) '                                                     '
          WRITE(6,*) ' <<< FIELD PATTERN STORED IN FFPAT.DAT >>> '
          WRITE(6,*) ' '
C
1000      FORMAT(1X,'INTEGRAL ',I3,', OUT OF ',I3,' COMPLETED')
1010      FORMAT(1X,I3,2X,F6.2,2X,E14.8)
C
          RETURN
          END
C
C
          SUBROUTINE HAN1(Z,H0,H1)
C
C         Computing Hankel Functions for n=0,1 with
C         Complex Argument, Z.   Direct Power Series Method for
C         CABS(Z) .LE. 5  and Hankel's Asymptotic Formula for
C         CABS(Z) .GT. 5.   Written 11/6/87 by M.A. Morgan
C
          INTEGER M,M2
          REAL C(34),DM,F(34),G0,P(34),Pi,P2
          COMPLEX Z,Z2,Z3,Z4,J0,J1,Y0,Y1,AM,CL,P0,P1,Q0,Q1
          COMPLEX E0,E1,X0,X1,H0,H1,j
          PI=3.1415927
          P2=2.0/PI
          j=(0.,1.)
          IF(CABS(Z).LE.5.0) THEN
C
C               Direct Power Series Method
C
                G0= 1.78072
                Z2=0.5*Z
                CL=CLOG(G0*Z2)
C
C       Computing F(m) = m !   and P(m) = 1 + 1/2 + 1/3 + ....+ 1/m
C
                F(1)=1.0
```

```
              P(1)=1.0
              DO 11 M=2,34
                    F(M)=M*F(M-1)
                    P(M)=P(M-1)+1.0/M
11            CONTINUE
C
C   Computing Power Series Coefficients
C
              DM=-1.0
              DO 22 M=1,34
                    C(M)=DM/(F(M)*F(M))
                    DM=-DM
22            CONTINUE
C
C   Computing J0 and J1
C
              J0=(1.,0.)
              J1=(0.,0.)
              M=0
33            M=M+1
              M2=2*M
              AM=C(M)*(Z2**M2)
              J0=J0+AM
              J1=J1-M*AM
              IF((CABS(AM).GT.1.0E-10).AND.(M.LT.34)) GO TO 33
              J1=J1/Z2
C
C   Computing Y0 and Y1
C
              M=0
              Y0=CL*J0
              Y1=Z2*CL*J1-0.5*J0
44            M=M+1
              M2=2*M
              AM=C(M)*P(M)*(Z2**M2)
              Y0=Y0-AM
              Y1=Y1+M*AM
              IF((CABS(AM).GT.1.0E-10).AND.(M.LT.34)) GO TO 44
              Y0=P2*Y0
              Y1=P2*Y1/Z2
              H0=J0-j*Y0
              H1=J1-j*Y1
              RETURN
        ELSE
C
C   Hankel' Asymptotic Formula (Abram. & Stegun p. 364
C
              Z2=Z*Z
              Z3=Z*Z2
              Z4=Z*Z3
              P0=1.0-.0703125/Z2+.1121521/Z4
              Q0=-.125/Z+.0732422/Z3
              P1=1.0+.1171875/Z2-.1441956/Z4
              Q1=.375/Z-.10253906/Z3
              X0=(Z-.25*PI)
              X1=(Z-.75*PI)
```

```
            E0=CEXP(-j*X0)
            E1=CEXP(-j*X1)
            AM=CSQRT(P2/Z)
            H0=AM*(P0-j*Q0)*E0
            H1=AM*(P1-j*Q1)*E1
      ENDIF
C
      RETURN
      END
C
C
      SUBROUTINE CSMINV(A,NDIM,N,DETERM,COND,INORM)
C
C     INORM - FLAG TO NORMALIZE COLUMNS AND ROWS OF MATRIX A
C     MATRIX NORMALIZATION BY M.A. MORGAN
C     APRIL 24,1978
C
C     A      - MATRIX TO INVERT              - INPUT/OUTPUT
C     NDIM   -                               - INPUT
C     N      -                               - INPUT
C     DETERM - DETERMINATE OF A              - OUTPUT
C     COND   - CONDITION NUMBER OF A         - OUTPUT
C     INORM  - INTEGER NORMALIZATION FLAG    - INPUT
C
C
      COMPLEX A(100,100),PIVOT(100),AMAX,T,SWAP,DETERM,U
      INTEGER I,J,K,L,IPIVOT(100),INDEX(100,2),IROW,ICOLUM,L1,JROW
      INTEGER JCOLUM,N,INORM
      REAL TEMP,ALPHA(100),COL(100),ROW(100),AJK,SUMAXA,SUMROW,SUMAXI
C
      IF(NDIM.GT.100) THEN
           WRITE(*,*) ' ERROR IN INVERTION CALL... DIMENSION > 100 '
           STOP
      ENDIF
      IF(N.GT.NDIM) THEN
           WRITE(*,*) ' ERROR IN INVERTION CALL... N > MAX DIM. '
           STOP
      ENDIF
      IF(INORM.NE.1) GO TO 7
      DO 3 K = 1, N
           COL(K) = 0.0
           DO 1 J = 1,N
                AJK = CABS(A(J,K))
                IF(AJK.GT.COL(K)) COL(K) = AJK
1          CONTINUE
           DO 2 J = 1, N
2          A(J,K) = A(J,K)/COL(K)
3     CONTINUE
C     ROW NORMALIZING
      DO 6 J = 1, N
           ROW(J) = 0.0
           DO 4 K = 1, N
                AJK = CABS(A(J,K))
                IF(AJK.GT.ROW(J)) ROW(J) = AJK
4          CONTINUE
```

```
          DO 5 K = 1, N
5              A(J,K) = A(J,K)/ROW(J)
6     CONTINUE
7     CONTINUE
      DETERM = CMPLX(1.0,0.0)
      SUMAXA = 0.0
      DO 20 J = 1, N
          ALPHA(J) = 0.0
          SUMROW = 0.0
          DO 10 I = 1, N
               ALPHA(J) = ALPHA(J) + A(J,I)* CONJG(A(J,I))
10        SUMROW = SUMROW + CABS(A(J,I))
          ALPHA(J) = SQRT(ALPHA(J))
          IF(SUMROW.GT.SUMAXA) SUMAXA = SUMROW
20    IPIVOT(J) = 0
      DO 600 I = 1, N
C
C
          AMAX = CMPLX(0.0,0.0)
          DO 105 J = 1, N
               IF (IPIVOT(J)-1) 60, 105, 60
60             DO 100 K = 1, N
                    IF (IPIVOT(K)-1) 80, 100, 740
80                  TEMP = AMAX*CONJG(AMAX) - A(J,K)*CONJG(A(J,K))
                    IF(TEMP)85,85,100
85                       IROW = J
                         ICOLUM = K
                         AMAX = A(J,K)
100            CONTINUE
105       CONTINUE
          IPIVOT(ICOLUM) = IPIVOT(ICOLUM) + 1
C
C
          IF (IROW-ICOLUM) 140, 260, 140
140            DETERM = -DETERM
               DO 200 L = 1, N
                    SWAP = A(IROW,L)
                    A(IROW,L) = A(ICOLUM,L)
200            A(ICOLUM,L) = SWAP
               SWAP = ALPHA(IROW)
               ALPHA(IROW) = ALPHA(ICOLUM)
               ALPHA(ICOLUM) = SWAP
260            INDEX(I,1) = IROW
               INDEX(I,2) = ICOLUM
               PIVOT(I) = A(ICOLUM,ICOLUM)
               U = PIVOT(I)
               DETERM = DETERM*U
               DETERM = DETERM/ALPHA(ICOLUM)
               TEMP = PIVOT(I)*CONJG(PIVOT(I))
               IF(TEMP)330,720,330
C
C
330            A(ICOLUM,ICOLUM) = CMPLX(1.0,0.0)
               DO 350 L = 1, N
                    U = PIVOT(I)
350            A(ICOLUM,L) = A(ICOLUM,L)/U
```

```
C
C
380             DO 550 L1 = 1, N
                     IF(L1-ICOLUM) 400, 550, 400
400                  T = A(L1,ICOLUM)
                     A(L1,ICOLUM) = CMPLX(0.0,0.0)
                     DO 450 L = 1, N
                          U = A(ICOLUM,L)
450                  A(L1,L) = A(L1,L) - U*T
550        CONTINUE
600   CONTINUE
C
C
620   DO 710 I = 1, N
           L = N + 1 - I
           IF (INDEX(L,1) - INDEX(L,2)) 630, 710, 630
630             JROW = INDEX(L,1)
                JCOLUM = INDEX(L,2)
                DO 705 K = 1, N
                     SWAP = A(K,JROW)
                     A(K,JROW) = A(K,JCOLUM)
                     A(K,JCOLUM) = SWAP
705             CONTINUE
710   CONTINUE
      SUMAXI = 0.0
      DO 910 I = 1, N
           SUMROW = 0.0
           DO 900 J = 1, N
900        SUMROW = SUMROW + CABS(A(I,J))
           IF(SUMROW.GT.SUMAXI) SUMAXI = SUMROW
910   CONTINUE
      COND = SUMAXA*SUMAXI
      IF(INORM.NE.1) GO TO 955
           DO 950 K = 1, N
                DO 950 J = 1, N
950        A(J,K) = A(J,K)/(ROW(K)*COL(J))
955   CONTINUE
      RETURN
720   WRITE(*,730)
730   FORMAT( ' MATRIX IS SINGULAR')
740   RETURN
      END
C
C
```

APPENDIX F. DIELECTRIC CYLINDER SCATTERING PROGRAM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     PLANEWAVE SCATTERING BY A DIELECTRIC CYLINDER                    C
C     E - WAVE  (TM CASE)                                              C
C     H - WAVE  (TE CASE)                                              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      COMPLEX GAMMA(0:200,2),SIGMA(2),ER,MU,KR,YR,ZR
      COMPLEX*16 JA(0:200),DJA(0:200),KRRA
      COMPLEX*16 J(0:200),DJ(0:200)
      REAL*8 Y(0:200),DY(0:200),YA(0:200),DYA(0:200),JB(0:200)
      REAL*8 DJB(0:200),RA,KORA
      REAL KO,PI,SIGMAN(2),A,B
      INTEGER I,II,MODE,ARES,N
C
      OPEN(3,FILE='C: MSFORT DECTEM.DAT')
      PI = 3.1415927
C
      WRITE(*,*) 'PERMITTIVITY FORMAT IS  " a + jb, " '
      WRITE(*,*) 'Enter Dielectric Constant (REAL PART, a) '
      READ(*,*) A
      WRITE(*,*) 'Enter Dielectric Constant (IMAGINARY PART, b) '
      READ(*,*) B
      ER = CMPLX(A,B)
C
      WRITE(*,*) 'PERMEABILITY FORMAT IS  " a + jb, " '
      WRITE(*,*) 'Enter Permeability Constant (REAL PART, a) '
      READ(*,*) A
      WRITE(*,*) 'Enter Permeability Constant (IMAGINARY PART, b) '
      READ(*,*) B
      MU = CMPLX(A,B)
C
      WRITE(*,*) 'Enter the wave number (ko) '
      READ(*,*) KO
C
      WRITE(*,*) 'Enter Cylinder Radius (IN WAVENUMBER UNITS)  '
      WRITE(*,*) 'WARNING: Do Not Enter Zero ! '
      READ(*,*) KORA
C
      KR = CSQRT(MU*ER)
      YR = CSQRT(ER/MU)
      ZR = CSQRT(MU/ER)
      RA = KORA/KO
      KRRA = KR*RA
C
      WRITE(*,*) 'Enter No. of Modes:    '
      READ(*,*) MODE
      WRITE(*,*) 'Enter the angular resolution    '
      READ(*,*) ARES
C
```

```
      WRITE(3,*) 'Cylinder Scattering vs. angle'
      WRITE(3,110) ER,MU,RA,MODE,K0,KRRA
      WRITE(3,*) ' '
C
      CALL BES(MODE,K0RA,JB,Y,DJB,DY)
      CALL DCBJNS (DCMPLX(K0RA,0.0D+00),MODE,J,DJ)
      CALL DCBJNS ((KRRA*K0),MODE,JA,DJA)
      WRITE(*,*) ' RETURNED FROM FINAL BESSEL CALL'
C
C     CALCULATING GAMMAs
C
      DO 10, N = 0, MODE
           GAMMA(N,1) = (JA(N)*DJ(N)-YR*DJA(N)*J(N))/(JA(N)*CMPLX(DJ(N)
     C     ,-DY(N)) - YR*DJA(N)*CMPLX(J(N),-Y(N)))
           GAMMA(N,2) = (JA(N)*DJ(N)-ZR*DJA(N)*J(N))/(JA(N)*CMPLX(DJ(N)
     C     ,-DY(N)) - ZR*DJA(N)*CMPLX(J(N),-Y(N)))
           WRITE(*,1000) N,GAMMA(N,1),GAMMA(N,2)
10    CONTINUE
C
C     CALCULATING SIGMAs
C
      DO 30, I = 180, -180, -ARES
        DO 40, II = 1, 2
           SIGMA(II) = (0.0,0.0)
           DO 20, N = 1, MODE
                SIGMA(II) = SIGMA(II)+2.0*GAMMA(N,II)*COS(N*PI*I/180.0)
20         CONTINUE
                SIGMA(II) = SIGMA(II) + GAMMA(0,II)
                SIGMAN(II) = ((4.0/K0)*(CABS(SIGMA(II)))**2)
40      CONTINUE
        WRITE(3,120) I, SIGMAN(1), SIGMAN(2)
30    CONTINUE
C
C
110   FORMAT(1X,'Er                =         ',F6.4,1X,F6.4,/,
     C' Mu              =         ',F6.4,1X,F6.4,/,
     C' RADIUS (METERs) =         ',F8.5,/,
     C' MAX MODE        =         ',I3,/,
     C' K0              =         ',F8.5,/,
     C' KrRa            =         ',F8.5,1X,F8.5)
120   FORMAT(1X,I4,2(3X,E14.4))
1000  FORMAT(1X,I3,1X,2(E12.4,1X,E12.4,4X))
C
C
      STOP
      END
```

APPENDIX G. SOFTWARE SOURCES

1. DISSPLA
   Integrated Software Systems Corporation
   10505 Sorrento Valley Road
   San Diego, CA 92121

2. CURVE-DIGITIZER
   West Coast Consultants
   4202 Genesee Avenue, Suite 309
   San Diego, CA 92117

3. Microsoft FORTRAN
   16011 NE 36th Way
   BOX 97017
   Redmond, WA 98073

4. Microway NDP FORTRAN
   POB 79
   Kingston, MA 02364

5. Prof. M.A. Morgan, Code 62Mw
   Naval Postgraduate School
   Monterey, CA 93943

6. LT T.B. Welch III
   c/o T.B. Welch Jr.
   1318 Walthour Road
   Savannah, GA 31410

LIST OF REFERENCES

1. Mittra, R. (Editor), *Computer Techniques in Electromagnetics*, Pergamon Press, New York, NY, 1973.

2. Welch, B.A., *Concept Evaluation: Field Feedback Computation of Electromagnetic Scattering*, Master's Thesis, Naval Postgraduate School, Monterey, CA, June 1980.

3. Morgan, M.A. and Welch, B.A., "Field Feedback Formulation for Electromagnetic Scattering", *IEEE Transactions in Antennas and Propagation*, December 1986.

4. Morgan, M.A., Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 2 April 1988.

5. Morgan, M.A., Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 13 April 1988.

6. Morgan, M.A., Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 26 May 1988.

7. Morgan, M.A., Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 9 January 1989

8. Morgan, M.A., Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 20 May 1988

9. Morgan, M.A. Unpublished notes, Naval Postgraduate School, Monterey, CA, dated 3 February 1989

10. Richmond, J.H., "Scattering by a Dielectric Cylinder of Arbitrary Cross-Section", *IEEE Transanction on Antennas and Propagation*, May 1965

11. Richmond, J.H. "TE - Wave Scattering by a Dielectric Cylinder of Arbitrary Cross-Section", *IEEE Transactions on Antennas and Propagation*, July 1966

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Professor John P. Powers, Chairman Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Michael A. Morgan Code 62 Mw Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	10
5.	Professor Ramakrishna Janaswamy Code 62 Js Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor Richard W. Adler Code 62 Ab Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Dr. Arthur Jordan Code 1114SE Office of Naval Research 800 North Quincy Street Arlington, VA 22217	1
8.	Dr. Bill Pala Code 5316 Naval Research Laboratory 4555 Overlook Avenue, South West Washington, DC 20375	1

9. Mr. Peter Hoag | 1
Code 25D
Office of Naval Technology
800 North Quincy Street
Arlington, VA 22217

10. Mr. Charles Heber | 1
AST Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

11. Mr. Daniel Carpenter | 1
TRW Military Electronics and Aviation Division
One Rancho Carmel
San Diego, CA 92128

12. Dr. David Lewis | 1
AST Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

13. Mr. Joseph Mosko | 1
Code 35203
Naval Weapons Center
China Lake, CA 93555

14. Ms. Pamela Overfelt | 1
Code 3814
Naval Weapons Center
China Lake, CA 93555

15. LT Thaddeus B. Welch III | 2
1318 Walthour Road
Savannah, GA 31410