

**A Finite Element Boundary Integral
Formulation for Radiation and
Scattering by Cavity Antennas
Using Tetrahedral Elements**

**J. Gong, J. L. Volakis, A. Chatterjee and
J. M. Jin**

**National Aeronautics and
Space Administration
Ames Research Center
Moffett Field CA 94035**

January 15, 1992

MR0409

Report Title: A Finite Element Boundary Integral Formulation for
Radiation and Scattering by Cavity Antennas Using
Tetrahedral Elements

Report Authors: J. Gong, J.L. Volakis, A. Chatterjee and J.M. Jin

Primary University Collaborator: John L. Volakis

Primary NASA-Ames Collaborator: Alex Woo

University Address: Radiation Laboratory
Department of Electrical Engineering
and Computer Science
The University of Michigan
Ann Arbor MI 48109-2122

Date: January 15, 1992

Funds for the support of this study were in part allocated by the NASA-Ames Research Center, Moffett Field, California, under interchange No. NCA2-653.

**A FINITE ELEMENT BOUNDARY INTEGRAL
FORMULATION FOR RADIATION AND SCATTERING
BY CAVITY ANTENNAS USING TETRAHEDRAL
ELEMENTS**

J. Gong, J.L. Volakis, A. Chatterjee and J.M. Jin

*Radiation Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122*

ABSTRACT

A hybrid finite element-boundary integral formulation is developed using tetrahedral/triangular elements for discretizing the cavity/aperture of microstrip antennas/arrays. The tetrahedral elements with edge based linear expansion functions are chosen for modeling the volume region and triangular elements are used for discretizing the aperture. The edge-based expansion functions are divergenceless thus removing the requirement to introduce a penalty term and the tetrahedral elements permit greater geometrical adaptability than the rectangular bricks. The underlying theory and resulting expressions are discussed in detail together with some numerical scattering examples for comparison and demonstration.

Introduction

Recently, a hybrid finite element formulation was developed for a characterization of the scattering and radiation properties associated with microstrip patch antennas and arrays residing in a cavity that is recessed in a ground plane [1], [2], [3]. The technique employs the finite element method (FEM) to model the substrate in the cavity region and the mesh was terminated at the aperture of the cavity via the boundary integral method. By virtue of the FEM, the analysis is applicable to patch antennas, slots and arrays which reside on or are embedded in the layered dielectric substrate. Various feed structures and impedance loads can also be modeled within the context of the FEM without difficulty. As demonstrated in [3], this hybrid version of the finite element method proved very successful and accurate in treating complex antenna configurations and large arrays. The last is owed to the sparsity of the finite element matrix and although the boundary integral resulted in a partially full matrix it did not burden the memory requirements because it was Toeplitz in form. Specifically, when the system is solved via the biconjugate gradient method in conjunction with the fast Fourier transform (FFT) [4, 5], the required memory is only $6.25N_t + 10.5N_s$, where N_t and N_s denote, respectively, the total number of unknowns in the entire cavity and the unknowns or edge elements at the aperture of the cavity.

So far the implementation of the proposed finite element method has only been carried out by subdividing the cavity volume using rectangular bricks (rectangular hexahedra). Obviously, this limits the utility of the method to those geometries and antennas which fit in a rectangular uniform grid. Consequently, circular patches, non-rectangular slots or irregular cavities and feeding lines cannot be modeled with these discretization elements. A more adaptable volume element is the tetrahedron (see figure 1(a)) which leads to a discretization of the cavity surface in terms of triangles as illustrated in figures 1(b) and 1(c). In the following, we describe the implementation of the proposed hybrid FE-BI method using tetrahedra and triangular facets for volume and surface elements, respectively. The analysis for generating the required system is outlined and some preliminary results are presented which validate the formulation.

Basic Equations

Consider the geometry given in figure 2, that of a cavity in a ground plane. A source is placed in the cavity and we are interested in computing the radiated field in the region above S_{cav} (i.e. in region I). Using the procedure discussed by Jin and Volakis [1–3], this type of problem can be readily handled by subdividing the computational domain into two regions to be referred to as regions I and II. In region II, which encompasses the volume enclosed by the cavity walls and S_{cav} , the finite element method will be employed to formulate the fields. The primary reason for using the finite element method is its adaptability in modeling a variety of cavities and radiating elements. The fields in region I (exterior region) will be computed via the boundary integral method. This amounts to introducing equivalent sources over

the cavity's aperture which are then integrated to obtain the radiated fields. The exterior and interior fields are then coupled by imposing the continuity condition across the aperture.

1. Interior Region Formulation

From Maxwell's equations we obtain the vector wave equation

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \right) - k_0^2 \epsilon_r \mathbf{E} = -jk_0 Z_0 \mathbf{J}^i + \nabla \times \left(\frac{1}{\mu_r} \mathbf{M}^i \right) \quad (1)$$

The solution of this equation is equivalent to minimizing the variational function

$$\begin{aligned} F(\mathbf{E}) &= \frac{1}{2} \iiint_{V_{\text{cav}}} \left\{ \frac{1}{\mu_r} (\nabla \times \mathbf{E}) \cdot (\nabla \times \mathbf{E}) - k_0^2 \epsilon_r \mathbf{E} \cdot \mathbf{E} \right\} dv \\ &+ \iiint_{V_{\text{cav}}} \mathbf{E} \cdot \left[jk_0 Z_0 \mathbf{J}^i - \nabla \times \left(\frac{1}{\mu_r} \mathbf{M}^i \right) \right] dv \\ &+ jk_0 Z_0 \iint_S \mathbf{E} \cdot (\mathbf{H} \times \hat{\mathbf{z}}) ds \end{aligned} \quad (2)$$

Before taking the variation of F with respect to \mathbf{E} we must first discretize it for numerical implementation. To this end, we subdivide V_{cav} into N_e small tetrahedral elements of volume V_e . Within each volume element (say the e th element) we expand the field as

$$\mathbf{E} = \sum_{i=1}^{N_e} E_i^e \mathbf{V}_i^e \quad (3)$$

where \mathbf{V}_i^e are the basis elements for the e th element and E_i^e are the unknown coefficients of the expansion. Referring to figure 3, \mathbf{V}_i^e are given by

$$\mathbf{V}_{7-i}^e(\mathbf{r}) = \begin{cases} \mathbf{f}_{7-i} + \mathbf{g}_{7-i} \times \mathbf{r} & \mathbf{r} \in V_e \\ 0 & \text{outside element} \end{cases} \quad (4)$$

$$\mathbf{f}_{7-i} = \frac{b_{7-i}}{6V_e} \mathbf{r}_{i_1} \times \mathbf{r}_{i_2} \quad \mathbf{r}_{i_1}, \mathbf{r}_{i_2} : \text{position vectors of vertices } i_1 \text{ and } i_2 \text{ (see Table 1)} \quad (5)$$

$$\mathbf{g}_{7-i} = \frac{b_i b_{7-i}}{6V_e} \mathbf{e}_i \quad (6)$$

$$\mathbf{e}_i = \frac{(\mathbf{r}_{i_2} - \mathbf{r}_{i_1})}{b_i} \quad (7)$$

$$b_i = |\mathbf{r}_{i_2} - \mathbf{r}_{i_1}| = \text{length of the } i\text{th edge (see Table 1)}$$

$$V_e = \text{element's volume}$$

and

$$i = 1, 2, 3, \dots, 6.$$

To understand the physical meaning of the expansion (3) it is necessary to examine the properties of the expansion functions \mathbf{V}_i . We observe that

$$\nabla \cdot \mathbf{V}_i^e = 0 \quad (8)$$

$$\nabla \times \mathbf{V}_i^e = 2\mathbf{g}_i \quad (9)$$

and

$$\mathbf{V}_i^e(\mathbf{r}^j) \cdot \mathbf{e}_j = \delta_{ij} \quad (10)$$

where $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ and \mathbf{r}^j has its tip on the j th edge of the tetrahedron. The last property of \mathbf{V}_i^e points out that E_i^e in the expansion (3) is simply the fields at the i th edge of the tetrahedron. The basis functions \mathbf{V}_i^e then provide a linear transition of the fields from one of the edges to the other and have the important property of being divergenceless. This is quite essential because such a condition is equivalent to stating that $\nabla \cdot \mathbf{E} = 0$ within the element, one of Maxwell's equations which was implied in deriving the wave equation (1).

Substituting (3) into (2) yields

$$F = F_{1V} + F_{2V} + F_S \quad (11)$$

where

$$\begin{aligned} F_{1V} &= \sum_{e=1}^{N_e} F_{1V}^e \\ F_{2V} &= \sum_{e=1}^{N_e} F_{2V}^e \\ F_{1V}^e &= \frac{1}{2} \sum_{j=1}^{N_v} \sum_{i=1}^{N_v} E_j^e E_i^e \iiint_{V_e} \left\{ \frac{1}{\mu_r} (\nabla \times \mathbf{V}_i^e) \cdot (\nabla \times \mathbf{V}_j^e) - k_0^2 \epsilon_r \mathbf{V}_i^e \cdot \mathbf{V}_j^e \right\} dv \\ F_{2V}^e &= \sum_{i=1}^{N_v} E_i^e \iiint_{V_e} \mathbf{V}_i^e \cdot \left[j k_0 Z_0 \mathbf{J}^i - \nabla \times \left(\frac{1}{\mu_r} \mathbf{M}^i \right) \right] dv \\ N_e &= \text{number volume elements} \\ N_v &= 6 = \text{number of tetrahedron edges} \end{aligned}$$

and F_S will be considered later. Taking the variation of $F(E)$ and setting it to zero gives

$$\begin{aligned} \frac{\partial F}{\partial E_i} &= \sum_{e=1}^{N_e} \frac{\partial F^e(E_i^e)}{\partial E_i^e} \\ &= \sum_{e=1}^{N_e} \left\{ \sum_{j=1}^{N_v} E_j^e \iiint_{V_e} \left(\frac{1}{\mu_r} \nabla \times \mathbf{V}_i^e \cdot \nabla \times \mathbf{V}_j^e - k_0^2 \epsilon_r \mathbf{V}_i^e \cdot \mathbf{V}_j^e \right) dv \right. \\ &\quad \left. + \iiint_{V_e} \cdot \left[j k_0 Z_0 \mathbf{J}^i - \nabla \times \left(\frac{1}{\mu_r} \mathbf{M}^i \right) \right] dv \right\} + \frac{\partial F_S}{\partial E_i} = 0 \end{aligned}$$

This can also be rewritten as

$$\frac{\partial F}{\partial E_i} = \sum_{e=1}^{N_e} \frac{\partial F^e}{\partial E_i^e} = \sum_{e=1}^{N_e} \left\{ [A_{ij}^e] \{E_j^e\} + \{K_i^e\} \right\} + \frac{\partial F_S}{\partial E_i} = 0 \quad (12)$$

where $[A_{ij}^e]$ is the volume element matrix whose elements are given by

$$A_{ij}^e = \iiint_{V_e} \left\{ \frac{1}{\mu_r} (\nabla \times \mathbf{V}_i^e) \cdot (\nabla \times \mathbf{V}_j^e) - k_0^2 \epsilon_r \mathbf{V}_i \cdot \mathbf{V}_j^e \right\} dv \quad (13)$$

The elements of the excitation vector $\{K_i^e\}$ are given by

$$K_j^e = \iiint_{V_e} \mathbf{V}_j^e \left[jk_0 Z_0 \mathbf{J}^i - \nabla \times \left(\frac{1}{\mu_r} \mathbf{M}^i \right) \right] dv \quad (14)$$

Explicit values for A_{ij}^e in terms of the geometrical parameters of V_e are worked out in the Appendix A.

Boundary Integral Equation

To solve (12), it is necessary to specify the discrete form of the boundary integral F_s . This can only be accomplished by replacing \mathbf{H} with a functional of \mathbf{E} or alternatively by imposing a condition on S_V which relates \mathbf{E} and \mathbf{H} . Such a condition or equation is supplied from the exterior region boundary equation. In particular we have

$$\begin{aligned} \mathbf{H} &= \mathbf{H}^i + \mathbf{H}^r - j2k_0 Y_0 \iint_{S'_{\text{cav}}} \left(\bar{\mathbf{I}} + \frac{1}{k_0^2} \nabla \nabla \right) G_0(\mathbf{r}, \mathbf{r}') \cdot \mathbf{M}(\mathbf{r}') ds' \\ &= \mathbf{H}^i + \mathbf{H}^r - j2k_0 Y_0 \iint_{S_{\text{cav}}} \left(\bar{\mathbf{I}} + \frac{1}{k_0^2} \nabla \nabla \right) G_0(\mathbf{r}, \mathbf{r}') \cdot (\mathbf{E}(\mathbf{r}') \times \hat{z}) ds' \end{aligned} \quad (15)$$

in which \mathbf{H}^i is the incident field, if any, from the exterior region, \mathbf{H}^r is the reflected field due to \mathbf{H}^i if the aperture is closed, $\bar{\mathbf{I}}$ is the unit dyad and

$$G_0(\mathbf{r}, \mathbf{r}') = \frac{e^{-jk_0|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|} \quad (16)$$

is the free space Green's function with \mathbf{r} and \mathbf{r}' representing the observation and integration points. The factor of two in front of the integral in (15) is due to image theory.

Substituting the above expression (15) into the last integral of (2) yields

$$\begin{aligned} F_s &= jk_0 Z_0 \left\{ \iint_{S_{\text{cav}}} \mathbf{E} \cdot [(\mathbf{H}^i + \mathbf{H}^r) \times \hat{z}] ds \right. \\ &\quad \left. - 2k_0^2 \iint_{S_{\text{cav}}} (\mathbf{E}(\mathbf{r}) \times \hat{z}) \cdot \left\{ \iint_{S_{\text{cav}}} \left(\bar{\mathbf{I}} + \frac{1}{k_0^2} \nabla \nabla \right) G_0(\mathbf{r}, \mathbf{r}') \cdot (\mathbf{E}(\mathbf{r}') \times \hat{z}) ds' \right\} ds \right\} \\ &= jk_0 Z_0 \left\{ \iint_{S_{\text{cav}}} \mathbf{E} \cdot [(\mathbf{H}^i + \mathbf{H}^r) \times \hat{z}] ds \right. \\ &\quad + 2k_0^2 \left[\iint_{S_{\text{cav}}} (\hat{z} \times \mathbf{E}(\mathbf{r})) \cdot \left\{ \iint_{S_{\text{cav}}} (\mathbf{E}(\mathbf{r}') \times \hat{z}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \right. \\ &\quad \left. + 2 \left[\iint_{S_{\text{cav}}} \nabla \cdot (\mathbf{E}(\mathbf{r}) \times \hat{z}) \cdot \left\{ \iint_{S_{\text{cav}}} \nabla' \cdot (\mathbf{E}(\mathbf{r}') \times \hat{z}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \right] \right\} \quad (17) \end{aligned}$$

To discretize these surface integrals we must again expand \mathbf{E} and in this case it is necessary to discretize the surface using triangular elements which coincide with one of the faces associated with the chosen volume elements. To this end we introduce the expansion

$$\mathbf{E} = \sum_{i=1}^{N_{se}} E_i \mathbf{S}_i = \sum_{i, e \in S_{cav}} E_i^e \mathbf{S}_i^e \quad (18)$$

$$\mathbf{S}_i(\mathbf{r}) = \begin{cases} \frac{l_i}{2A_i^+} \hat{\mathbf{n}}_i^+ \times (\mathbf{r} - \mathbf{r}_i^+) & \mathbf{r} \in T_i^+ \\ -\frac{l_i}{2A_i^-} \hat{\mathbf{n}}_i^- \times (\mathbf{r} - \mathbf{r}_i^-) & \mathbf{r} \in T_i^- \end{cases} \quad (19)$$

where A_i^\pm are the areas of the T_i^+ and T_i^- triangles (see Figure 3) and the last sum in (18) is in terms of the global cavity volume element indices. The main properties of \mathbf{S}_i are

$$\nabla_S \cdot \mathbf{S}_i(\mathbf{r}) = 0 \quad \mathbf{r} \in T_i = T_i^+ + T_i^- \quad (20)$$

$$\nabla_S \times \mathbf{S}_i(\mathbf{r}) = \pm \frac{l_i}{A_i^\pm} \hat{\mathbf{n}}_i^\pm \quad \mathbf{r} \in T_i^\pm \quad (21)$$

and

$$\mathbf{S}_i(\mathbf{r}^j) \cdot \mathbf{e}_j = \delta_{ij} \quad (22)$$

where \mathbf{r}^j denotes the vector on the j th edge, and for this application $\hat{\mathbf{n}}_i^\pm = \hat{\mathbf{z}}$. Consequently, E_i in (18) is simply the field at the i th edge shared by the triangle pair, and N_{se} then denotes the number of edges generated in the discretization process of the surface S_{cav} . To solve the system resulting from (12), it is necessary that these edges belong to one of the volume elements which border the aperture of the cavity. Of course, the field at an edge which is located on a perfectly conducting portion of the surface or at the periphery of the aperture must be set to zero a priori and the same must also be done for those edges of the volume elements which border a conductor. It should be clear from the above representation that the field in each surface triangle is given by the linear sum of three basis functions unless that element borders a conductor, in which case one or more of the three coefficients may be zero.

Substituting (18) into (17) and differentiating with respect to E_i^e (this differentiation does not apply to the \mathbf{E} field which is extracted from the integral representation of \mathbf{H}) gives

$$\begin{aligned} \frac{\partial F^S}{\partial E_i^e} &= 2jk_0 Z_0 \iint_{T_i} \mathbf{S}_i^e \cdot (\mathbf{H}^i \times \hat{\mathbf{z}}) ds \quad (23) \\ &+ 2k_0^2 \sum_{j, e \in S_{cav}} E_j^e \iint_{T_i} (\hat{\mathbf{z}} \times \mathbf{S}_i^e(\mathbf{r})) \cdot \left\{ \iint_{T_j} (\mathbf{S}_j^e(\mathbf{r}') \times \hat{\mathbf{z}}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \\ &+ 2 \sum_{j, e \in S_{cav}} E_j^e \iint_{T_i} \nabla_S \cdot (\mathbf{S}_i^e(\mathbf{r}) \times \hat{\mathbf{z}}) \cdot \left\{ \iint_{T_j} \nabla'_S \cdot (\mathbf{S}_j^e(\mathbf{r}') \times \hat{\mathbf{z}}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \end{aligned}$$

where the presence of $T_i = T_i^+ + T_i^-$ denotes integration over the entire triangle pair. Thus, the complete form of the system (12) is

$$\sum_{e=1}^{N_e} \{ [A_{ij}^e] \{ E_j^e \} + \{ K_i^e \} \} + \sum_{i,j,e \in S_{\text{cav}}} \{ [B_{ij}^e] \{ E_j^e \} + \{ L_i^e \} \} = 0 \quad (24)$$

with

$$\begin{aligned} B_{ij}^e = & +2k_0^2 \iint_{T_i} \hat{z} \times \mathbf{S}_i^e(\mathbf{r}) \cdot \left\{ \iint_{T_j} (\mathbf{S}_j^e(\mathbf{r}') \times \hat{z}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \\ & + 2 \iint_{T_i} \nabla_S \cdot (\mathbf{S}_i^e(\mathbf{r}) \times \hat{z}) \cdot \left\{ \iint_{T_j} \nabla'_S \cdot (\mathbf{S}_j^e(\mathbf{r}') \times \hat{z}) G_0(\mathbf{r}, \mathbf{r}') ds' \right\} ds \end{aligned} \quad (25)$$

and

$$L_i^e = 2jk_0 Z_0 \iint_{T_i} \mathbf{S}_i^e \cdot (\mathbf{H}^i \times \hat{z}) ds \quad (26)$$

are the excitation elements which are non-zero for scattering computation. Note that the matrix $[A_{ij}^e]$ is sparse and banded, but $[B_{ij}^e]$ is full. However, because it only accounts for the interaction among the surface elements, it is a relatively small matrix. It should not therefore appreciably impact the memory requirements to any degree.

The computation of the surface matrix elements B_{ij}^e must be done carefully because $G_0(\mathbf{r}, \mathbf{r}')$ is singular when $\mathbf{r} = \mathbf{r}'$ and this occurs when $i = j$. To facilitate its evaluation we rewrite it as

$$B_{ij} = B_{ij}^{++} + B_{ij}^{+-} + B_{ij}^{-+} + B_{ij}^{--} \quad (27)$$

where

$$\begin{aligned} B_{ij}^{pq} = & -\frac{k_0^2 l_i l_j}{8\pi A_i^p A_j^q} \iint_{T_i^p} \iint_{T_j^q} \bar{\rho}_i^p(\mathbf{r}) \cdot \bar{\rho}_j^q(\mathbf{r}') \frac{e^{-jkR}}{R} ds' ds \\ & + \frac{l_i l_j}{2\pi A_i^p A_j^q} \epsilon_{pq} \iint_{T_i^p} \iint_{T_j^q} \frac{e^{-jkR}}{R} ds' ds \end{aligned} \quad (28)$$

in which $R = |\mathbf{r} - \mathbf{r}'|$ and the superscripts p and q denote either $+$ or $-$. Also,

$$\epsilon_{pq} = \begin{cases} 1 & p = q \\ -1 & p \neq q \end{cases}$$

and

$$\bar{\rho}_i^\pm(\mathbf{r}) = \pm(\mathbf{r} - \mathbf{r}_i^\pm) \quad (29)$$

where \mathbf{r}_i^\pm is the position vector of the T_i^\pm triangle vertex opposite to the i th (shared) edge between the two triangles T_i^+ and T_i^- . To evaluate B_{ij}^{pq} , it is convenient to introduce local coordinate variables. The area coordinates have been found useful for this purpose. Referring to the triangle in Figure 4 we denote its vertices by \mathbf{r}_n ($n = 1, 2$ or 3), its edge vectors by \mathbf{l}_n and $\bar{\rho}_n = \mathbf{r} - \mathbf{r}_n$ coinciding with the definition (29) if this is a $+$ triangle. The vectors $\bar{\rho}_n$ drawn from the triangle vertices to a

position \mathbf{r} within the triangle separate that triangle in three smaller ones whose area is denoted by A_n . It is then readily seen that $\bar{\rho}_n$ can be written as

$$\bar{\rho}_n = \frac{A_{n+1}}{A} \mathbf{l}_{n-1} - \frac{A_{n-1}}{A} \mathbf{l}_{n+1} = \xi_{n+1} \mathbf{l}_{n-1} - \xi_{n-1} \mathbf{l}_{n+1} \quad (30)$$

where $A = \sum_{n=1}^3 A_n$ and $\xi_{n\pm 1}$ are referred to as the area coordinates. By varying these from 0 to unity, we can generate all possible positions of ρ_n within the triangle's surface. In terms of the area coordinates we also find that the global position vector can be written as

$$\mathbf{r} = \xi_{n-1} \mathbf{r}_{n-1} + \xi_{n+1} \mathbf{r}_{n+1} + \xi_n \mathbf{r}_n \quad (31)$$

with $\sum_{n=1}^3 \xi_n = 1$. Alternatively, since $\mathbf{r}_{n\pm 1} - \mathbf{r}_{n\mp 1} = \mp \mathbf{l}_n$ and $\mathbf{r}_{n\mp 1} - \mathbf{r}_n = \mp \mathbf{l}_{n\pm 1}$, we can express \mathbf{r} as

$$\mathbf{r} = \mp (\xi_{n\pm 1} \mathbf{l}_n - \xi_n \mathbf{l}_{n\pm 1}) + \mathbf{r}_{n\mp 1} \quad (32)$$

Also, we can show that the differential area in terms of these area coordinates is

$$\begin{aligned} ds &= \left| \left(\frac{\partial \mathbf{r}}{\partial \xi_{n\pm 1}} \times \frac{\partial \mathbf{r}}{\partial \xi_n} \right) \right| d\xi_{n\pm 1} d\xi_n = |\mathbf{l}_n \times \mathbf{l}_{n\pm 1}| d\xi_{n\pm 1} d\xi_n \\ &= 2A d\xi_{n\pm 1} d\xi_n \end{aligned} \quad (33)$$

The integrals in (28) can now be readily rewritten in terms of the new coordinates ξ_n . Before doing so, though, we first simplify it by employing midpoint integration for the T_i^{pq} integrals. This gives

$$\begin{aligned} B_{ij}^{pq} &= -k_0^2 \frac{l_i l_j}{8\pi A_j^q} \bar{\rho}_i^p(\mathbf{r}_c) \cdot \iint_{T_j^q} \bar{\rho}_j^q(\mathbf{r}') \frac{e^{-jkR_c}}{R_c} ds' \\ &\quad + \frac{l_i l_j}{2\pi A_j^q} \epsilon_{pq} \iint_{T_j^q} \frac{e^{-jkR_c}}{R_c} ds' \end{aligned} \quad (34)$$

in which $R_c = |\mathbf{r}_c - \mathbf{r}'|$ and \mathbf{r}_c is the position vector whose tip is at the centroid of the T_i^p triangles. The integrals in (34) can now be evaluated numerically and analytically as described in (6) and (7).

Code Implementation and Validation

Based on the presented formulation, a computer code was written which is listed in Appendix C. The code relies on a preprocessor to supply all required information pertaining to the geometry, mesh discretization and material parameters of the cavity-backed antenna. In particular the following tables of data must be supplied to the code (see Appendix B):

Table 1

Volume element No., Global edge No., Pair of global node numbers forming the edge, Element dielectric constants (six line entries are required per volume element)

Table 2

Global edge No., Unnormalized vector coordinates joining the nodes forming the edge.

Table 3

Volume element No., Global node Nos forming the element, element's dielectric constant.

Table 4

Global node No., (x, y, z) coordinates of the node.

Table 5

List of edge Nos on the perfectly conducting walls/cavity surface.

Table 6

Volume elements bordering open surface (global Nos.), Nodes of triangle coinciding with surface, corresponding edge Nos.

Table 7

List of edge Nos. at the boundary line joining the PEC and open/dielectric surface of the cavity.

As listed above, some redundancy exists among the information provided in the Tables. This is only done to simplify the processor and in the future we shall consider a more concise input data list. Regardless of this, it should be clear that the user of the code must generate these tables on his/her own from the universal tables outputted by the employed commercial mesh generation package.

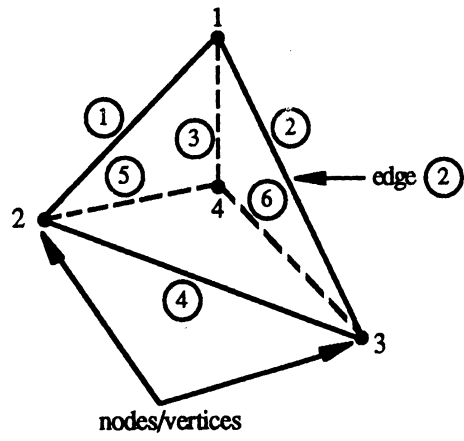
From the supplied generated data the analysis portion of the code (processor) which we have written computes the elements A_{ij} , K_i , B_{ij} and L_i . As noted earlier, the $[A]$ matrix is large but because it is very sparse and banded, its storage and fill time is very small. This is actually the main advantage of the methodology along with the geometrical adaptability of the tetrahedral elements. The boundary matrix $[B]$ is unfortunately full and its computation is cumbersome, consuming a large portion of the code. For an efficient solution of the resulting system, it is necessary to force this matrix to be Toeplitz requiring that all surface triangles be identical. However, such requirement cannot be imposed (externally) on most commercial mesh generation packages. A simple approach is to allow the preprocessor to generate only those volume elements up to one cell below S_{cav} . The last layer of volume elements, bordering S_{cav} is then appended externally.

At the moment, the solution of the system is done without any provisions to force the $[B]$ matrix to be Toeplitz in form. This was done as a first step in the development of the final code since our initial and foremost goal was to test the validity of the code. The scattering patterns given in figures 5 and 6 were generated with this version of the code. The comparison with other numerical data clearly demonstrates the validity of the formulation.

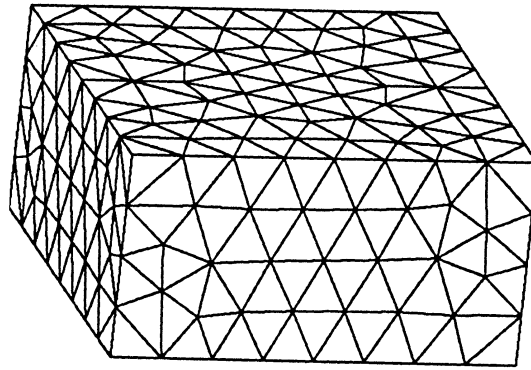
In the next few months we shall concentrate on the development of a more efficient code as described above. More importantly, we shall consider the modeling of practical antenna geometries and arrays. The new code will also allow input impedance computations and modeling of feeds, lumped loads and distributed/resistive loads.

REFERENCES

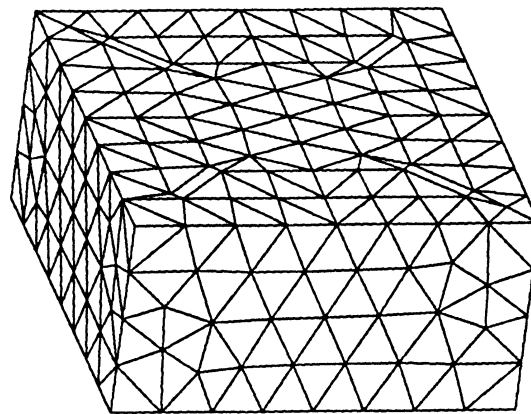
- [1] J. M. Jin and J. L. Volakis, "TE scattering by an inhomogeneously filled aperture in a thick conducting plane," *IEEE Trans. Antennas Propagat.*, vol. 38, pp. 1280-1286, Aug. 1990.
- [2] J. M. Jin and J. L. Volakis, "TM scattering by an inhomogeneously filled aperture in a thick conducting plane," *Proc. Inst. Elec. Eng., part H*, vol. 137, pp. 153-159, Aug. 1990.
- [3] J. M. Jin and J. L. Volakis, "A finite element – boundary integral formulation for scattering by three-dimensional cavity-backed apertures," *IEEE Trans. Antennas Propagat.*, Nov. 1991
- [4] J.L. Volakis and K.Barkeshli, " Application of the Conjugate Gradient FFT Method to Radiation and scattering," in *Application of Iterative Methods to Electromagnetics and Signal Processing*, ed.T. Sarkar, Elsevier Pub. Co., pp.159-240,1991
- [5] J.M. Jin and J.L. Volakis, "A Biconjugate Gradient Method for Scattering by Plates," to appear in *Electromagnetics*, 1992
- [6] S.M. Rao, D.R. Wilton and A.W. Glisson, "Electromagnetic Scattering by Surfaces of Arbitrary Shape," *IEEE Trans. Antennas Propagat.*, Vol. AP-30, pp.409-418, May 1982
- [7] D.R. Wilton, S.M. Rao, A.W. Glisson and D.H. Schaubert, "Potential Integrals for Uniform and Linear Source Distributions on Polygonal and Polyhedral Domains," *IEEE Trans. Antennas Propagat.*, Vol. AP-32, pp.276-281, March 1984



(a).



(b).



(c).

Figure 1. (a).Tetrahedron Geometry (b). Mesh with Circular Patch at aperture (c). Mesh with four slots at aperture.

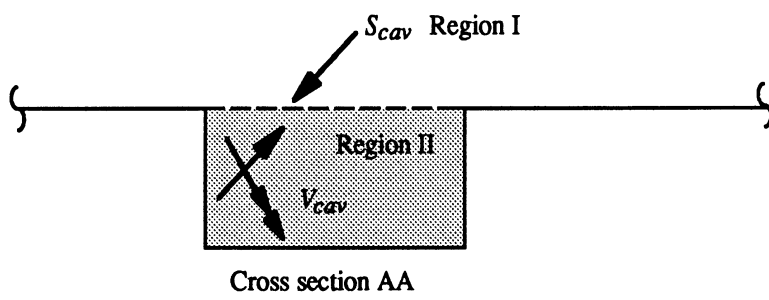
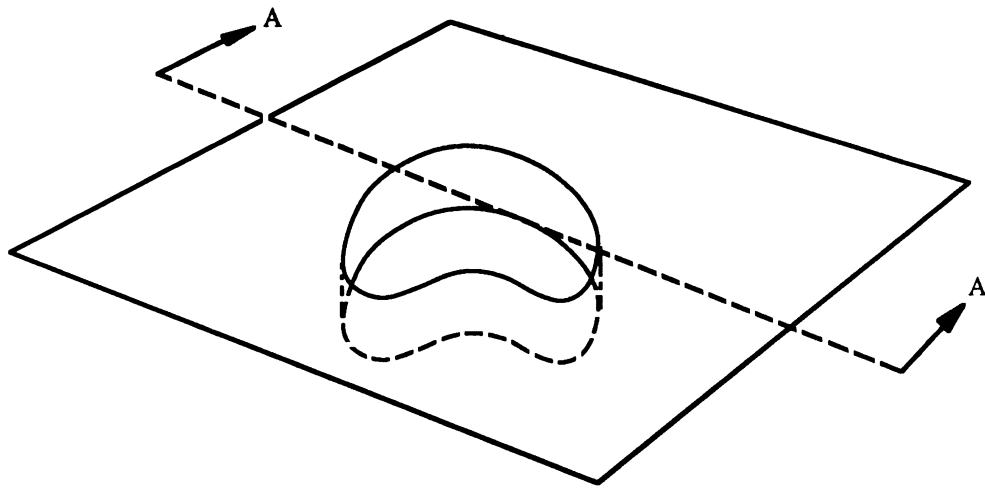


Figure 2. Cavity Geometry Recessed in a Ground Plane

TABLE 1

Edge Number	Vertex Number	
	i_1	i_2
①	1	2
②	1	3
③	1	4
④	2	3
⑤	4	2
⑥	3	4

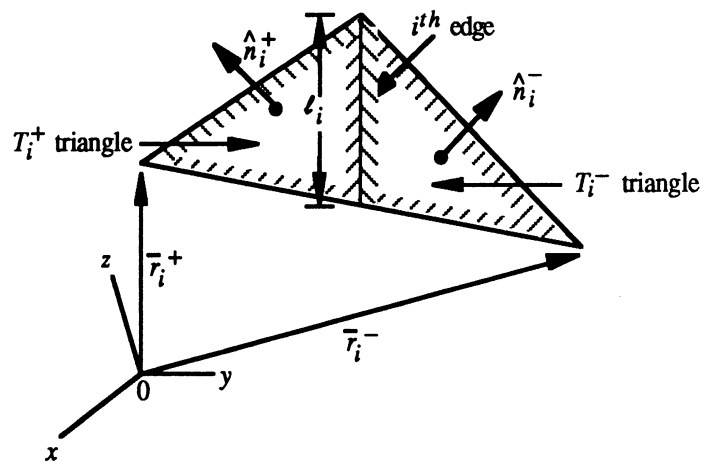


Figure 3. Pair of triangles sharing the i^{th} edge (i^{th} pair)

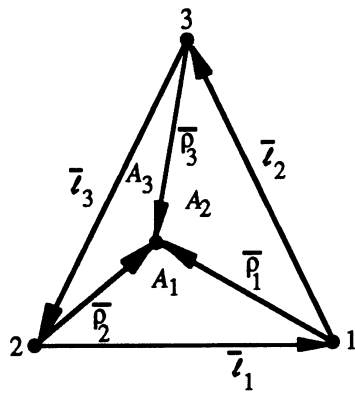


Figure 4. Illustration of the local vectors for a triangle on S_{cav}

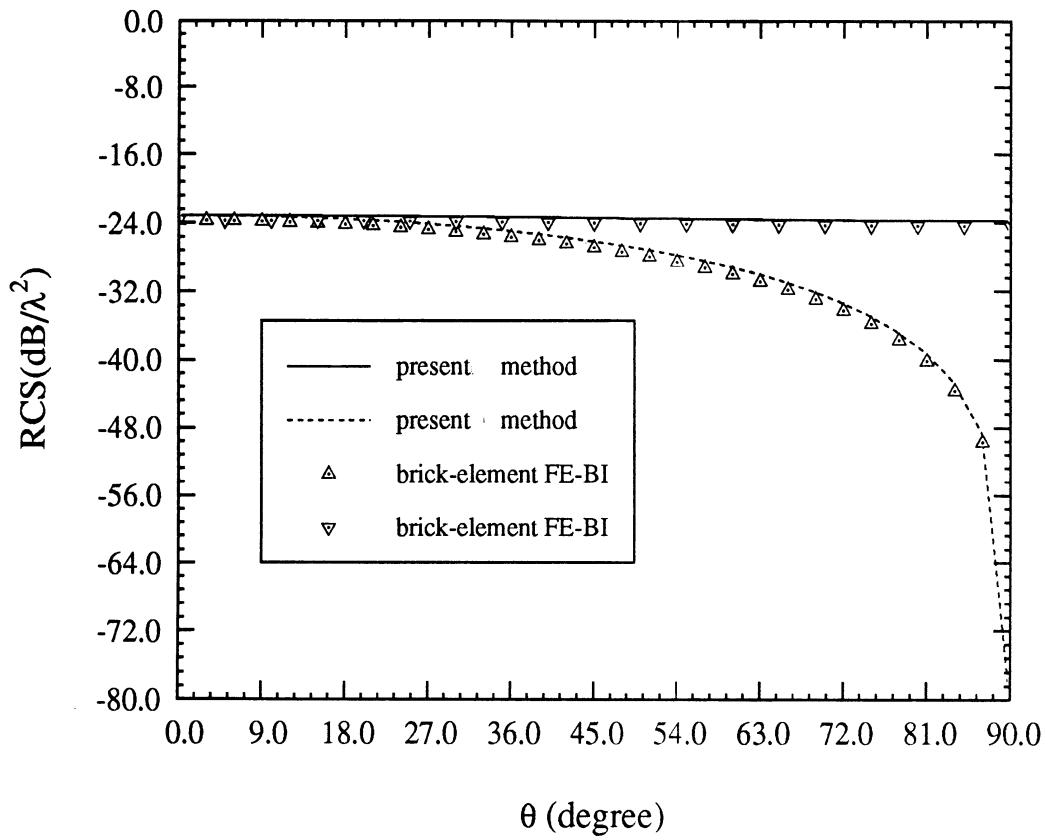


Figure 5. Comparison of bistatic RCS of the fields scattered by $0.2\lambda \times 0.2\lambda \times 0.1\lambda$ rectangular cavity of empty filling, with the incident field of E_z^i -pol and $\theta^i = 45^\circ$.

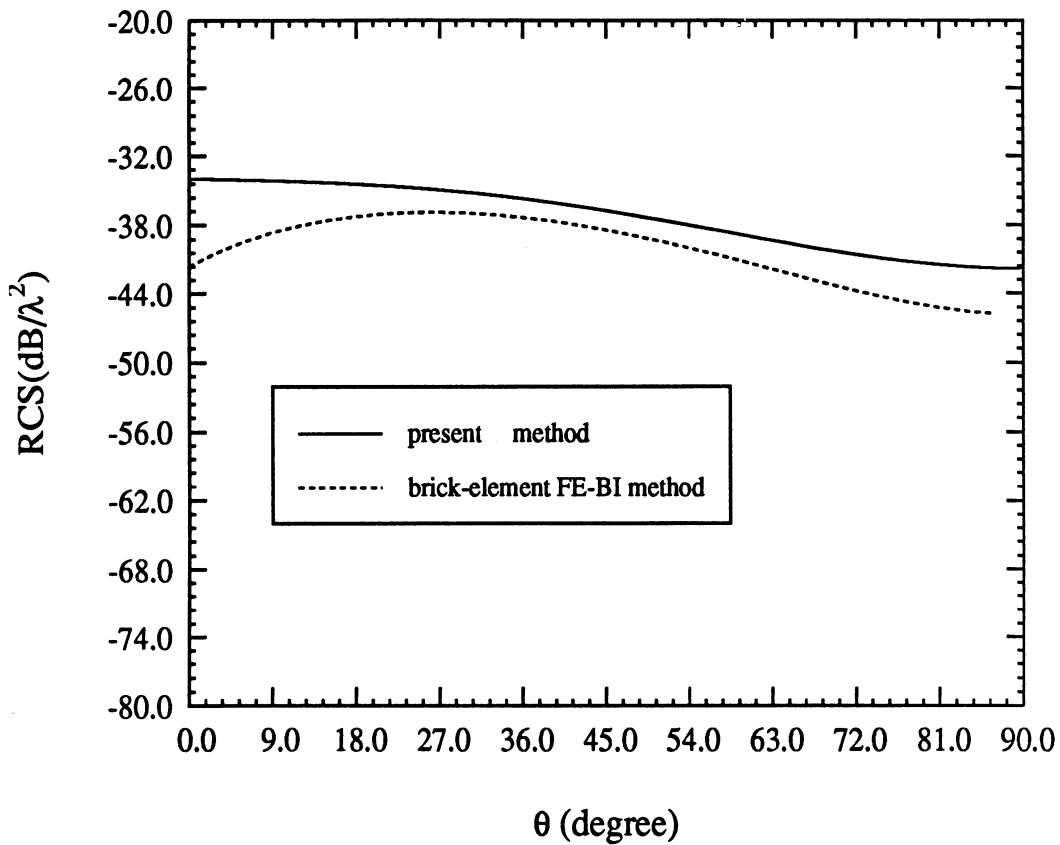


Figure 6. Demonstration of the scattering by two different patch geometries at the aperture center of $0.8\lambda \times 0.8\lambda \times 0.02\lambda$ cavity with the same incidence as described in figure 5. The dashed curve represents the bistatic RCS pattern by a rectangular patch of an area $0.32\lambda^2$, calculated from the brick-element FE-BI code [3]. The upper curve denotes for demonstration the scattering by a circular patch of an area $0.28\lambda^2$.

Appendix A

Computation of Matrix Elements A_{ij}

Appendix A

The derivation of the $[A]$ matrix elements in (12) amounts to evaluating the integral in (13). We have

$$\int_{V_e} \frac{1}{\mu_r} (\nabla \times \mathbf{V}_i^e) \cdot (\nabla \times \mathbf{V}_j^e) = \frac{4}{\mu_r} \mathbf{g}_i \cdot \mathbf{g}_j V_e \quad (1)$$

since from (9) $\nabla \times \mathbf{V}_i^e = 2\mathbf{g}_i$. Also, from (4) we have

$$\begin{aligned} \epsilon_r \int_{V_e} \mathbf{V}_i^e \cdot \mathbf{V}_j^e dv &= \epsilon_r \int_{V_e} \{(\mathbf{f}_i \cdot \mathbf{f}_j) + (\mathbf{r} \cdot \mathbf{D}) + (\mathbf{g}_i \times \mathbf{r}) \cdot (\mathbf{g}_j \times \mathbf{r})\} dv \quad (2) \\ &= \epsilon_r (I_1 + I_2 + I_3) \end{aligned}$$

where

$$\mathbf{D} = (\mathbf{f}_i \times \mathbf{g}_j) + (\mathbf{f}_j \times \mathbf{g}_i)$$

and

$$I_1 = \int_{V_e} \mathbf{f}_i \cdot \mathbf{f}_j dv \quad (3)$$

$$I_2 = \int_{V_e} \mathbf{r} \cdot \mathbf{D} dv \quad (4)$$

$$I_3 = \int_{V_e} (\mathbf{g}_i \times \mathbf{r}) \cdot (\mathbf{g}_j \times \mathbf{r}) dv \quad (5)$$

Since \mathbf{f} is a constant vector, I_1 reduces to

$$I_1 = \mathbf{f}_i \cdot \mathbf{f}_j V_e \quad (6)$$

Since

$$\begin{aligned} x &= \sum_{i=1}^4 L_i x_i \\ y &= \sum_{i=1}^4 L_i y_i \\ z &= \sum_{i=1}^4 L_i z_i \end{aligned}$$

where L_i are the shape functions for the tetrahedral element and x_i, y_i, z_i ($i = 1, \dots, 4$) denote the x, y and z co-ordinates of the vertices of the tetrahedral element. Using the standard formula for volume integration within a tetrahedral element and simplifying, we have

$$I_2 = \frac{V_e}{4} \left[D_x \sum_{i=1}^4 x_i + D_y \sum_{i=1}^4 y_i + D_z \sum_{i=1}^4 z_i \right] \quad (7)$$

where D_m is the m th component of \mathbf{D} . The evaluation of I_3 can be simplified by the use of basic vector identities. Therefore,

$$\begin{aligned}
I_3 &= \mathbf{g}_i \cdot \mathbf{g}_j \int_{V_e} |\mathbf{r}|^2 dv - \int_{V_e} (\mathbf{g}_i \cdot \mathbf{r})(\mathbf{g}_j \cdot \mathbf{r}) dv \\
&= (g_{iy}g_{jy} + g_{iz}g_{jz}) \int_{V_e} x^2 dv + (g_{ix}g_{jx} + g_{iz}g_{jz}) \int_{V_e} y^2 dv + (g_{ix}g_{jx} + g_{iy}g_{jy}) \int_{V_e} z^2 dv \\
&\quad - (g_{ix}g_{jy} + g_{jx}g_{iy}) \int_{V_e} xy dv - (g_{ix}g_{jz} + g_{jx}g_{iz}) \int_{V_e} zx dv - (g_{iy}g_{jz} + g_{jy}g_{iz}) \int_{V_e} yz dv
\end{aligned} \tag{8}$$

where g_{im} represents the m th component of the vector \mathbf{g}_i . Each of the integrals above can be easily evaluated analytically and the result expressed in the following general form:

$$\int_{V_e} a_l a_m dv = \frac{V_e}{20} \left[\sum_{i=1}^4 a_{li} a_{mi} + \sum_{i=1}^4 a_{li} \sum_{i=1}^4 a_{mi} \right] \tag{9}$$

where $l, m = 1, \dots, 3$ and a_1 represents the variable x , a_2 stands for the variable y and a_3 denotes the variable z .

Appendix B

Example Input Data Files/Tables

*** TABLE 1 ***

1.
20
34

:element #
>:edge # of element
:node # of edge
l:node # of edge
:dielectric constant of medium

a b c d e

1 1 1 3 1
1 2 3 18 1
1 3 3 9 1
1 4 1 18 1
1 5 1 9 1
1 6 9 18 1
2 1 1 3 1
2 7 3 15 1
2 2 3 18 1
2 8 1 15 1
2 4 1 18 1
2 9 15 18 1
3 10 2 9 1
3 11 9 11 1
3 12 4 9 1
3 13 2 11 1
3 14 2 4 1
3 15 4 11 1
4 16 5 6 1
4 17 6 18 1
4 18 6 13 1
4 19 5 18 1
4 20 5 13 1
4 21 13 18 1
5 16 5 6 1
5 22 6 11 1
5 17 6 18 1
5 23 5 11 1
5 19 5 18 1
5 24 11 18 1
6 25 7 13 1
6 26 13 15 1
6 27 8 13 1
6 28 7 15 1
6 29 7 8 1
6 30 8 15 1
7 31 3 10 1
7 32 10 18 1
7 33 9 10 1
7 2 3 18 1
7 3 3 9 1
7 6 9 18 1

8 34 4 12 1
8 35 12 17 1
8 36 11 12 1
8 37 4 17 1
8 15 4 11 1
8 38 11 17 1
9 39 6 19 1
9 22 6 11 1
9 40 6 12 1
9 41 11 19 1
9 42 12 19 1
9 36 11 12 1
10 42 12 19 1
10 43 12 14 1
10 40 6 12 1
10 44 14 19 1
10 39 6 19 1
10 45 6 14 1
11 44 14 19 1
11 43 12 14 1
11 46 14 17 1
11 42 12 19 1
11 47 17 19 1
11 35 12 17 1
12 39 6 19 1
12 17 6 18 1
12 22 6 11 1
12 48 18 19 1
12 41 11 19 1
12 24 11 18 1
13 39 6 19 1
13 45 6 14 1
13 17 6 18 1
13 44 14 19 1
13 48 18 19 1
13 49 14 18 1
14 48 18 19 1
14 49 14 18 1
14 50 17 18 1
14 44 14 19 1
14 47 17 19 1
14 46 14 17 1
15 41 11 19 1
15 24 11 18 1
15 38 11 17 1
15 48 18 19 1
15 47 17 19 1
15 50 17 18 1
16 42 12 19 1
16 36 11 12 1
16 35 12 17 1
16 41 11 19 1
16 47 17 19 1
16 38 11 17 1
17 15 4 11 1
17 51 4 10 1
17 37 4 17 1
17 52 10 11 1
17 38 11 17 1

17 53 10 17 1
18 11 9 11 1
18 33 9 10 1
18 12 4 9 1
18 52 10 11 1
18 15 4 11 1
18 51 4 10 1
19 24 11 18 1
19 32 10 18 1
19 6 9 18 1
19 52 10 11 1
19 11 9 11 1
19 33 9 10 1
20 38 11 17 1
20 53 10 17 1
20 50 17 18 1
20 52 10 11 1
20 24 11 18 1
20 32 10 18 1
21 45 6 14 1
21 49 14 18 1
21 54 13 14 1
21 17 6 18 1
21 18 6 13 1
21 21 13 18 1
22 55 8 16 1
22 56 16 17 1
22 57 15 16 1
22 58 8 17 1
22 30 8 15 1
22 59 15 17 1
23 60 3 20 1
23 7 3 15 1
23 61 3 16 1
23 62 15 20 1
23 63 16 20 1
23 57 15 16 1
24 63 16 20 1
24 64 10 16 1
24 61 3 16 1
24 65 10 20 1
24 60 3 20 1
24 31 3 10 1
25 65 10 20 1
25 64 10 16 1
25 53 10 17 1
25 63 16 20 1
25 66 17 20 1
25 56 16 17 1
26 60 3 20 1
26 2 3 18 1
26 7 3 15 1
26 67 18 20 1
26 62 15 20 1
26 9 15 18 1
27 60 3 20 1
27 31 3 10 1
27 2 3 18 1
27 65 10 20 1

```

27 67 18 20 1
27 32 10 18 1
28 65 10 20 1
28 53 10 17 1
28 32 10 18 1
28 66 17 20 1
28 67 18 20 1
28 50 17 18 1
29 62 15 20 1
29 9 15 18 1
29 59 15 17 1
29 67 18 20 1
29 66 17 20 1
29 50 17 18 1
30 63 16 20 1
30 57 15 16 1
30 56 16 17 1
30 62 15 20 1
30 66 17 20 1
30 59 15 17 1
31 30 8 15 1
31 68 8 14 1
31 58 8 17 1
31 69 14 15 1
31 59 15 17 1
31 46 14 17 1
32 26 13 15 1
32 54 13 14 1
32 27 8 13 1
32 69 14 15 1
32 30 8 15 1
32 68 8 14 1
33 9 15 18 1
33 49 14 18 1
33 21 13 18 1
33 69 14 15 1
33 26 13 15 1
33 54 13 14 1
34 59 15 17 1
34 46 14 17 1
34 50 17 18 1
34 69 14 15 1
34 9 15 18 1
34 49 14 18 1

```

*** TABLE 2 ***

a:edge #
b:(x2-x1)
c:(y2-y1)
d:(z2-z1)

a	b	c	d
1	0.0000000	0.0000000	-0.1000000
2	0.1000000	0.1000000	0.1000000
3	0.0000000	0.1000000	0.1000000

```

63 -3.3333339E-02 5.0000001E-02 3.3333331E-02
64 0.1000000 -0.1000000 0.0000000
65 6.6666663E-02 -5.0000001E-02 3.3333331E-02
66 -3.3333339E-02 -5.0000001E-02 3.3333331E-02
67 -3.3333339E-02 -5.0000001E-02 -6.6666670E-02
68 0.0000000 0.1000000 0.0000000
69 -0.1000000 -0.1000000 0.1000000

```

*** TABLE 3 ***

a:element #
b:node of the element
c:node of the element
d:node of the element
e:node of the element
f:dielectric constant of the element

a	b	c	d	e	f
1	3	1	18	9	1
2	3	1	15	18	1
3	9	2	11	4	1
4	6	5	18	13	1
5	6	5	11	18	1
6	13	7	15	8	1
7	10	3	18	9	1
8	12	4	17	11	1
9	6	19	11	12	1
10	12	19	14	6	1
11	14	19	12	17	1
12	6	19	18	11	1
13	6	19	14	18	1
14	18	19	14	17	1
15	11	19	18	17	1
16	12	19	11	17	1
17	4	11	10	17	1
18	9	11	10	4	1
19	18	11	10	9	1
20	17	11	10	18	1
21	14	6	18	13	1
22	16	8	17	15	1
23	3	20	15	16	1
24	16	20	10	3	1
25	10	20	16	17	1
26	3	20	18	15	1
27	3	20	10	18	1
28	10	20	17	18	1
29	15	20	18	17	1
30	16	20	15	17	1
31	8	15	14	17	1
32	13	15	14	8	1
33	18	15	14	13	1
34	17	15	14	18	1

***TABLE 4 ***

4 0.1000000 0.1000000 0.0000000
5 0.0000000 0.1000000 0.0000000
6 0.1000000 0.0000000 0.0000000
7 0.1000000 0.0000000 0.1000000
8 0.1000000 0.0000000 0.0000000
9 0.0000000 0.1000000 0.0000000
10 0.0000000 -0.1000000 0.0000000
11 0.1000000 0.1000000 0.0000000
12 0.0000000 -0.1000000 0.1000000
13 0.1000000 0.0000000 0.0000000
14 0.0000000 0.0000000 -0.1000000
15 0.1000000 0.0000000 0.1000000
16 0.0000000 0.0000000 -0.1000000
17 -0.1000000 -0.1000000 0.1000000
18 0.0000000 -0.1000000 0.1000000
19 -0.1000000 -0.1000000 0.0000000
20 0.0000000 -0.1000000 0.0000000
21 -0.1000000 0.0000000 0.0000000
22 -0.1000000 0.0000000 0.1000000
23 -0.1000000 0.0000000 0.0000000
24 0.0000000 -0.1000000 0.0000000
25 0.0000000 0.1000000 0.0000000
26 -0.1000000 -0.1000000 0.0000000
27 0.0000000 0.1000000 0.1000000
28 -0.1000000 0.0000000 0.0000000
29 0.0000000 0.0000000 -0.1000000
30 -0.1000000 0.0000000 0.1000000
31 0.0000000 0.1000000 0.0000000
32 0.1000000 0.0000000 0.1000000
33 0.0000000 0.0000000 -0.1000000
34 0.1000000 0.0000000 0.0000000
35 0.0000000 -0.1000000 0.0000000
36 0.0000000 0.0000000 -0.1000000
37 0.1000000 -0.1000000 0.0000000
38 0.0000000 -0.1000000 -0.1000000
39 -6.6666663E-02 -5.0000001E-02 3.3333331E-02
40 -0.1000000 0.0000000 0.0000000
41 3.3333339E-02 -5.0000001E-02 -6.6666670E-02
42 3.3333339E-02 -5.0000001E-02 3.3333331E-02
43 0.1000000 -0.1000000 0.0000000
44 -6.6666663E-02 5.0000001E-02 3.3333331E-02
45 0.0000000 -0.1000000 0.0000000
46 -0.1000000 0.0000000 0.0000000
47 3.3333339E-02 5.0000001E-02 3.3333331E-02
48 3.3333339E-02 5.0000001E-02 -6.6666670E-02
49 -0.1000000 0.0000000 0.1000000
50 0.0000000 0.0000000 0.1000000
51 0.0000000 -0.1000000 0.0000000
52 0.1000000 0.1000000 0.1000000
53 0.1000000 0.0000000 0.0000000
54 0.0000000 0.0000000 -0.1000000
55 -0.1000000 0.0000000 0.0000000
56 0.0000000 0.1000000 0.0000000
57 0.0000000 0.0000000 -0.1000000
58 -0.1000000 0.1000000 0.0000000
59 0.0000000 0.1000000 -0.1000000
60 6.6666663E-02 5.0000001E-02 3.3333331E-02
61 0.1000000 0.0000000 0.0000000
62 -3.3333339E-02 5.0000001E-02 -6.6666670E-02

```

:node #
:x coordinate of the node
:y coordinate of the node
:z coordinate of the node

```

a	x	y	z
1	-1.000000E-01	-1.000000E-01	0.000000E+00
2	-1.000000E-01	1.000000E-01	0.000000E+00
3	-1.000000E-01	-1.000000E-01	-1.000000E-01
4	-1.000000E-01	1.000000E-01	-1.000000E-01
5	1.000000E-01	1.000000E-01	0.000000E+00
6	1.000000E-01	1.000000E-01	-1.000000E-01
7	1.000000E-01	-1.000000E-01	0.000000E+00
8	1.000000E-01	-1.000000E-01	-1.000000E-01
9	-1.000000E-01	0.000000E+00	0.000000E+00
10	-1.000000E-01	0.000000E+00	-1.000000E-01
11	0.000000E+00	1.000000E-01	0.000000E+00
12	0.000000E+00	1.000000E-01	-1.000000E-01
13	1.000000E-01	0.000000E+00	0.000000E+00
14	1.000000E-01	0.000000E+00	-1.000000E-01
15	0.000000E+00	-1.000000E-01	0.000000E+00
16	0.000000E+00	-1.000000E-01	-1.000000E-01
17	0.000000E+00	0.000000E+00	-1.000000E-01
18	0.000000E+00	0.000000E+00	0.000000E+00
19	3.333334E-02	5.000000E-02	-6.666667E-02
20	-3.333334E-02	-5.000000E-02	-6.666667E-02

** TABLE 5 ***

n-pec edge #

1
5
14
10
3
31
12
51
33
13
15
34
16
23
22
40
36
20
18
45
29
25
27
68
54
8

7
61
28
30
55
57
37
58
64
53
43
35
46
56

*** TABLE 6 ***

a:element #
b1,b2,b3:edge # of the element
c1,c2:node # of the edge b1
d1,d2:node # of the edge b2
e1,e2:node # of the edge b3
(in c1,c2,d1,d2,e1 and e2, there are just three
different numbers corresponding to three node #).

a b1 c1 c2 b2 d1 d2 b3 e1 e2

3 10 2 9 13 2 11 11 9 11
6 25 7 13 28 7 15 26 13 15
1 5 1 9 4 1 18 6 9 18
2 8 1 15 4 1 18 9 15 18
4 20 5 13 19 5 18 21 13 18
5 23 5 11 19 5 18 24 11 18
19 11 9 11 6 9 18 24 11 18
33 26 13 15 21 13 18 9 15 18

*** TABLE 7 ***

node numbere on outer boundary of dielectric surface

1
2
5
7
9
11
13
15

Appendix C
Computer Code Listing

```

=====
C  PROCESSE A UNIVERSAL FILE OBTAINED FROM IDEAS AND CONVERTS THE NODAL INFO
C  TO EDGE INFO NEEDED FOR CONSTRUCTING AN EDGE-BASED THREE-DIMENSIONAL FINITE
C  ELEMENT MESH USING TETRAHEDRAL ELEMENTS.
C(1)STORES THE NODE NUMBERS AND RESPECTIVE NODAL COORDINATES IN 'NODDY'
C(2)STORES THE ELEMENT NUMBERS AND CORRESPONDING NODES IN 'ELNO'
C(3)PROCESSES 'NODDY' AND 'ELNO' AND STORES THE EDGE NOS. AND NODAL CONNECTIONS
C  IN 'GLOB' AND EDGE NOS. WITH CORRESPONDING EDGE VECTOR IN 'EDGY'
C  NOTE: EDGE OVERLAPS ARE TAKEN CARE OF.
C  STORAGE LIMIT: 800 NODES, 3000 ELEMENTS, 4300 EDGES
=====

```

```

PROGRAM UNV_FILE_PROCESSOR
CHARACTER STRING*80, YASTRN*40, UNV*20
INTEGER N1(3000,4),TAB(4300,2),NN(100),TR(3000,10),NUN(3000)
INTEGER BC(3000,3),EDGV(18000,3),MAT(3000),ZE(10)
INTEGER NC(10), PEDGE(1000)
REAL X(800),Y(800),Z(800)
COMMON /BANK/N1,X,Y,Z,NE,MAT
COMMON /DBASE/EDGV
COMMON /LOCAL/NCOUNT
COMMON /PECEDGE/PEDGE,NSURF
DO I=1,8
ZE(I)=0
ENDDO
OPEN(8,FILE='ESURFC')
OPEN(9,FILE='ESURFD')
WRITE(6,*)'NAME OF UNIVERSAL FILE ?'
READ(5, '(A)')UNV
WRITE(6,*)'INPUT # OF UNCONNECTED PEC SECTIONS(<10)''
READ(5,*)NPC
OPEN(1,FILE=UNV)
OPEN(2,FILE='ENODDY')
OPEN(3,FILE='ELNO')
OPEN(4,FILE='EGLOB')
OPEN(7,FILE='EDGY')
OPEN(10,FILE='OUTB')

```

C....PROCESSING UNIVERSAL FILE FOR INFO ON VARIOUS PARAMETERS

```

IT=1
15 READ(1, '(A)')STRING
L=0
IF (STRING(4:6).EQ.'151') THEN
WRITE(6,*)'ENCOUNTERED HEADER'
GO TO 15
ELSEIF (STRING(4:6).EQ.'747') THEN
2 READ(1, '(A)')STRING
IF (STRING(5:6).NE.'-1') THEN
IF (STRING(9:9).NE.' ') THEN
WRITE(4,*)STRING(2:13)
DO IK=1,4
READ(1, '(A)')STRING
ENDDO
ENDIF
GO TO 2
ENDIF

```

C....PROCESSING NODAL INFO; DATA STORED IN 'NODDY'
ELSEIF (STRING(4:6).EQ.' 15') THEN

```

READ(1,'(A)')STRING
IF (STRING(5:6).NE.'-1') THEN
WRITE(2,*)STRING(7:10),' ',STRING(41:53),' ',STRING(54:66),' '
1
    L=L+1
    GO TO 5
ENDIF
L1=L
WRITE(6,*)'THERE ARE ',L,' NODES.'
WRITE(4,*)L
...PROCESSING ELEMENT INFO; DATA STORED IN 'ELNO'
ELSEIF (STRING(4:6).EQ.'71') THEN
READ(1,'(A)')STRING
IF (STRING(5:6).NE.'-1') THEN
READ(1,'(A)')YASTRN
WRITE(3,*)STRING(7:10),' ',YASTRN(7:10),' ',YASTRN(17:20),
1
    ' ',YASTRN(27:30),' ',YASTRN(37:40),' ',STRING(49:50)
    L=L+1
    GO TO 10
ELSE
WRITE(6,*)'THERE ARE ',L,' ELEMENTS.'
WRITE(4,*)L
L2=L
ENDIF
ELSEIF (STRING(4:6).EQ.'752') THEN
READ(1,'(A)')STRING
IF (STRING(19:20).EQ.'0') THEN
READ(STRING(9:10),'(I2)')NG
READ(STRING(57:60),'(I4)')NUMB
    IF(NG.EQ.IT.AND.NG.LE.NPC) THEN
        NC(IT)=NUMB
WRITE(6,*)'THERE ARE ',NC(IT),' NODES ON ',IT,' TH PEC SURFACE'
        IT=IT+1
        L=0
        ELSEIF(NG.EQ.(NPC+1)) THEN
            ND=NUMB
WRITE(6,*)'THERE ARE ',ND,' NODES ON DIE SURFACE'
            L=0
            ELSEIF(NG.EQ.(NPC+2)) THEN
                NB=NUMB
PRINT*, 'THERE ARE ',NB,' NODES AT DIE SURFACE OUTERS'
                L=0
            ENDIF
        ELSEIF (STRING(5:6).EQ.'-1') THEN
            GO TO 20
        ELSEIF (STRING(1:1).EQ.' ') THEN
            DO I=1,4
                IF (STRING((20*I-3):20*I).NE.' ') THEN
                    L=L+1
                    READ(STRING((20*I-3):20*I),'(I4)')TR(L,NG)
                ENDIF
            ENDDO
        ENDIF
        GO TO 30
    ENDIF
    GO TO 15
CLOSE(1)

```

```

-----
DO LL=1,NB
WRITE(10,*)TR(LL,NPC+2)
ENDDO

```

```

....DATA FROM 'NODDY' AND 'ELNO' STORED IN
N1 - TABLE OF NODES MAKING UP THE CORRESPONDING ELEMENT
X,Y,Z - NODAL COORDINATE TABLE

```

```

REWIND 2
REWIND 3
DO I=1,L1
  READ(2,*)NA,X(I),Y(I),Z(I)
ENDDO
DO I=1,L2
  READ(3,*)NK,N1(I,1),N1(I,2),N1(I,3),N1(I,4),MAT(I)
ENDDO
C....CLOSE 'NODDY' AND 'ELNO' FOR GOOD
WRITE(2,*)ZE
WRITE(3,*)ZE
CLOSE(2)
CLOSE(3)
NCOUNT=0
WRITE(6,*)'BE PATIENT #*!/?/#@*!!!'
DO NE=1,L2
C....STORE EDGE INFO IN AN INTEGER ARRAY 'TAB' AFTER CHECKING FOR
C OVERLAP. THE SUBROUTINE 'COMPARE' IS THE HEART OF THE PROGRAM.
  CALL COMPARE(TAB)
C....COMMENTING OUT THE FOLLOWING STATEMENT CAUSES A SPEEDUP OF 250%
C WRITE(6,*)'PROCESSED ELEMENT NO. ',NE,' :EDGE COUNT= ',NCOUNT
ENDDO
WRITE(6,*)'EDGE COUNT = ',NCOUNT
REWIND(4)
C WRITE(6,*)'NO. OF DIELECTRIC LAYERS?'
C READ(5,*)ITEMP
C READ(4,*)(XTMP,I=1,ITEMP)
READ(4,*)XTMP
READ(4,*)(NJ,I=1,2)
DO I=1,(6*L2)
  READ(4,*)NK,EDGV(I,1),EDGV(I,2),EDGV(I,3),NJ
ENDDO
WRITE(4,*)ZE
WRITE(7,*)ZE
CLOSE(4)
CLOSE(7)
NSURF=0
C----
DO 79 IT=1,NPC
LSURF=0
C WRITE(8,*)'ON THE',IT,'TH PEC SURFACE FOR ON-SURFACE EDGES'
  WRITE(6,*)'PROCESSING THE',IT,'TH PEC SURFACE.'
  DO J=1,NC(IT)
    DO K=1,NC(IT)
      DO L=1,NCOUNT
        IF (TAB(L,1).NE.0) THEN
          IF ((TR(J,IT).EQ.TAB(L,1)).AND.(TR(K,IT).EQ.TAB(L,2))) THEN
            WRITE(8,*)L
            TAB(L,1)=0
            NSURF=NSURF+1
            PEDGE(NSURF)=L
            LSURF=LSURF+1
            GO TO 40
          ENDIF
        ENDIF
      ENDDO
    ENDDO
  ENDDO
40 PRINT*, 'THERE ARE',LSURF,'ON-SURFACE EDGES ON',IT,'TH PEC SURFACE'
79 CONTINUE
WRITE(8,*)'0'
WRITE(6,*)'TOTAL NO. OF PEC ON-SURFACE EDGES= ',NSURF
WRITE(6,*)'FEM MATRIX TO BE SOLVED IS OF ORDER ',(NCOUNT-NSURF)
C-----
WRITE(6,*)
WRITE(6,*)'PROCESSING DIE SURFACE FOR ON-SURFACE EDGES.'

```

```

NCNT=0
DO IK=1,L2
  NUN(IK)=0
ENDDO
DO I=1,ND
  DO J=1,L2
    DO K=1,4
      IF (TR(I,(NPC+1)).EQ.N1(J,K)) THEN
        NUN(J)=NUN(J)+1
        BC(J,NUN(J))=N1(J,K)
        IF (NUN(J).EQ.3) THEN
          CALL EDGE(J,BC(J,1),BC(J,2),BC(J,3),NFLAG)
          IF(NFLAG.EQ.0)GOTO 80
          NCNT=NCNT+1
          GO TO 80
        ENDIF
      ENDIF
    ENDDO
  ENDDO
)
ENDDO
WRITE(6,*)'THERE ARE ',NCNT,' DIE ON-SURFACE TRIANGLES.'
CLOSE(8)
WRITE(9,*)ZE
WRITE(10,*)ZE
CLOSE(9)
CLOSE(10)
STOP
END

```

=====

'COMPARE' CHECKS FOR ALL POSSIBLE EDGES AVOIDING OVERLAP AND STORES THE INFO IN AN ARRAY 'TAB'. THE FILE 'EDGY' CONTAINS THE ELEMENT NO. AND THE SIX EDGE VECTORS CORRESPONDING TO THAT ELEMENT.

=====

```

SUBROUTINE COMPARE(TAB)
INTEGER N1(3000,4),MAT(3000)
REAL X(800),Y(800),Z(800)
INTEGER NT(6,2),TAB(4300,2)
COMMON /BANK/N1,X,Y,Z,NE,MAT
COMMON /LOCAL/NCOUNT
NT(1,1)=N1(NE,1)
NT(1,2)=N1(NE,2)
NT(2,1)=N1(NE,1)
NT(2,2)=N1(NE,3)
NT(3,1)=N1(NE,1)
NT(3,2)=N1(NE,4)
NT(4,1)=N1(NE,2)
NT(4,2)=N1(NE,3)
NT(5,1)=N1(NE,2)
NT(5,2)=N1(NE,4)
NT(6,1)=N1(NE,3)
NT(6,2)=N1(NE,4)
....ARRANGING NODE COUPLETS IN ASCENDING ORDER; A PERSONAL CHOICE
DO I=1,6
  IF (NT(I,1).GT.NT(I,2))THEN
    NTMP=NT(I,1)
    NT(I,1)=NT(I,2)
    NT(I,2)=NTMP
  ENDIF
ENDDO
....A BRUTE FORCE SEARCH FOR OVERLAPPING EDGES
L=NCOUNT
DO II=1,6
  IF (NE.EQ.1) GO TO 32
  DO I=1,NCOUNT
    IF ((TAB(I,1).EQ.NT(II,1)).AND.(TAB(I,2).EQ.NT(II,2))) THEN
      WRITE(4,*)NE,I,NT(II,1),NT(II,2),MAT(NE)
    
```

```

          GO TO 35
        ENDIF
      ENDDO
32      L=L+1
        TAB(L,1)=NT(II,1)
        TAB(L,2)=NT(II,2)
        WRITE(4,*)NE,L,NT(II
1      WRITE(7,*)L,X(NT(II,2))-X(NT(II,1)),Y(NT(II,2))-Y(NT(II,1)),
          Z(NT(II,2))-Z(NT(II,1))
35      ENDDO
        NCOUNT=L
        RETURN
      END

```

```

C=====
C  DETERMINES EDGES CORRESPONDING TO THE SURFACE ELEMENT
C=====

```

```

      SUBROUTINE EDGE(M,J1,J2,J3,NFLAG)
      INTEGER EDGV(18000,3),TMP(3),PEDGE(1000),NC(10)
      COMMON /DBASE/EDGV
      COMMON /PECEDGE/PEDGE,NSURF
      DO IJ=(6*M)-5,(6*M)
        IF (J1.EQ.EDGV(IJ,2)) THEN
          IF (J2.EQ.EDGV(IJ,3)) THEN
            TMP(1)=EDGV(IJ,1)
            L1=IJ
          ELSEIF (J3.EQ.EDGV(IJ,3)) THEN
            TMP(2)=EDGV(IJ,1)
            L2=IJ
          ENDIF
        ELSEIF (J2.EQ.EDGV(IJ,2)) THEN
          IF (J3.EQ.EDGV(IJ,3)) THEN
            TMP(3)=EDGV(IJ,1)
            L3=IJ
          ENDIF
        ENDIF
      ENDDO
      CHECK OVERLAPPING WITH PEC EDGES
      ML=0
      DO LL=1,3
        DO K=1,NSURF
          IF (PEDGE(K).EQ.TMP(LL)) THEN
            ML=ML+1
          END IF
        END DO
      END DO
      IF (ML.GT.3) THEN
        PRINT*,'PEDGE(K)',PEDGE(K)
      ENDIF
      IF (ML.EQ.3) THEN
        NFLAG=0
      ELSE
        NFLAG=1
        WRITE(9,*)M,TMP(1),EDGV(L1,2),EDGV(L1,3),TMP(2),EDGV(L2,2),
1      EDGV(L2,3),TMP(3),EDGV(L3,2),EDGV(L3,3)
      C      WRITE(9,*)M,J1,J2,J3,TMP
      ENDIF
101     CONTINUE
      RETURN
      END

```

```

C *****
C PROGRAM TO PRECESS DATA FROM PREP1.FTN. IT OUTPUT THE DATA FILES
C 1. PLTAE0.DAT (FOR BI); 2. TAB1 (FOR ON-SURFACE NEW/OLD NUMBERING
C SYSTEM CONTRAST TABLES.) TAB1:FOR TRIANGLES; TAB2 FOR NODES;
C TAB3: FOR EDGES. (BE CARE OF THE MAXIMUM DIMENSIONS DEFINED
C IN THE "DIMENSION" PARAMETER STATEMENT, BEFORE RUNNING!

```

```

*****
PARAMETER (MAXA=1500,MAX=1000)
INTEGER ITAB1 (MAX), ITAB2 (MAX), ITAB3 (MAX), ITAB5 (MAX,2)
INTEGER IA (3), IB (6), IC (MAX), ID (3,2), IP, ITAB6 (MAX,3)
INTEGER ITAB6A (MAX,3), ITAB3A (MAX), ITAB3B (MAX)
INTEGER ITAB5A (MAX,2), ITAB5B (MAX,2)
REAL XYZ (MAXA,3), TAB4 (MAX,3)
100 OPEN (2,FILE='enoddy')
READ (2,*) I1,XYZ (I1,1),XYZ (I1,2),XYZ (I1,3)
IF (I1.EQ.0)GO TO 200
GO TO 100
200 CLOSE (2)
OPEN (3,FILE='esurfc')
J1=1
300 READ (3,*) IC (J1)
IF (IC (J1).EQ.0)GO TO 400
J1=J1+1
GO TO 300
400 CLOSE (3)
LL=J1-1
OPEN (4,FILE='esurfd')
I=0
J=1
L1=1
L2=1
500 READ (4,*) IP, IA (1), IB (1), IB (2), IA (2), IB (3), IB (4), IA (3), IB (5), IB (6)
IF (IP.EQ.0)GO TO 600
K1=0
DO 10 I1=1,3
DO 5 J1=1,2
ID (I1,J1)=IB (J1+K1)
5 CONTINUE
K1=K1+2
10 CONTINUE
I=I+1
ITAB1 (I)=IP
DO 20 I1=1,3
ITAB6A (I,I1)=IA (I1)
20 CONTINUE
DO 30 I1=1,6
DO 25 J1=1,J
IF (IB (I1).EQ.ITAB2 (J1))GO TO 30
25 CONTINUE
ITAB2 (J)=IB (I1)
J=J+1
30 CONTINUE
DO 40 I1=1,3
DO 32 J1=1,LL
IF (IA (I1).EQ.IC (J1)) THEN
DO 31 K1=1,L2
IF (IA (I1).EQ.ITAB3B (K1))GO TO 40
31 CONTINUE
ITAB3B (L2)=IA (I1)
ITAB5B (L2,1)=ID (I1,1)
ITAB5B (L2,2)=ID (I1,2)
L2=L2+1
GO TO 40
ELSE
END IF
32 CONTINUE
DO 35 K1=1,L1
IF (IA (I1).EQ.ITAB3A (K1)) GO TO 40
35 CONTINUE
ITAB3A (L1)=IA (I1)
ITAB5A (L1,1)=ID (I1,1)
ITAB5A (L1,2)=ID (I1,2)

```

```

      L1=L1+1
40  CONTINUE
      GO TO 500
600  CLOSE(4)
      II=I
      JJ=J-1
      INEDS=L1-1
      IEXEDS=L2-1
      KK=(L1-1)+(L2-1)
      DO 50 I1=1, JJ
      DO 50 J1=1, 3
      M=ITAB2(I1)
      TAB4(I1, J1)=XYZ(M, J1)
50  CONTINUE
      DO 55 I1=1, L1-1
      ITAB3(I1)=ITAB3A(I1)
      DO 55 J1=1, 2
      DO 51 K1=1, JJ
      M1=ITAB5A(I1, J1)
      IF(M1.EQ.ITAB2(K1)) THEN
      ITAB5(I1, J1)=K1
      ELSE
      END IF
51  CONTINUE
55  CONTINUE
      M=L1-1
      DO 60 I1=1, L2-1
      ITAB3(M+I1)=ITAB3B(I1)
      DO 60 J1=1, 2
      DO 59 K1=1, JJ
      M1=ITAB5B(I1, J1)
      IF(M1.EQ.ITAB2(K1)) THEN
      ITAB5(M+I1, J1)=K1
      ELSE
      END IF
59  CONTINUE
60  CONTINUE
      DO 70 I1=1, II
      DO 65 J1=1, 3
      M=ITAB6A(I1, J1)
      DO 63 K1=1, KK
      IF(M.EQ.ITAB3(K1)) THEN
      ITAB6(I1, J1)=K1
      GO TO 65
      ELSE
      END IF
63  CONTINUE
65  CONTINUE
70  CONTINUE
      OPEN(14, FILE='PLATE0.DAT')
      WRITE(14, *) 'FLAG1'
      WRITE(14, *) 'NODCRDS'
      DO I1=1, JJ
      WRITE(14, *) I1, (TAB4(I1, J1), J1=1, 3)
      END DO
      WRITE(14, *) '-1, 0, 0, 0'
      WRITE(14, *) 'FLAG2'
      WRITE(14, *) 'NUMBER OF INTERIOR EDGES'
      WRITE(14, *) INEDS
      WRITE(14, *) 'NUMBER OF EXTERIOR EDGES'
      WRITE(14, *) IEXEDS
      WRITE(14, *) 'EDGNODS'
      DO I1=1, KK
      WRITE(14, *) I1, (ITAB5(I1, J1), J1=1, 2)
      END DO
      WRITE(14, *) 'FLAG3'

```



```
WRITE(14,*)'NUMBER OF TRIANGLES'  
WRITE(14,*)II  
WRITE(14,*)'TRIEDGS'  
DO I1=1,II  
WRITE(14,*)I1,(ITAB6(I1,J1),J1=1,3)  
END DO  
CLOSE(14)  
OPEN(11,FILE='TAB1')  
OPEN(21,FILE='TAB2')  
OPEN(31,FILE='TAB3')  
DO I1=1,II  
WRITE(11,*)I1,ITAB1(I1)  
END DO  
DO I1=1,JJ  
WRITE(21,*)I1,ITAB2(I1)  
END DO  
DO I1=1,KK  
WRITE(31,*)I1,ITAB3(I1)  
END DO  
CLOSE(11)  
CLOSE(21)  
CLOSE(31)  
END
```

From jxgum Thu Dec 26 14:34:33 1991
Received: by jaguar.engin.umich.edu (5.64/1.35)
id 5600a3122.000bdb7; Thu, 26 Dec 91 14:30:09 -0500
Date: Thu, 26 Dec 91 14:30:09 -0500
From: Jian Gong <jxgum>
Message-Id: <5600a3122.000bdb7@jaguar.engin.umich.edu>
To: volakis
Status: R

PROGRAM FEM_MOM

```
C*****
C*      THE PROGRAM INCLUDES TWO SECTIONS: ONE IS FEM TO SOLVE THE      *
C* THE INTERIOR REGION; THE OTHER IS MOM (OR BI) TO DEAL WITH THE      *
C* THE EXTERIOR REGION. ALL DATA FILES NEEDED BY THE TWO METHODS HAVE *
C* BEEN CREATED BY "PREP1.FTN" AND "PREP2.FTN". USER ONLY NEED TO KEY *
C* IN SOME INFO REQUESTED BY THIS CODE, WHICH INDICATES WHERE THE INFO *
C* MAY BE OBTAINED.                                                    *
C*      THE OUTPUT MAY BE THE SURFACE FIELD (MAGNITUDE/PHASE) OR      *
C* SCATTERED/RADIATED FIELDS (IN TERMS OF RCS)                          *
C*      IN THE FIRST STAGE, THE PROGRAM FEM-MOM SOLVES THE PROBLEM OF *
C* SCATTERING FROM A CAVITY ON AN INFINITE GROUND PLANE (PEC). THE     *
C* ARBITRARILY SHAPED TOP SURFACE MODELED WITH TRIANGULAR PATCHES.THE *
C* INTERIOR IS DIRCRETIZED INTO TETRAHEDRA ELEMENTS.                  *
C*****
```

```
INCLUDE 'CONST.INC'
INCLUDE 'DIM.INC'
```

```
REAL NODCRDS (MAXNOD, 3), EDGLEN (MAXEDG), R (MAXTRI), PI, VRCS, DEG,
* RCOND, FANGO, ANGL0, STEPANG, ALPHA, THETA1, PHII, THETA0, PHIO,
* AE (MAXA, MAXA)
```

```
INTEGER NINTEDG, EDGNODS (MAXEDG, 2), NTRIS, TRIEDGS (MAXTRI, 3), CT
INTEGER TRISIGN (MAXTRI, 3), IPVT (MAXA), K, PTYPE, FIX, NUMPTS, STYPE
INTEGER EDST (MAXA, 2), EDGSGN (MAXZ), TAB3 (MAXEDG)
COMPLEX A (MAXA, MAXA), VC (MAXA), EN (MAXA), DUM (MAXA)
COMPLEX EZ (MAXZ, MAXZ), V (MAXZ), ES (MAXZ), PJ
CHARACTER*40 MESH_FILE, RES_FILE, OUT_FILE
```

```
2 FORMAT(' ', F15.10, ' ', ' ', F17.10)
```

```
PI=3.14159265359
PJ=(0., 1.)
```

```
PRINT*, 'INPUT TOTAL # OF EDGES'
READ*, NED
PRINT*, 'INPUT TOTAL # OF PEC EDGES'
READ*, NES
PRINT*, 'INPUT # OF ON-DIE-SURFACE NODES'
READ*, NNOD
```

```
CALL USER (MESH_FILE, STYPE, RES_FILE, OUT_FILE, PTYPE, ALPHA,
* THETA1, PHII, FIX, FANGO, ANGL0, NUMPTS, STEPANG)
```

```
CALL MESHDATA (MESH_FILE, NODCRDS, NINTEDG, EDGNODS, NTRIS, TRIEDGS)
```

```
IF (STYPE.EQ.RESIST) THEN
CALL GET_R (RES_FILE, NTRIS, R)
ENDIF
```

```
CALL ACREATOR (NED, NES, EDST, A)
NET=NED-NES
```

```
CALL ZCREATOR (NODCRDS, NINTEDG, EDGNODS, NTRIS, TRIEDGS, STYPE, R,
* EDGLEN, TRISIGN, EZ)
```

```
PRINT*, 'FINISH Z CREATING'
```

```

CALL COMB1(MESH_FILE, TRISIGN, NNOD, TAB3, EDGSGN)

PRINT*, 'FINISH COMB1'

CALL COMB2(NINTEDG, EDST, TAB3, EDGSGN, EZ, A)
DO I=1, NET
DO J=1, NET
IF (CABS(A(I, J)).NE.0.) THEN
A(I, J)=CZERO*A(I, J)
ENDIF
ENDDO
ENDDO

PRINT*, 'FINISH COMB2'

CALL CGECO(A, MAXA, NET, IPVT, RCOND, DUM)

OPEN(9, FILE='RCS.DAT')
OPEN(8, FILE='RCS1.DAT')

IF (PTYPE.EQ.BISTAT) THEN

CALL EXCIT(NODCRDS, NINTEDG, EDGNODS, NTRIS, TRIEDGS, EDGLEN,
*          TRISIGN, ALPHA, THETA, PHI, V)

DO II=1, NINTEDG
LL=EDST(TAB3(II), 1)
VC(LL)=CZERO*V(II)
ENDDO

CALL BACK_SUBST(A, VC, NET, IPVT, EN)

WRITE(212, *) 'ES'
DO II=1, NINTEDG
LL=EDST(TAB3(II), 1)
ES(II)=EN(LL)
WRITE(212, *) ES(II)
ENDDO

WRITE(9, *) 'E FIELD ON SURFACE (ES)'
WRITE(9, *) 'MAGNITUDE'
DO 5 CT=1, NINTEDG
ESR=REAL(ES(CT))
ESI=(ES(CT)-CONJG(ES(CT)))/2./PJ
ESMAG=SQRT(ESR**2+ESI**2)
WRITE(9, *) CT, ESMAG
5 CONTINUE
WRITE(9, *) 'PHASE'
DO 8 CT=1, NINTEDG
ESR=REAL(ES(CT))
ESI=(ES(CT)-CONJG(ES(CT)))/2./PJ
ESPH=ATAN(ESI/ESR)
ESPH=180.*ESPH/PI
WRITE(9, *) CT, ESPH
8 CONTINUE
WRITE(9, *) ' '
END IF

IF (FIX.EQ.ANGPHI) THEN
PHIO=FANGO
THETAO=ANG10
ELSE
THETAO=FANGO
PHIO=ANG10

```

END IF

DO 10 K=1,NUMPTS

IF (PTYPE.EQ.BACKSCAT) THEN

THETA=THETAO

PHI=PHIO

CALL EXCIT(NODCRDS,NINTEDG,EDGNODS,NTRIS,TRIEDGS,EDGLEN,
* TRISIGN,ALPHA,THETA,PHI,V)

DO II=1,NINTEDG

LL=EDST(TAB3(II),1)

VC(LL)=CZERO*V(II)

END DO

CALL BACK_SUBST(A,VC,NET,IPVT,EN)

END IF

WRITE(212,*)'ES'

DO II=1,NINTEDG

LL=EDST(TAB3(II),1)

ES(II)=EN(LL)

WRITE(212,*)ES(II)

ENDDO

CALL RCS(NODCRDS,NINTEDG,EDGNODS,NTRIS,TRIEDGS,EDGLEN,
* TRISIGN,ES,THETAO,PHIO,RCSTH,RCS PHI)

IF (FIX.EQ.ANGPHI) THEN

DEG=THETAO*180./PI

THETAO=THETAO+STEPANG

ELSE

DEG=PHIO*180./PI

PHIO=PHIO+STEPANG

END IF

IF (K.EQ.1) THEN

WRITE(9,*)'INPUT DATA'

WRITE(8,*)'INPUT DATA'

END IF

WRITE(9,2)DEG,RCSTH

WRITE(8,2)DEG,RCS PHI

10 CONTINUE

CLOSE(212)

CLOSE(9)

CLOSE(8)

STOP

END

SUBROUTINE MESHDATA(MESH_FILE,NODCRDS,NINTEDG,EDGNODS,NTRIS,
* TRIEDGS)

C*****
C* SR GET_MESH READS IN MESH DATA FROM FILE SPECIFIED BY *
C* USER. *
C*****

INCLUDE 'DIM.INC'

REAL NODCRDS(MAXNOD,3)

```

    INTEGER I,NODE,NINTEDG,NEXTEDG,NEDG,J,EDGNODS (MAXEDG,2),NTRIS,
*      TRIEDGS (MAXTRI,3),K,DUM
    CHARACTER*120 LINE
    CHARACTER*40 MESH_FILE

    OPEN (UNIT=7,FILE=MESH_FILE)
100 CONTINUE
    READ(7,'(A)',END=1000) LINE
    IF (LINE(2:6).EQ.'FLAG1') THEN
        READ(7,'(A)') LINE
        I=1
200 READ(7,*) NODE,NODCRDS(I,1),NODCRDS(I,2),NODCRDS(I,3)
        IF (NODE.EQ.-1) THEN
            GOTO 300
        END IF
        I=I+1
        GOTO 200
300 CONTINUE

        ELSE IF (LINE(2:6).EQ.'FLAG2') THEN
            READ(7,'(A)') LINE
            READ(7,*) NINTEDG
            READ(7,'(A)') LINE
            READ(7,*) NEXTEDG
            READ(7,'(A)') LINE
            NEDG=NINTEDG+NEXTEDG
            DO 400 J=1,NEDG
                READ(7,*) DUM,EDGNODS(J,1),EDGNODS(J,2)
400 CONTINUE

            ELSE IF (LINE(2:6).EQ.'FLAG3') THEN
                READ(7,'(A)') LINE
                READ(7,*) NTRIS
                READ(7,'(A)') LINE
                DO 500 K=1,NTRIS
                    READ(7,*) DUM,TRIEDGS(K,1),TRIEDGS(K,2),TRIEDGS(K,3)
500 CONTINUE

                END IF
                GOTO 100
000 CONTINUE
        CLOSE(7)
        RETURN
        END

    SUBROUTINE ZCREATOR(NODCRDS,NINTEDG,EDGNODS,NTRIS,TRIEDGS,
*      STYPE,R,EDGLEN,TRISIGN,Z)
*****
    SR FILLZ CALCULATES ELEMENTS OF IMPEDANCE MATRIX Z.
*****

    INCLUDE 'CONST.INC'
    INCLUDE 'DIM.INC'
    REAL NODCRDS(MAXNOD,3),RC,R(MAXTRI),CTRD,LONGEST,EDGLEN(MAXEDG),
*      OSIGN,SSIGN,PI,ETA,SAREA,SRCAREA
    INTEGER NEDG,NINTEDG,EDGNODS(MAXEDG,2),NTRIS,TRIEDGS(MAXTRI,3),
*      TRISIGN(MAXTRI,3),EDGUSE(MAXEDG),P,Q,NUMTRIS,M(3),N(3),
*      SINGFLAG,SINGPT,INM,OUTM,V,W,K,SRC,OBS,I1,I2,I3,I4,I5,
*      STYPE,RESPT
    COMPLEX J,FSG,PFSG,A,PHI,Z(MAXZ,MAXZ),UNIFNC,LINFNC,PHISING,ASING,
*      RESFNC,VINT,ZRES
    EXTERNAL FSG,PFSG,UNIFNC,LINFNC,RESFNC

    REAL OBSV(NMAX,NMAX),SRCV(NMAX,NMAX)

```

```

COMMON /TRICONST/OBSV, SRCV
REAL RM(3), RN(3)
COMMON /BASES/RM, RN
REAL NEAR
COMMON /ADJ/NEAR

DO 3 I1=1, MAXEDG
EDGUSE(I1)=0
3 CONTINUE

DO 5 I2=1, MAXTRI
DO 7 I3=1, 3
TRISIGN(I2, I3)=0
7 CONTINUE
5 CONTINUE
DO 8 I4=1, NINTEDG
DO 9 I5=1, NINTEDG
Z(I4, I5)=(0.0, 0.0)
9 CONTINUE
8 CONTINUE
J=(0.0, 1.0)
PI=3.1415926536
ETA=376.7343091821
SRC=1
OBS=0

DO 10 P=1, NTRIS

DO 25 I1=1, 3
M(I1)=TRIEDGS(P, I1)
25 CONTINUE

CALL GET_VERTS(M, OBS, NODCRDS, EDGNODS)

DO 20 Q=1, NTRIS

DO 30 I2=1, 3
N(I2)=TRIEDGS(Q, I2)
30 CONTINUE

CALL GET_VERTS(N, SRC, NODCRDS, EDGNODS)

RC=CTRD()

CALL EDGE_LEN(M, N, NODCRDS, EDGNODS, EDGUSE, EDGLEN, LONGEST)

IF (P.EQ.Q) THEN
SINGFLAG=1
SINGPT=1
NEAR=1.0
INM=7
OUTM=1
RESPT=7
ELSE IF (RC.LE.LONGEST) THEN
SINGFLAG=1
SINGPT=1
NEAR=1.0
INM=7
OUTM=1

ELSE
SINGFLAG=0
NEAR=0.0
INM=7

```

```

OUTM=1
END IF

CALL AC_INT (FSG,OUTM, INM, PHI)

IF (SINGFLAG.EQ.1) THEN
CALL SINT (SINGPT, UNIFNC, PHISING)
SAREA=SRCAREA ()
PHISING=PHISING/SAREA
PHI=PHI+PHISING
END IF

DO 40 V=1,3

IF (M(V).LE.NINTEDG) THEN
DO 50 W=1,3

IF (N(W).LE.NINTEDG) THEN

DO 60 K=1,3
RM(K)=OBSV(V,K)
RN(K)=SRCV(W,K)
60 CONTINUE

CALL AC_INT (PFSG,OUTM, INM, A)

IF (SINGFLAG.EQ.1) THEN
CALL SINT (SINGPT, LINFNC, ASING)
ASING=ASING/SAREA
A=A+ASING
END IF

IF ((P.EQ.Q).AND.(STYPE.EQ.RESIST)) THEN
CALL SINT (RESPT, RESFNC, VINT)
ZRES=ETA/4.*EDGLEN(M(V))*EDGLEN(N(W))*OSIGN*SSIGN*
* R(Q)*VINT/SAREA
ELSE
ZRES=(0.0,0.0)
END IF

Z(M(V),N(W))=Z(M(V),N(W))-0.5*PI*EDGLEN(M(V))
* EDGLEN(N(W))*OSIGN*SSIGN*(A-PHI/(PI*PI))

ELSE
EDGUSE(N(W))=1
END IF

50 CONTINUE

ELSE
EDGUSE(M(V))=1
END IF

40 CONTINUE

20 CONTINUE

10 CONTINUE

RETURN
END

```

```

SUBROUTINE GET_VERTS (EDG, FLAG, NODCRDS, EDGNODS)
*****

```

```

C*  SR GET_VERTS RETURNS THE VERTICES OF A TRIANGLE THRU      *
C*  COMMON BLOCK "TRICONST" GIVEN THE NUMBERS OF THE      *
C*  TRIANGLE EDGES.  THE CALLING ROUTINE SPECIFIES IF THE  *
C*  TRIANGLE IS SOURCE OR OBSERVATION.  THE TRIANGLE      *
C*  EDGE NUMBERS ARE RECEIVED ARRAY EDG.  THE VERTICES    *
C*  (POSITION VECTORS) ARE THEN ASSIGNED SUCH THAT THE    *
C*  VERTEX IN OBSV/SRCV ARRAY POSITION (I,-) IS OPPOSITE  *
C*  THE EDGE IN EDG ARRAY POSITION (I).                    *
C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL NODCRDS (MAXNOD, 3)
INTEGER EDG (3), NODE (3), ORDNODE (3), EDGNODS (MAXEDG, 2),
*      I, J1, J2, K1, K2, SRC, OBS, FLAG

```

```

REAL OBSV (NMAX, NMAX), SRCV (NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV

```

```

SRC=1
OBS=0

```

```

NODE (1) = EDGNODS (EDG (1), 1)
NODE (2) = EDGNODS (EDG (1), 2)
IF ( (EDGNODS (EDG (1), 1) .NE. EDGNODS (EDG (2), 1)) .AND.
*   (EDGNODS (EDG (1), 2) .NE. EDGNODS (EDG (2), 1)) ) THEN
NODE (3) = EDGNODS (EDG (2), 1)
ORDNODE (1) = EDGNODS (EDG (2), 1)
ELSE
NODE (3) = EDGNODS (EDG (2), 2)
ORDNODE (1) = EDGNODS (EDG (2), 2)
END IF

```

```

DO 10 I=2, 3
IF ( (EDGNODS (EDG (I), 1) .NE. NODE (1)) .AND.
*   (EDGNODS (EDG (I), 2) .NE. NODE (1)) ) THEN
ORDNODE (I) = NODE (1)
ELSE IF ( (EDGNODS (EDG (I), 1) .NE. NODE (2)) .AND.
*   (EDGNODS (EDG (I), 2) .NE. NODE (2)) ) THEN
ORDNODE (I) = NODE (2)
ELSE
ORDNODE (I) = NODE (3)
END IF
10 CONTINUE

```

```

IF (FLAG.EQ.OBS) THEN
DO 20 J1=1, 3
DO 30 K1=1, 3
OBSV (J1, K1) = NODCRDS (ORDNODE (J1), K1)
30 CONTINUE
20 CONTINUE
END IF

```

```

IF (FLAG.EQ.SRC) THEN
DO 50 J2=1, 3
DO 40 K2=1, 3
SRCV (J2, K2) = NODCRDS (ORDNODE (J2), K2)
40 CONTINUE
50 CONTINUE
END IF

```

```

RETURN
END

```



```

SUBROUTINE EDGE_LEN(M,N,NODCRDS,EDGNODS,EDGUSE,EDGLEN, LONGEST)
*****
SR EDGE_LEN UPDATES THE EDGE LENGTH TABLE GIVEN THE          *
EDGE NUMBERS OF THE SOURCE AND OBSERVATION TRIANGLES.      *
IT ALSO RETURNS THE LENGTH OF THE LONGEST EDGE OF THE      *
SOURCE AND OBSERVATION TRIANGLES.                          *
*****

INCLUDE 'DIM.INC'

REAL NODCRDS(MAXNOD,3),LONGEST,VECT1(3),VECT2(3),DIFF(3),VDOT,
*   EDGLEN(MAXEDG)
INTEGER EDGNODS(MAXEDG,2),EDGUSE(MAXEDG),M(3),N(3),I,J,K,P,
*   NODE1,NODE2,S

LONGEST=0.0

DO 10 I=1,3

IF (EDGUSE(M(I)).EQ.0) THEN
NODE1=EDGNODS(M(I),1)
NODE2=EDGNODS(M(I),2)
DO 20 J=1,3
VECT1(J)=NODCRDS(NODE1,J)
VECT2(J)=NODCRDS(NODE2,J)
20 CONTINUE
CALL VECT_DIFF(1.0,VECT1,1.0,VECT2,DIFF)
EDGLEN(M(I))=SQRT(VDOT(DIFF,DIFF))
END IF
IF (EDGLEN(M(I)).GT.LONGEST) THEN
LONGEST=EDGLEN(M(I))
END IF
10 CONTINUE

DO 30 K=1,3

IF (EDGUSE(N(K)).EQ.0) THEN

IF ((N(K).NE.M(1)).AND.(N(K).NE.M(2)).AND.(N(K).NE.M(3))) THEN
NODE1=EDGNODS(N(K),1)
NODE2=EDGNODS(N(K),2)
DO 40 P=1,3
VECT1(P)=NODCRDS(NODE1,P)
VECT2(P)=NODCRDS(NODE2,P)
40 CONTINUE
CALL VECT_DIFF(1.0,VECT1,1.0,VECT2,DIFF)
EDGLEN(N(K))=SQRT(VDOT(DIFF,DIFF))
END IF
END IF
IF (EDGLEN(N(K)).GT.LONGEST) THEN
LONGEST=EDGLEN(N(K))
END IF
30 CONTINUE
RETURN
END

SUBROUTINE UPDATE_SIGNS(M,N,V,W,P,Q,TRISIGN,EDGUSE,OSIGN,SSIGN)
*****
SR UPDATE_SIGNS UPDATES THE TRIANGLE SIGN AND EDGE          *
USE TABLES FOR AN INPUT OBSERVATION INTERIOR EDGE AND    *
AN INPUT SOURCE INTERIOR EDGE.                             *
*
*
THE TRIANGLE SIGN TABLE IS AN ARRAY OF TRIANGLE          *
NUMBERS VS. TRIANGLE EDGE CURRENT DIRECTIONS. THE        *

```

```

C*          SIGN IN POSITION (K,L) CORRESPONDS TO EDGE N IN      *
C*          TRIANGLE EDGE TABLE (TRIEDGS) POSITION (K,L). THE  *
C*          SIGN DESIGNATES THE CURRENT DIRECTION ACROSS THE NTH *
C*          EDGE IN THE KTH TRIANGLE.                             *
C*          CASES:  -1-CURRENT DIRECTED ACROSS INTERIOR EDGE N   *
C*                   INTO TRIANGLE K                             *
C*                   +1-CURRENT DIRECTED ACROSS INTERIOR EDGE N  *
C*                   OUT OF TRIANGLE K                           *
C*                   0-EXTERIOR EDGE, NO CURRENT                 *
C*          SAY TRIANGLES A AND B SHARE EDGE N. ONE TRIANGLE MUST *
C*          BE DESIGNATED AS + (SAY A) AND THE OTHER AS - (B).  *
C*          CURRENT IS DIRECTED ACROSS EDGE N FROM + -> -      *
C*          (A -> B).                                           *
C*                                                                *
C*          CURRENT DIRECTIONS ARE ASSIGNED ON THE BASIS OF THE  *
C*          EDGE USE TABLE. THE EDGE USE TABLE RECORDS THE    *
C*          NUMBER OF TIMES AN EDGE HAS BEEN USED BY DIFFERENT  *
C*          TRIANGLES.                                           *
C*          CASES:  0-EDGE NEVER USED                             *
C*                   1-EDGE USED BY SOME TRIANGLE I             *
C*                   2-EDGE USED BY TRIANGLE I AND TRIANGLE J,  *
C*                   WHERE I DOES NOT EQUAL J                    *
C*                                                                *
C*          THIS ROUTINE CHECKS THE EDGE USE TABLE FOR INPUT    *
C*          INTERIOR EDGE N, AND UPDATES THE TRIANGLE SIGN TABLE *
C*          ACCORDING TO...                                       *
C*          EDGE N NEVER USED BEFORE                             -> +1 *
C*          EDGE N USED PREVIOUSLY BY DIFFERENT TRIANGLE -> -1  *
C*          THE ROUTINE THEN UPDATES THE EDGE USE TABLE, AND    *
C*          RETURNS THE TRIANGLE EDGE SIGNS.                     *
C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL OSIGN,SSIGN
INTEGER TRISIGN(MAXTRI,3),EDGUSE(MAXEDG),M(3),N(3),V,W,P,Q

```

```

IF (EDGUSE(M(V)).EQ.0) THEN
EDGUSE(M(V))=1
TRISIGN(P,V)=1
OSIGN=1.0

```

```

ELSE IF (TRISIGN(P,V).EQ.0) THEN
EDGUSE(M(V))=2
TRISIGN(P,V)=-1
OSIGN=-1.0

```

```

ELSE
IF (TRISIGN(P,V).EQ.1) THEN
OSIGN=1.0
ELSE
OSIGN=-1.0
END IF
END IF

```

```

IF (EDGUSE(N(W)).EQ.0) THEN
EDGUSE(N(W))=1
TRISIGN(Q,W)=1
SSIGN=1.0
ELSE IF (TRISIGN(Q,W).EQ.0) THEN
EDGUSE(N(W))=2
TRISIGN(Q,W)=-1
SSIGN=-1.0
ELSE
IF (TRISIGN(Q,W).EQ.1) THEN

```

```

SSIGN=1.0
ELSE
SSIGN=-1.0
END IF
END IF

```

```

RETURN
END

```

```

REAL FUNCTION CTRD()

```

```

*****
FNC CTRD COMPUTES THE DIFFERENCE BETWEEN OBSERVATION          *
AND SOURCE TRIANGLE CENTROIDS.                                *
*****

```

```

INCLUDE 'DIM.INC'
REAL VECT1(3), VECT2(3), COEFF, DIFF(3)
INTEGER I

```

```

REAL OBSV(NMAX, NMAX), SRCV(NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV

```

```

DO 10 I=1, 3
VECT1(I)=OBSV(1, I)+OBSV(2, I)+OBSV(3, I)
VECT2(I)=SRCV(1, I)+SRCV(2, I)+SRCV(3, I)
10 CONTINUE
COEFF=1./3.
CALL VECT_DIFF(COEFF, VECT1, COEFF, VECT2, DIFF)
CTRD=SQRT(VDOT(DIFF, DIFF))
RETURN
END

```

```

COMPLEX FUNCTION FSG(LP1, LP2, LP3, L1, L2, L3)

```

```

*****
FNC FSG RETURNS THE VALUE OF THE...                            *
1. FREE SPACE GREEN'S FNC, EXP(-JKR)/R, IF NEAR=0.0          *
2. FREE SPACE GREEN'S FUNCTION WITH SINGULARITY              *
   REMOVED, [EXP(-JKR)-1]/R, IF NEAR=1.0                    *
THE ARGUMENTS OF FSG ARE THE LOCAL AREA COORDINATES         *
OF OBSERVATION/SOURCE POINTS WITHIN GIVEN                   *
OBSERVATION/SOURCE TRIANGLES.                                *
*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL LP1, LP2, LP3, L1, L2, L3, COEFF(NMAX), R, PI, VECTS(NMAX, NMAX),
*   VSUM(3), VDOT
INTEGER I, K, M
COMPLEX J

```

```

REAL OBSV(NMAX, NMAX), SRCV(NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV
REAL NEAR
COMMON /ADJ/NEAR

```

```

J=(0.0, 1.0)
PI=3.1415926536
COEFF(1)=L1
COEFF(2)=L2
COEFF(3)=L3
COEFF(4)=-1.0*LP1
COEFF(5)=-1.0*LP2
COEFF(6)=-1.0*LP3
DO 10 I=1, 3

```

```

DO 20 K=1,3
VECTS (I,K)=OBSV (I,K)
VECTS (I+3,K)=SRCV (I,K)
20 CONTINUE
10 CONTINUE

CALL SUM_VECTS (6,VECTS,COEFF,VSUM)
R=SQRT (VDOT (VSUM,VSUM))
IF (R.LE.1E-6) THEN
FSG=(0.0,-6.283185307)
ELSE
FSG=(CEXP (-1*J*2.0*PI*R)-NEAR)/R
ENDIF
RETURN
END

```

```

COMPLEX FUNCTION PFSG(LP1,LP2,LP3,L1,L2,L3)
C*****
C* FNC PFSG RETURNS THE VALUE OF... *
C* 1. (R-RI).(R'-RJ)*EXP(-JKR)/R, IF NEAR=0.0 *
C* 2. (R-RI).(R'-RJ)*[EXP(-JKR)-1]/R IF NEAR=1.0 *
C* (NOTE REMOVED SINGULARITY) *
C* WHERE RI/RJ IS THE POSITION VECTOR TO THE ITH/JTH *
C* VERTEX OF THE OBSERVATION/SOURCE TRIANGLE. *
C* THE ARGUMENTS OF PFSG ARE THE LOCAL AREA COORDINATES *
C* OF OBSERVATION/SOURCE POINTS WITHIN GIVEN *
C* OBSERVATION/SOURCE TRIANGLES. *
C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL LP1,LP2,LP3,L1,L2,L3,COEFF (NMAX),R,ARG,PI,VECTS (NMAX,NMAX),
* RO (3),RP (3),RMINRP (3),RMINRI (3),RPMINRJ (3),IMAG,VDOT
INTEGER I,K
COMPLEX J

```

```

REAL OBSV (NMAX,NMAX),SRCV (NMAX,NMAX)
COMMON /TRICONST/OBSV,SRCV
REAL NEAR
COMMON /ADJ/NEAR
REAL RI (3),RJ (3)
COMMON /BASES/RI,RJ

```

```

J=(0.0,1.0)
PI=3.1415926536

```

```

COEFF (1)=L1
COEFF (2)=L2
COEFF (3)=L3
DO 10 I=1,3
DO 20 K=1,3
VECTS (I,K)=OBSV (I,K)
20 CONTINUE
10 CONTINUE
CALL SUM_VECTS (3,VECTS,COEFF,RO)

```

```

COEFF (1)=LP1
COEFF (2)=LP2
COEFF (3)=LP3
DO 30 I=1,3
DO 40 K=1,3
VECTS (I,K)=SRCV (I,K)

```

```

40 CONTINUE
30 CONTINUE
  CALL SUM_VECTS (3, VECTS, COEFF, RP)

  CALL VECT_DIFF (1.0, RO, 1.0, RP, RMINRP)
  CALL VECT_DIFF (1.0, RO, 1.0, RI, RMINRI)
  CALL VECT_DIFF (1.0, RP, 1.0, RJ, RMINRJ)
  R=SQRT (VDOT (RMINRP, RMINRP))
  ARG=VDOT (RMINRI, RMINRJ)
  IF (R.LE.1E-6) THEN
    IMAG=-2*PI*ARG
    PFSG=CMPLX (0.0, IMAG)
  ELSE
    PFSG=ARG* (CEXP (-1*J*2.0*PI*R) -NEAR) /R
  ENDIF
  RETURN
END

```

```

      COMPLEX FUNCTION UNIFNC (L1, L2, L3)
*****
      FNC UNIFNC RETURNS THE VALUE OF THE INTEGRATION          *
      OVER THE SOURCE TRIANGLE OF THE REMOVED SINGULARITY      *
      PHI: 1/R. THE ARGUMENTS OF UNIFNC ARE THE LOCAL          *
      AREA COORDINATES OF THE OBSERVATION PT. WITHIN THE      *
      OBSERVATION TRIANGLE.                                     *
*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL L1, L2, L3, COEFF (NMAX), VECTS (NMAX, NMAX), R (3), UNIPOT,
*   LINPOT (3), NHAT (3), P (3), VERTS (4, 3)
INTEGER I, J, UNIFLAG, LINFLAG, CT1, CT2
COMPLEX DUM

```

```

REAL OBSV (NMAX, NMAX), SRCV (NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV

```

```

COEFF (1)=L1
COEFF (2)=L2
COEFF (3)=L3
DO 10 I=1, 3
DO 20 J=1, 3
VECTS (I, J)=OBSV (I, J)

```

```

20 CONTINUE
10 CONTINUE
  CALL SUM_VECTS (3, VECTS, COEFF, R)

```

```

UNIFLAG=1
LINFLAG=0

```

```

DO 30 CT1=1, 3
DO 40 CT2=1, 3
VERTS (CT1, CT2)=SRCV (CT1, CT2)

```

```

40 CONTINUE
30 CONTINUE
  CALL POT_INTS_SS (3, R, VERTS, UNIFLAG, UNIPOT, LINFLAG, LINPOT, NHAT, P)
  DUM=(0.0, 0.0)
  UNIFNC=DUM+UNIPOT
  RETURN
END

```

```

      COMPLEX FUNCTION LINFNC (L1, L2, L3)
*****
      FNC LINFNC RETURNS THE VALUE OF THE INTEGRATION          *

```

```

C*          OVER THE SOURCE TRIANGLE OF THE REMOVED SINGULARITY      *
C*          A:  (R-RI) . (P'-P) / R + (R-RI) . (P-PJ) / R          *
C*          WHERE P' / P / PJ IS THE PROJECTION OF POSITION VECTOR    *
C*          R' / R / RJ ONTO THE PLANE OF THE SOURCE TRIANGLE, AND   *
C*          RI / RJ IS THE POSITION VECTOR TO THE ITH / JTH VERTEX OF *
C*          THE OBSERVATION / SOURCE TRIANGLE.  THE ARGUMENTS OF    *
C*          LINFNC ARE THE LOCAL AREA COORDINATES OF THE           *
C*          OBSERVATION PT. WITHIN THE OBSERVATION TRIANGLE.      *
C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL L1, L2, L3, COEFF (NMAX), VECTS (NMAX, NMAX), R (3), UNIPOT,
*   LINPOT (3), NHAT (3), P (3), RMINRI (3), C2, PJ (3), PMINPJ (3), VDOT,
*   VERTS (4, 3)
INTEGER MPT, INDX, I, J, UNIFLAG, LINFLAG, CT1, CT2
COMPLEX DUM

```

```

REAL OBSV (NMAX, NMAX), SRCV (NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV
REAL RI (3), RJ (3)
COMMON /BASES/RI, RJ

```

```

COEFF (1) = L1
COEFF (2) = L2
COEFF (3) = L3
DO 10 I = 1, 3
  DO 20 J = 1, 3
    VECTS (I, J) = OBSV (I, J)

```

```

20 CONTINUE

```

```

10 CONTINUE

```

```

CALL SUM_VECTS (3, VECTS, COEFF, R)

```

```

UNIFLAG = 1

```

```

LINFLAG = 1

```

```

DO 30 CT1 = 1, 3

```

```

  DO 40 CT2 = 1, 3

```

```

    VERTS (CT1, CT2) = SRCV (CT1, CT2)

```

```

40 CONTINUE

```

```

30 CONTINUE

```

```

CALL POT_INTS_SS (3, R, VERTS, UNIFLAG, UNIPOT, LINFLAG, LINPOT, NHAT, P)

```

```

CALL VECT_DIFF (1.0, R, 1.0, RI, RMINRI)

```

```

C2 = VDOT (NHAT, RJ)

```

```

CALL VECT_DIFF (1.0, RJ, C2, NHAT, PJ)

```

```

CALL VECT_DIFF (1.0, P, 1.0, PJ, PMINPJ)

```

```

DUM = (0.0, 0.0)

```

```

LINFNC = DUM + VDOT (RMINRI, LINPOT) + VDOT (RMINRI, PMINPJ) * UNIPOT

```

```

RETURN

```

```

END

```

```

SUBROUTINE SUM_VECTS (NVECTS, VECTS, COEFF, VSUM)

```

```

C*****

```

```

C*   SR SUM_VECTS SUMS N VECTORS EACH MULTIPLIED BY A          *

```

```

C*   COEFFICIENT, I.E. VSUM = C1*V1 + C2*V2 + ... CN*VN.      *

```

```

C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL VECTS (NMAX, NMAX), COEFF (NMAX), VSUM (3)

```

```

INTEGER NVECTS, I, J, K

```

```

DO 10 I=1,3
VSUM(I)=0.0
10 CONTINUE

DO 20 J=1,NVECTS
DO 30 K=1,3
VSUM(K)=VSUM(K)+COEFF(J)*VECTS(J,K)
30 CONTINUE
20 CONTINUE
RETURN
END

```

BLOCK DATA

```

*****
THIS BLOCK DATA SEGMENT INITIALIZES COMMON BLOCK
INTCONST WHICH CONTAINS THE LOCAL AREA COORDINATE
INTEGRATION POINTS AND WEIGHTS USED IN ALL
INTEGRATION ROUTINES (SRS AC_INT, SINT, VECTINT).
THE 1, 4, AND 7-POINT FORMULAS WERE DEVELOPED
ESPECIALLY FOR TRIANGULAR REGIONS.
*****

```

```

REAL W(12),ZETA1(12),ZETA2(12),ZETA3(12)
COMMON /INTCONST/ZETA1,ZETA2,ZETA3,W

```

```

DATA ZETA1(1),ZETA2(1),ZETA3(1),W(1)

```

```

*/0.3333333333,0.3333333333,0.3333333333,1.0/

```

```

DATA ZETA1(2),ZETA2(2),ZETA3(2),W(2),
* ZETA1(3),ZETA2(3),ZETA3(3),W(3),
* ZETA1(4),ZETA2(4),ZETA3(4),W(4),
* ZETA1(5),ZETA2(5),ZETA3(5),W(5)
*/0.3333333333,0.3333333333,0.3333333333,-0.5625,
* 0.6,0.2,0.2,0.5208333333,
* 0.2,0.6,0.2,0.5208333333,
* 0.2,0.2,0.6,0.5208333333/

```

```

DATA ZETA1(6),ZETA2(6),ZETA3(6),W(6),
* ZETA1(7),ZETA2(7),ZETA3(7),W(7),
* ZETA1(8),ZETA2(8),ZETA3(8),W(8),
* ZETA1(9),ZETA2(9),ZETA3(9),W(9),
* ZETA1(10),ZETA2(10),ZETA3(10),W(10),
* ZETA1(11),ZETA2(11),ZETA3(11),W(11),
* ZETA1(12),ZETA2(12),ZETA3(12),W(12)
*/0.3333333333,0.3333333333,0.3333333333,0.225,
* 0.0597158717,0.4701420641,0.4701420641,0.1323941527,
* 0.4701420641,0.0597158717,0.4701420641,0.1323941527,
* 0.4701420641,0.4701420641,0.0597158717,0.1323941527,
* 0.7974269853,0.1012865073,0.1012865073,0.1259391805,
* 0.1012865073,0.7974269853,0.1012865073,0.1259391805,
* 0.1012865073,0.1012865073,0.7974269853,0.1259391805/

```

END

SUBROUTINE AC_INT(FNC,OUTM,INM,VOU)

```

*****
SR AC_INT PERFORMS A DOUBLE SURFACE INTEGRATION OVER
TWO TRIANGLES OF A FUNCTION IN LOCAL AREA COORDINATES.*
THE USER CAN CHOOSE 1,4, OR 7 POINTS FOR EACH SURFACE *
INTEGRATION. NOTE: THE RESULTING VALUE IS NORMALIZED *
TO THE AREAS OF THE TRIANGLES.
*****

```

```
INTEGER OUTM, INM, OUTINDX, ININDX, I, J, K, L, M, N, P
COMPLEX FNC, VOUT, VIN
```

```
REAL W(12), ZETA1(12), ZETA2(12), ZETA3(12)
COMMON /INTCONST/ZETA1, ZETA2, ZETA3, W
```

```
IF (OUTM.EQ.1) THEN
OUTINDX=1
ELSE IF (OUTM.EQ.4) THEN
OUTINDX=2
ELSE
OUTINDX=6
END IF
```

```
IF (INM.EQ.1) THEN
ININDX=1
ELSE IF (INM.EQ.4) THEN
ININDX=2
ELSE
ININDX=6
END IF
```

```
VOUT=(0.0,0.0)
```

```
DO 10 I=OUTINDX,OUTINDX+OUTM-1
VIN=(0.0,0.0)
DO 20 J=ININDX,ININDX+INM-1
VIN=VIN+W(J)*FNC(ZETA1(J),ZETA2(J),ZETA3(J),
* ZETA1(I),ZETA2(I),ZETA3(I))
20 CONTINUE
VOUT=VOUT+W(I)*VIN
10 CONTINUE
RETURN
END
```

```
SUBROUTINE SINT(MPT, FN, VINT)
```

```
C*****
C* SR SINT PERFORMS A SINGLE SURFACE INTEGRATION OVER *
C* A TRIANGLE OF A FUNCTION IN LOCAL AREA COORDINATES. *
C* THE USER CAN CHOOSE 1, 4, OR 7 POINTS FOR THE SURFACE *
C* INTEGRATION. NOTE: THE RESULTING VALUE IS NORMALIZED *
C* TO THE AREA OF THE TRIANGLE. *
C*****
```

```
INTEGER MPT, INDX, I
COMPLEX FN, VINT
```

```
REAL ZETA1(12), ZETA2(12), ZETA3(12), W(12)
COMMON /INTCONST/ZETA1, ZETA2, ZETA3, W
```

```
IF (MPT.EQ.1) THEN
INDX=1
ELSE IF (MPT.EQ.4) THEN
INDX=2
ELSE
INDX=6
END IF
```

```
VINT=(0.0,0.0)
```

```
DO 10 I=INDX,INDX+MPT-1
VINT=VINT+W(I)*FN(ZETA1(I),ZETA2(I),ZETA3(I))
10 CONTINUE

RETURN
END
```



```

SUBROUTINE POT_INTS_SS (NUMSIDES, R, VERTS, UNIFLAG, UNIPOT, LINFLAG,
*
*                               LINPOT, NHAT, P)
*****
SR POT_INTS_SS COMPUTES THE POTENTIAL INTEGRALS FOR
*
* UNIFORM AND LINEARLY VARYING SURFACE SOURCES
*
* DISTRIBUTED ON A PLANAR POLYGON S. THE INTEGRALS ARE
*
* CALCULATED USING CLOSED FORM ANALYTICAL EXPRESSIONS.
*****
REAL R(3), VERTS(4,3), UNIPOT, LINPOT(3), V1(3), V2(3), N(3), NMAG,
* NHAT(3), RIPLUS(3), RIMIN(3), UIHAT(3), D, CAPRIMIN, CAPRIPLUS,
* LIMIN, LIPLUS, MAGPIO, PIOHAT(3), RIO, VDOT, VMAG, UNIELEM,
* LINELEM(3), RDUM(3), C2, RPROJ(3), P(3), DIFF(3), MARG, PARG
INTEGER NUMSIDES, UNIFLAG, LINFLAG, I, J, K, M, Q, S, T, U, CT, INDX

DO 10 J=1,3
  VERTS (NUMSIDES+1, J)=VERTS (1, J)
10 CONTINUE

DO 20 K=1,3
  V1 (K)=VERTS (2, K)-VERTS (1, K)
  V2 (K)=VERTS (3, K)-VERTS (1, K)
20 CONTINUE
  CALL CROSS (V1, V2, N)
  NMAG=VMAG (N)
DO 30 M=1,3
  NHAT (M)=N (M) /NMAG
30 CONTINUE

DO 35 INDX=1,3
  RDUM (INDX)=VERTS (1, INDX)
35 CONTINUE
  CALL VECT_DIFF (1.0, R, 1.0, RDUM, DIFF)
  D=VDOT (NHAT, DIFF)

  CALL VECT_DIFF (1.0, R, D, NHAT, RPROJ)
  C2=VDOT (NHAT, RPROJ)
  CALL VECT_DIFF (1.0, RPROJ, C2, NHAT, P)

UNIPOT=0.0
DO 40 Q=1,3
  LINPOT (Q)=0.0
40 CONTINUE

DO 50 I=1, NUMSIDES

DO 60 S=1,3
  RIMIN (S)=VERTS (I, S)
  RIPLUS (S)=VERTS (I+1, S)
60 CONTINUE

CALL I_PARAMS (RIPLUS, RIMIN, R, NHAT, D, P,
* NHAT, CAPRIMIN, CAPRIPLUS, LIMIN, LIPLUS,
* MAGPIO, PIOHAT, RIO)

IF (UNIFLAG.EQ.1) THEN

IF (MAGPIO.LE.1.0E-10) THEN
  UNIELEM=0.0
  LSE
  IF (LIMIN.LT.0) THEN
    MARG=RIO**2 / (ABS (LIMIN) +SQRT (RIO**2+LIMIN**2))
  ELSE

```

```

MARG=CAPRIMIN+LIMIN
END IF
IF (LIPLUS.LT.0) THEN
PARG=RIO**2/(ABS(LIPLUS)+SQRT(RIO**2+LIPLUS**2))
ELSE
PARG=CAPRIPLUS+LIPLUS
END IF
NIELEM = VDOT(PIOHAT,UIHAT) * ( MAGPIO *
*      ALOG(PARG/MARG) -ABS(D) *
*      (ATAN2( (MAGPIO*LIPLUS), (RIO*RIO+ABS(D)*CAPRIPLUS))
*      -ATAN2( (MAGPIO*LIMIN), (RIO*RIO+ABS(D)*CAPRIMIN))) )
END IF
UNIPOT=UNIPOT+UNIELEM
END IF

IF (LINFLAG.EQ.1) THEN

IF ((RIO.LE.1.0E-10).AND.(D.LE.1.0E-10)) THEN
DO 65 T=1,3
LINELEM(CT)=UIHAT(CT) * (LIPLUS*CAPRIPLUS -
*      LIMIN*CAPRIMIN)
LINPOT(CT)=LINPOT(CT)+LINELEM(CT)
65 CONTINUE
ELSE
IF (LIMIN.LT.0) THEN
MARG=RIO**2/(ABS(LIMIN)+SQRT(RIO**2+LIMIN**2))
ELSE
MARG=CAPRIMIN+LIMIN
END IF
IF (LIPLUS.LT.0) THEN
PARG=RIO**2/(ABS(LIPLUS)+SQRT(RIO**2+LIPLUS**2))
ELSE
PARG=CAPRIPLUS+LIPLUS
END IF
DO 70 T=1,3
LINELEM(T) = UIHAT(T) * ( RIO*RIO *
*      LOG(PARG/MARG) +
*      LIPLUS*CAPRIPLUS-LIMIN*CAPRIMIN )
LINPOT(T)=LINPOT(T)+LINELEM(T)
70 CONTINUE
END IF
END IF

50 CONTINUE

IF (LINFLAG.EQ.1) THEN
DO 80 U=1,3
LINPOT(U)=LINPOT(U)/2.0
80 CONTINUE
END IF

RETURN
END

SUBROUTINE I_PARAMS(RIPLUS,RIMIN,R,NHAT,D,P,
*      UIHAT,CAPRIMIN,CAPRIPLUS,LIMIN,LIPLUS,
*      MAGPIO,PIOHAT,RIO)
C*****
C* SR I_PARAMS CALCULATES THE GEOMETRICAL PARAMETERS FOR *
C* POT_INTS_SS WHICH ARE DEPENDENT ON ITH POLYGON SIDE. *
C*****

REAL RIPLUS(3),RIMIN(3),R(3),NHAT(3),D,P(3),UIHAT(3),CAPRIMIN,
*      CAPRIPLUS,LIMIN,LIPLUS,MAGPIO,PIOHAT(3),RIO,PIMIN(3),

```

```

*      PIPLUS (3) , LVECT (3) , LIHAT (3) , DIFF (3) , PLUSDIFF (3) , PIO (3) , C2 ,
*      LMAG , VDOT , VMAG
INTEGER J , K , M

C2=VDOT (NHAT , RIMIN)
CALL VECT_DIFF (1.0 , RIMIN , C2 , NHAT , PIMIN)

C2=VDOT (NHAT , RIPLUS)
CALL VECT_DIFF (1.0 , RIPLUS , C2 , NHAT , PIPLUS)

CALL VECT_DIFF (1.0 , RIPLUS , 1.0 , RIMIN , LVECT)
LMAG=VMAG (LVECT)
DO 10 J=1 , 3
LIHAT (J)=LVECT (J) /LMAG
10 CONTINUE

CALL CROSS (LIHAT , NHAT , UIHAT)

CALL VECT_DIFF (1.0 , R , 1.0 , RIMIN , DIFF)
CAPRIMIN=VMAG (DIFF)

CALL VECT_DIFF (1.0 , R , 1.0 , RIPLUS , DIFF)
CAPRIPLUS=VMAG (DIFF)

CALL VECT_DIFF (1.0 , PIMIN , 1.0 , P , DIFF)
LIMIN=VDOT (DIFF , LIHAT)

CALL VECT_DIFF (1.0 , PIPLUS , 1.0 , P , PLUSDIFF)
LIPLUS=VDOT (PLUSDIFF , LIHAT)

MAGPIO=ABS (VDOT (PLUSDIFF , UIHAT))

IF (MAGPIO.LT.1.0E-10) THEN
DO 20 K=1 , 3
PIOHAT (K)=0.0
20 CONTINUE
ELSE
CALL VECT_DIFF (1.0 , PLUSDIFF , LIPLUS , LIHAT , PIO)
DO 30 M=1 , 3
PIOHAT (M)=PIO (M) /MAGPIO
30 CONTINUE
END IF

RIO=SQRT (MAGPIO*MAGPIO + D*D)

RETURN
END

SUBROUTINE VECT_DIFF (C1 , VECT1 , C2 , VECT2 , VDIFF)
*****
SR VECT_DIFF TAKES THE DIFFERENCE OF TWO VECTORS,
EACH MULTIPLIED BY A COEFFICIENT,
I.E. VDIFF = C1*VECT1 - C2*VECT2
*****

REAL C1 , C2 , VECT1 (3) , VECT2 (3) , VDIFF (3)
INTEGER I

DO 10 I=1 , 3
VDIFF (I)=C1*VECT1 (I) -C2*VECT2 (I)
10 CONTINUE
RETURN
END

```

```

      REAL FUNCTION VMAG(VECT)
C*****
C* FNC VMAG COMPUTES THE MAGNITUDE OF A VECTOR.
C*****

      REAL VECT(3),VDOT

      VMAG=SQRT(VDOT(VECT,VECT))
      RETURN
      END

      REAL FUNCTION VDOT(VECT1,VECT2)
C*****
C* FNC VDOT COMPUTES THE DOT PRODUCT OF TWO VECTORS.
C*****

      REAL VECT1(3),VECT2(3)

      VDOT=VECT1(1)*VECT2(1)+VECT1(2)*VECT2(2)+VECT1(3)*VECT2(3)
      RETURN
      END

      SUBROUTINE CROSS(VECT1,VECT2,RCROSS)
C*****
C* SR CROSS COMPUTES THE CROSS PRODUCT OF TWO VECTORS,
C* I.E. VCROSS = VECT1 X VECT2
C*****

      REAL RCROSS(3),VECT1(3),VECT2(3)

      RCROSS(1)=VECT1(2)*VECT2(3)-VECT1(3)*VECT2(2)
      RCROSS(2)=VECT2(1)*VECT1(3)-VECT1(1)*VECT2(3)
      RCROSS(3)=VECT1(1)*VECT2(2)-VECT2(1)*VECT1(2)
      RETURN
      END

      REAL FUNCTION SRCAREA()
C*****
C* FNC SRCAREA COMPUTES THE AREA OF THE SOURCE TRIANGLE
C* GIVEN THE POSITION VECTORS TO THE VERTICES OF THE
C* SOURCE TRIANGLE.
C*****

      INCLUDE 'DIM.INC'

      REAL VECT1(3),VECT2(3),VECT3(3),DIFF(3),BASE,LIHAT(3),UIHAT(3),
* HEIGHT,VDOT,VMAG,V1(3),V2(3),N(3),NMAG,NHAT(3)
      INTEGER I,J,K,S

      REAL OBSV(NMAX,NMAX),SRCV(NMAX,NMAX)
      COMMON /TRICONST/OBSV,SRCV

      DO 10 I=1,3
      V1(I)=SRCV(2,I)-SRCV(1,I)
      V2(I)=SRCV(3,I)-SRCV(1,I)
10 CONTINUE
      CALL CROSS(V1,V2,N)
      NMAG=VMAG(N)
      DO 20 J=1,3
      NHAT(J)=N(J)/NMAG
20 CONTINUE

```

```

DO 30 K=1,3
VECT1(K)=SRCV(1,K)
VECT2(K)=SRCV(2,K)
VECT3(K)=SRCV(3,K)
30 CONTINUE
CALL VECT_DIFF(1.0,VECT2,1.0,VECT1,DIFF)
BASE=VMAG(DIFF)
DO 40 S=1,3
LIHAT(S)=DIFF(S)/BASE
40 CONTINUE
CALL CROSS(LIHAT,NHAT,UIHAT)
CALL VECT_DIFF(1.0,VECT3,1.0,VECT1,DIFF)
HEIGHT=ABS(VDOT(DIFF,UIHAT))
SRCAREA=.5*BASE*HEIGHT
RETURN
END

```

```

COMPLEX FUNCTION RESFNC(L1,L2,L3)
*****
FNC RESFNC RETURNS THE VALUE OF THE INTEGRAND OF THE *
RESISTIVE TERM: (R-RM).(R-RN) *
RM/RN ARE THE POSITION VECTORS TO THE MTH/NTH *
VERTICES OF THE GIVEN OBSERVATION TRIANGLE. *
THE ARGUMENTS OF RESFNC ARE THE LOCAL AREA *
COORDINATES OF OBSERVATION POINTS WITHIN THE *
OBSERVATION TRIANGLE. *
*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL L1,L2,L3,COEFF(NMAX),VECTS(NMAX,NMAX),R(3),PN(3),PM(3),VDOT
INTEGER I,K
COMPLEX DUM

```

```

REAL OBSV(NMAX,NMAX),SRCV(NMAX,NMAX)
COMMON /TRICONST/OBSV,SRCV
REAL RM(3),RN(3)
COMMON /BASES/RM,RN

```

```

COEFF(1)=L1
COEFF(2)=L2
COEFF(3)=L3
DO 10 I=1,3
DO 20 K=1,3
VECTS(I,K)=OBSV(I,K)
20 CONTINUE
10 CONTINUE
CALL SUM_VECTS(3,VECTS,COEFF,R)

CALL VECT_DIFF(1.0,R,1.0,RN,PN)

CALL VECT_DIFF(1.0,R,1.0,RM,PM)
DUM=(0.0,0.0)
RESFNC=DUM+VDOT(PN,PM)
RETURN
END

```

```

SUBROUTINE USR(MESH_FILE,STYPE,RES_FILE,OUT_FILE,PTYPE,ALPHA,
*
* THETA,PHI, FIX,FANGO,ANG10,NUMPTS,STEPANG)
*****
SR USR_DATA PROMPTS USER FOR, AND READS IN, DATA *
PERTAINING TO *
-FILE NAMES *
-INCIDENT FIELD *
-RCS OBSERVATION CUT *

```

C* VIA STANDARD I/O. *

C*****

INCLUDE 'CONST.INC'

REAL FANGO, ANG10, ANG20, STEPANG, ALPHA, THETA, PHII,
* RNUMPTS, PI
INTEGER PTYPE, FIX, NUMPTS, FLAG, STYPE
CHARACTER*40 MESH_FILE, RES_FILE, OUT_FILE

10 FORMAT(A40)

12 FORMAT(A, F7.2, A)

PI=3.1415926536

100 CONTINUE

PRINT*, 'ENTER MESH FILE NAME:'

READ(*, 10) MESH_FILE

PRINT*, 'ENTER SURFACE TYPE (1-PEC, 2-RESISTIVE):'

READ*, STYPE

IF (STYPE.EQ.RESIST) THEN

PRINT*, 'ENTER RESISTIVITY FILE NAME:'

READ(*, 10) RES_FILE

END IF

PRINT*, 'ENTER OUTPUT FILE NAME: '

READ(*, 10) OUT_FILE

PRINT*, 'ENTER PATTERN (1-BISTATIC, 2-BACKSCATTER): '

READ*, PTYPE

PRINT*, 'ENTER E-FIELD POLARIZATION ANGLE ALPHA (IN DEGREES): '

READ*, ALPHA

IF (PTYPE.EQ.BISTAT) THEN

PRINT*, 'ENTER ANGLES OF INCIDENCE...'

PRINT*, 'PHI (IN DEGREES): '

READ*, PHII

PRINT*, 'THETA (IN DEGREES): '

READ*, THETA

END IF

PRINT*, 'ENTER CUT SPECIFICATIONS...'

PRINT*, 'FIX (1-PHI, 2-THETA): '

READ*, FIX

IF (FIX.EQ.ANGPHI) THEN

PRINT*, 'ENTER FIXED OBSERVATION ANGLE PHI (IN DEGREES): '

READ*, FANGO

PRINT*, 'ENTER START OBSERVATION ANGLE THETA (IN DEGREES): '

READ*, ANG10

PRINT*, 'ENTER STOP OBSERVATION ANGLE THETA (IN DEGREES): '

READ*, ANG20

ELSE

PRINT*, 'ENTER FIXED OBSERVATION ANGLE THETA (IN DEGREES): '

READ*, FANGO

PRINT*, 'ENTER START OBSERVATION ANGLE PHI (IN DEGREES): '

READ*, ANG10

PRINT*, 'ENTER STOP OBSERVATION ANGLE PHI (IN DEGREES): '

READ*, ANG20

END IF

PRINT*, 'ENTER NUMBER OF OBSERVATION POINTS: '

READ*, NUMPTS

PRINT*, ' '

PRINT*, '----- FILES -----'

PRINT*, ' MESH: ', MESH_FILE

IF (STYPE.EQ.RESIST) THEN

PRINT*, ' RESISTIVITY: ', RES_FILE

END IF

PRINT*, ' OUTPUT: ', OUT_FILE

PRINT*, ' '

```

PRINT*, '----- SURFACE TYPE -----'
IF (STYPE.EQ.PEC) THEN
PRINT*, '          PEC'
ELSE
PRINT*, '          RESISTIVE'
END IF
PRINT*, ' '
PRINT*, '----- PATTERN TYPE -----'
IF (PTYPE.EQ.BISTAT) THEN
PRINT*, '          BISTATIC'
ELSE
PRINT*, '          BACKSCATTER'
END IF
PRINT*, ' '
PRINT*, '----- INCIDENT FIELD -----'
WRITE(*,12) '          ALPHA: ',ALPHA,' DEG.'
IF (PTYPE.EQ.BISTAT) THEN
WRITE(*,12) '          PHI: ',PHII,' DEG.'
WRITE(*,12) '          THETA: ',THETAI,' DEG.'
END IF
PRINT*, ' '
PRINT*, '----- OBSERVATION ANGLES -----'
IF (FIX.EQ.ANGPHI) THEN
WRITE(*,12) '          PHI: ',FANGO,' DEG.'
WRITE(*,12) '          START THETA: ',ANG10,' DEG.'
WRITE(*,12) '          STOP THETA: ',ANG20,' DEG.'
ELSE
WRITE(*,12) '          THETA: ',FANGO,' DEG.'
WRITE(*,12) '          START PHI: ',ANG10,' DEG.'
WRITE(*,12) '          STOP PHI: ',ANG20,' DEG.'
END IF
PRINT*, ' '
PRINT*, '---- NUMBER OF OBSERVATION POINTS ----'
PRINT*, '          ',NUMPTS
PRINT*, ' '

```

```

PRINT*, 'ABOVE DATA O.K. (1-YES, 2-NO)?'
READ*, FLAG
IF (FLAG.EQ.NO) THEN
GOTO 100
END IF

```

```

ALPHA=ALPHA*PI/180.
PHII=PHII*PI/180.
THETAI=THETAI*PI/180.
FANGO=FANGO*PI/180.
ANG10=ANG10*PI/180.
ANG20=ANG20*PI/180.

```

```

RNUMPTS=1.0*(NUMPTS-1)
STEPANG=(ANG20-ANG10)/RNUMPTS
RETURN
END

```

```

SUBROUTINE EXCIT(NODCRDS,NINTEDG,EDGNODS,NTRIS,TRIEDGS,EDGLEN,
*          TRISIGN,ALPHA,THETAI,PHII,V)
*****
SR EXCIT COMPUTES ELEMENTS OF EXCITATION VECTOR V: *
<EI,FM>. THE INCIDENT ELECTRIC FIELD IS A PLANE WAVE *
OF UNIT AMPLITUDE: *
          EI = EHAT EXP(J 2 PI KHAT.R) *
WHERE EHAT IS UNIT POLARIZATION VECTOR DETERMINED *
FROM ANGLE ALPHA, AND KHAT IS UNIT PROPAGATION VECTOR *
DETERMINED FROM INCIDENT ANGLES THETAI AND PHII. *
*****

```

```

INCLUDE 'DIM.INC'

REAL NODCRDS (MAXNOD, 3), EDGLEN (MAXEDG), ALPHA, PHII, THETAI
INTEGER NINTEDG, EDGNODS (MAXEDG, 2), NTRIS, TRIEDGS (MAXTRI, 3),
*   TRISIGN (MAXTRI, 3), I, P, K, M (3), S, T
COMPLEX V (MAXZ), ANS, PWFNC, PJ
EXTERNAL PWFNC

REAL OBSV (NMAX, NMAX), SRCV (NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV
REAL RM (3), RN (3)
COMMON /BASES/RM, RN
REAL EHAT (3), KHAT (3), DUMY (3)
COMMON /INCID/EHAT, KHAT

PI=3.14159265359
PJ=(0.0,1.0)

EHAT (1)=COS (ALPHA)*COS (THETAI)*COS (PHII)-SIN (ALPHA)*SIN (PHII)
EHAT (2)=COS (ALPHA)*COS (THETAI)*SIN (PHII)+SIN (ALPHA)*COS (PHII)
EHAT (3)=-COS (ALPHA)*SIN (THETAI)

KHAT (1)=SIN (THETAI)*COS (PHII)
KHAT (2)=SIN (THETAI)*SIN (PHII)
KHAT (3)=COS (THETAI)

DO I=1, 3
DUMY (I)=EHAT (I)
ENDDO
EHAT (1)=KHAT (3)*DUMY (2)-DUMY (3)*KHAT (2)
EHAT (2)=DUMY (3)*KHAT (1)-DUMY (1)*KHAT (3)
EHAT (3)=DUMY (1)*KHAT (2)-DUMY (2)*KHAT (1)

DO 10 I=1, MAXZ
V (I)=(0.0, 0.0)
10 CONTINUE

DO 20 P=1, NTRIS
DO 30 K=1, 3
M (K)=TRIEDGS (P, K)
30 CONTINUE
CALL GET_VERTS (M, 0, NODCRDS, EDGNODS)

DO 40 S=1, 3
IF (M (S).LE.NINTEDG) THEN
DO 50 T=1, 3
RM (T)=OBSV (S, T)
50 CONTINUE
CALL SINT (7, PWFNC, ANS)
V (M (S))=V (M (S))+TRISIGN (P, S)*EDGLEN (M (S))*ANS
*   *2.0*PI*PJ

END IF
40 CONTINUE
20 CONTINUE

DO I=1, 3
DUMY (I)=EHAT (I)
ENDDO
EHAT (1)=KHAT (2)*DUMY (3)-DUMY (2)*KHAT (3)
EHAT (2)=DUMY (1)*KHAT (3)-DUMY (3)*KHAT (1)
EHAT (3)=DUMY (2)*KHAT (1)-DUMY (1)*KHAT (2)

RETURN

```


END

COMPLEX FUNCTION PWFNC(L1,L2,L3)

```
*****
FNC PWFNC RETURNS THE VALUE OF THE (R-RM).EI WHERE EI
IS INCIDENT PLANE WAVE OF UNIT AMPLITUDE, AND RM IS
THE POSITION VECTOR TO THE MTH VERTEX OF THE
OBSERVATION TRIANGLE. THE ARGUMENTS OF PWFNC ARE THE
LOCAL AREA COORDINATES OF OBSERVATION POINTS WITHIN
GIVEN OBSERVATION TRIANGLE.
*****
```

INCLUDE 'DIM.INC'

REAL L1,L2,L3,R(3),RMINRM(3),COEFF(NMAX),VECTS(NMAX,NMAX),VDOT,PI
INTEGER I,S
COMPLEX J

COMMON /TRICONST/OBSV,SRCV

REAL RM(3),RN(3)

COMMON /BASES/RM,RN

REAL EHAT(3),KHAT(3)

COMMON /INCID/EHAT,KHAT

PI=3.14159265359

J=(0.0,1.0)

COEFF(1)=L1

COEFF(2)=L2

COEFF(3)=L3

DO 20 I=1,3

DO 30 S=1,3

VECTS(I,S)=OBSV(I,S)

30 CONTINUE

20 CONTINUE

CALL SUM_VECTS(3,VECTS,COEFF,R)

CALL VECT_DIFF(1.0,R,1.0,RM,RMINRM)

PWFNC=VDOT(EHAT,RMINRM)*CEXP(J*2.0*PI*VDOT(KHAT,R))

RETURN

END

SUBROUTINE BACK_SUBST(A,VC,NET,IPVT,EN)

```
*****
*****
```

INCLUDE 'DIM.INC'

REAL RCOND

INTEGER NET,LDA,N,JOB,INDX,IPVT(MAXA)

COMPLEX A(MAXA,MAXA),VC(MAXA),EN(MAXA),DUM(MAXA)

LDA=MAXA

JOB=0

DO 10 INDX=1,NET

EN(INDX)=VC(INDX)

10 CONTINUE

CALL CGESL(A,LDA,NET,IPVT,EN,JOB)

RETURN

END

SUBROUTINE RCS(NODCRDS,NINTEDG,EDGNODS,NTRIS,TRIEDGS,EDGLEN,

* TRISIGN,ES,THETAO,PHIO,RCSTH,RCSPHI)

```
*****
*****
```

FNC RCS RETURNS THE VALUE OF THE RADAR CROSS SECTION

IN DB:

*

*

```

C*          RCS = LIM[R->INFINITY] 4 PI R^2 |ES|^2/|EI|^2      *
C*          THE ARGUMENTS OF RCS ARE THE OBSERVATION ANGLES  *
C*          THETAO AND PHIO IN RADIANS.                      *
C*
C*          NOTE: IN DETERMINING THE RCS, THE SPHERICAL      *
C*          FAR FIELD COEFFICIENTS SR/STH/SPHI ARE COMPUTED: *
C*          ECOMP=SCOMP E-JKR/KR                            *
C*          THESE VALUES MAY BE EASILY EXTRACTED.          *
C*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL NODCRDS (MAXNOD, 3), EDGLEN (MAXEDG), THETAO, PHIO,
*   PI, ETA
INTEGER NINTEDG, EDGNODS (MAXEDG, 2), NTRIS, TRIEDGS (MAXTRI, 3),
*   TRISIGN (MAXTRI, 3), INDX, CT, K, Q, S, T, N (3)
COMPLEX H (3), ANS (3), ES (MAXZ), VECT (3), DUM (3), SR, STH, SPHI, J

```

```

REAL OBSV (NMAX, NMAX), SRCV (NMAX, NMAX)
COMMON /TRICONST/OBSV, SRCV
REAL RM (3), RN (3)
COMMON /BASES/RM, RN
REAL RHAT (3)
COMMON /FFOBS/RHAT

```

```

J=(0.0,1.0)
PI=3.1415925953
ETA=376.7343091821

```

```

DO 10 INDX=1,3
VECT (INDX)=(0.0,0.0)
10 CONTINUE
RHAT (1)=SIN (THETAO) *COS (PHIO)
RHAT (2)=SIN (THETAO) *SIN (PHIO)
RHAT (3)=COS (THETAO)
DO 20 Q=1, NTRIS

DO 30 CT=1,3
N (CT)=TRIEDGS (Q, CT)
30 CONTINUE
CALL GET_VERTS (N, 1, NODCRDS, EDGNODS)

DO 40 K=1,3
IF (N (K) .LE. NINTEDG) THEN
DO 50 S=1,3
RN (S)=SRCV (K, S)
50 CONTINUE
ALL VECTINT (7, ANS)
DO 60 T=1,3
VECT (T)=VECT (T)+ES (N (K)) *EDGLEN (N (K)) *TRISIGN (Q, K) *
*   ANS (T) /2.0
60 CONTINUE
END IF
40 CONTINUE
20 CONTINUE
CALL CPLXCROSS (RHAT, VECT, DUM)
CALL CPLXCROSS (RHAT, DUM, H)
STH=(H (1) *COS (THETAO) *COS (PHIO)+H (2) *COS (THETAO) *
*   SIN (PHIO)-H (3) *SIN (THETAO))
SPHI=(-H (1) *SIN (PHIO)+H (2) *COS (PHIO))

RCSTH=10.*ALOG10 (4.*PI*CABS (STH)**2)
RCSPHI=10.*ALOG10 (4.*PI*CABS (SPHI)**2)
RETURN
END

```

```

SUBROUTINE VECTINT(MPT,VINT)
*****
SR VECTINT PERFORMS A SINGLE SURFACE INTEGRATION OVER      *
    A TRIANGLE OF VECTOR FUNCTION FFARG.  LOCAL AREA      *
    COORDINATES ARE USED.  THE CALLING ROUTINE SPECIFIES *
    1,4, OR 7 POINTS FOR THE SURFACE INTEGRATION.        *
    NOTE: THE RESULTING VALUE IS NORMALIZED TO THE AREA  *
    OF THE TRIANGLE.                                     *
*****

```

```

INCLUDE 'DIM.INC'

```

```

INTEGER MPT,INDX,CT,I,K
COMPLEX VINT(3),ANS(3)

```

```

REAL ZETA1(12),ZETA2(12),ZETA3(12),W(12)
COMMON /INTCONST/ZETA1,ZETA2,ZETA3,W

```

```

IF (MPT.EQ.1) THEN
INDX=1
ELSE IF (MPT.EQ.4) THEN
INDX=2
ELSE
INDX=6
END IF

```

```

DO 10 CT=1,3
VINT(CT)=(0.0,0.0)
10 CONTINUE

```

```

DO 20 I=INDX,INDX+MPT-1
CALL FFARG(ZETA1(I),ZETA2(I),ZETA3(I),ANS)
DO 30 K=1,3
VINT(K)=VINT(K)+W(I)*ANS(K)
30 CONTINUE
20 CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE FFARG(LP1,LP2,LP3,VFUNC)
*****
SR FFARG RETURNS THE COMPLEX VECTOR VALUE OF THE      *
    E-FAR FIELD ARGUMENT:                             *
     $EFF \sim (R' - RN) \exp(j 2 \pi R' .RHAT)$       *
    WHERE RN IS THE POSITION VECTOR TO THE NTH VERTEX OF *
    THE SOURCE TRIANGLE.  THE ARGUMENTS OF FFARG ARE THE *
    LOCAL AREA COORDINATES OF SOURCE POINTS WITHIN THE *
    GIVEN SOURCE TRIANGLE.                             *
*****

```

```

INCLUDE 'DIM.INC'

```

```

REAL LP1,LP2,LP3,PI,VECTS(NMAX,NMAX),COEFF(NMAX),RP(3),PN(3),
*   VDOT
INTEGER I,K,S
COMPLEX VFUNC(3),J

```

```

REAL OBSV(NMAX,NMAX),SRCV(NMAX,NMAX)
COMMON /TRICONST/OBSV,SRCV
REAL RM(3),RN(3)
COMMON /BASES/RM,RN
REAL RHAT(3)
COMMON /FFOBS/RHAT

```

J=(0.0,1.0)
PI=3.1415926536

```
COEFF(1)=LP1
COEFF(2)=LP2
COEFF(3)=LP3
DO 10 I=1,3
DO 20 K=1,3
VECTS(I,K)=SRCV(I,K)
20 CONTINUE
10 CONTINUE
CALL SUM_VECTS(3,VECTS,COEFF,RP)

CALL VECT_DIFF(1.0,RP,1.0,RN,PN)
DO 30 S=1,3
VFUNC(S)=PN(S)*CEXP(J*2.0*PI*VDOT(RP,RHAT))
30 CONTINUE
RETURN
END
```

SUBROUTINE CPLXCROSS(VECT1,VECT2,VCROSS)

```
C*****
C* SR Cmplxcross computes the complex cross product of a
C* real vector and a complex vector,
C* I.E. VCROSS = VECT1 X VECT2
C*****
```

```
REAL VECT1(3)
COMPLEX VCROSS(3),VECT2(3)

VCROSS(1)=VECT1(2)*VECT2(3)-VECT1(3)*VECT2(2)
VCROSS(2)=VECT2(1)*VECT1(3)-VECT1(1)*VECT2(3)
VCROSS(3)=VECT1(1)*VECT2(2)-VECT2(1)*VECT1(2)
RETURN
END
```

```
C*****
SUBROUTINE CGECO(A,LDA,N,IPVT,RCOND,Z)
```

```
C*****
INTEGER LDA,N,IPVT(1)
COMPLEX A(LDA,1),Z(1)
REAL RCOND
```

```
C
C CGECO FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION
C AND ESTIMATES THE CONDITION OF THE MATRIX.
```

```
C IF RCOND IS NOT NEEDED, CGEFA IS SLIGHTLY FASTER.
C TO SOLVE A*X = B , FOLLOW CGECO BY CGESL.
C TO COMPUTE INVERSE(A)*C , FOLLOW CGECO BY CGESL.
C TO COMPUTE DETERMINANT(A) , FOLLOW CGECO BY CGEDI.
C TO COMPUTE INVERSE(A) , FOLLOW CGECO BY CGEDI.
```

```
C ON ENTRY
```

```
C A COMPLEX(LDA, N)
C THE MATRIX TO BE FACTORED.
```

```
C LDA INTEGER
C THE LEADING DIMENSION OF THE ARRAY A .
```

```
C N INTEGER
C THE ORDER OF THE MATRIX A .
```

```
C ON RETURN
```

```
C A AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
```

WHICH WERE USED TO OBTAIN IT.
THE FACTORIZATION CAN BE WRITTEN $A = L*U$ WHERE
L IS A PRODUCT OF PERMUTATION AND UNIT LOWER
TRIANGULAR MATRICES AND U IS UPPER TRIANGULAR.

IPVT INTEGER(N)
AN INTEGER VECTOR OF PIVOT INDICES.

RCOND REAL
AN ESTIMATE OF THE RECIPROCAL CONDITION OF A .
FOR THE SYSTEM $A*X = B$, RELATIVE PERTURBATIONS
IN A AND B OF SIZE EPSILON MAY CAUSE
RELATIVE PERTURBATIONS IN X OF SIZE EPSILON/RCOND .
IF RCOND IS SO SMALL THAT THE LOGICAL EXPRESSION
 $1.0 + RCOND .EQ. 1.0$
IS TRUE, THEN A MAY BE SINGULAR TO WORKING
PRECISION. IN PARTICULAR, RCOND IS ZERO IF
EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
UNDERFLOWS.

Z COMPLEX(N)
A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
IF A IS CLOSE TO A SINGULAR MATRIX, THEN Z IS
AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
 $NORM(A*Z) = RCOND*NORM(A)*NORM(Z)$.

LINPACK. THIS VERSION DATED 07/14/77 .
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.

SUBROUTINES AND FUNCTIONS

LINPACK CGEFA
BLAS CAXPY, CDOTC, CSSCAL, SCASUM
FORTRAN ABS, AIMAG, AMAX1, CMLPX, CONJG, REAL

INTERNAL VARIABLES

COMPLEX CDOTC, EK, T, WK, WKM
REAL ANORM, S, SCASUM, SM, YNORM
INTEGER INFO, J, K, KB, KP1, L

COMPLEX ZDUM, ZDUM1, ZDUM2, CSIGN1
REAL CABS1
 $CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))$
 $CSIGN1(ZDUM1, ZDUM2) = CABS1(ZDUM1)*(ZDUM2/CABS1(ZDUM2))$

COMPUTE 1-NORM OF A

ANORM = 0.0E0
DO 10 J = 1, N
ANORM = AMAX1(ANORM, SCASUM(N, A(1, J), 1))

10 CONTINUE

FACTOR

CALL CGEFA(A, LDA, N, IPVT, INFO)

RCOND = $1/(NORM(A)*(ESTIMATE\ OF\ NORM(INVERSE(A))))$.
ESTIMATE = $NORM(Z)/NORM(Y)$ WHERE $A*Z = Y$ AND $CTRANS(A)*Y = E$.
CTRANS(A) IS THE CONJUGATE TRANSPOSE OF A .
THE COMPONENTS OF E ARE CHOSEN TO CAUSE MAXIMUM LOCAL
GROWTH IN THE ELEMENTS OF W WHERE $CTRANS(U)*W = E$.
THE VECTORS ARE FREQUENTLY RESCALED TO AVOID OVERFLOW.

SOLVE $CTRANS(U)*W = E$

```

EK = CMPLX(1.0E0,0.0E0)
DO 20 J = 1, N
  Z(J) = CMPLX(0.0E0,0.0E0)
20 CONTINUE
DO 100 K = 1, N
  IF (CABS1(Z(K)) .NE. 0.0E0) EK = CSIGN1(EK,-Z(K))
  IF (CABS1(EK-Z(K)) .LE. CABS1(A(K,K))) GO TO 30
  S = CABS1(A(K,K))/CABS1(EK-Z(K))
  CALL CSSCAL(N,S,Z,1)
  EK = CMPLX(S,0.0E0)*EK
30 CONTINUE
  WK = EK - Z(K)
  WKM = -EK - Z(K)
  S = CABS1(WK)
  SM = CABS1(WKM)
  IF (CABS1(A(K,K)) .EQ. 0.0E0) GO TO 40
  WK = WK/CONJG(A(K,K))
  WKM = WKM/CONJG(A(K,K))
  GO TO 50
40 CONTINUE
  WK = CMPLX(1.0E0,0.0E0)
  WKM = CMPLX(1.0E0,0.0E0)
50 CONTINUE
  KP1 = K + 1
  IF (KP1 .GT. N) GO TO 90
  DO 60 J = KP1, N
    SM = SM + CABS1(Z(J)+WKM*CONJG(A(K,J)))
    Z(J) = Z(J) + WK*CONJG(A(K,J))
    S = S + CABS1(Z(J))
60 CONTINUE
  IF (S .GE. SM) GO TO 80
  T = WKM - WK
  WK = WKM
  DO 70 J = KP1, N
    Z(J) = Z(J) + T*CONJG(A(K,J))
70 CONTINUE
80 CONTINUE
90 CONTINUE
  Z(K) = WK
100 CONTINUE
  S = 1.0E0/SCASUM(N,Z,1)
  CALL CSSCAL(N,S,Z,1)
C
C SOLVE CTRANS(L)*Y = V
C
DO 120 KB = 1, N
  K = N + 1 - KB
  IF (K .LT. N) Z(K) = Z(K) + CDOTC(N-K,A(K+1,K),1,Z(K+1),1)
  IF (CABS1(Z(K)) .LE. 1.0E0) GO TO 110
  S = 1.0E0/CABS1(Z(K))
  CALL CSSCAL(N,S,Z,1)
110 CONTINUE
  L = IPVT(K)
  T = Z(L)
  Z(L) = Z(K)
  Z(K) = T
120 CONTINUE
  S = 1.0E0/SCASUM(N,Z,1)
  CALL CSSCAL(N,S,Z,1)
C
C YNORM = 1.0E0
C
C SOLVE L*V = Y
C
DO 140 K = 1, N
  L = IPVT(K)

```

```

      T = Z(L)
      Z(L) = Z(K)
      Z(K) = T
      IF (K .LT. N) CALL CAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
      IF (CABS1(Z(K)) .LE. 1.0E0) GO TO 130
      S = 1.0E0/CABS1(Z(K))
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM
130  CONTINUE
140  CONTINUE
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM

      SOLVE  U*Z = V

      DO 160 KB = 1, N
        K = N + 1 - KB
        IF (CABS1(Z(K)) .LE. CABS1(A(K,K))) GO TO 150
        S = CABS1(A(K,K))/CABS1(Z(K))
        CALL CSSCAL(N,S,Z,1)
        YNORM = S*YNORM
150  CONTINUE
        IF (CABS1(A(K,K)) .NE. 0.0E0) Z(K) = Z(K)/A(K,K)
        IF (CABS1(A(K,K)) .EQ. 0.0E0) Z(K) = CMPLX(1.0E0,0.0E0)
        T = -Z(K)
        CALL CAXPY(K-1,T,A(1,K),1,Z(1),1)
160  CONTINUE
      MAKE ZNORM = 1.0
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM

      IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
      RETURN
      END

```

```

*****
SUBROUTINE CGEFA(A, LDA, N, IPVT, INFO)
*****

```

```

      INTEGER LDA, N, IPVT(1), INFO
      COMPLEX A(LDA,1)

```

CGEFA FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION.

CGEFA IS USUALLY CALLED BY CGECO, BUT IT CAN BE CALLED DIRECTLY WITH A SAVING IN TIME IF RCOND IS NOT NEEDED. (TIME FOR CGECO) = (1 + 9/N)*(TIME FOR CGEFA) .

ON ENTRY

A COMPLEX(LDA, N)
 THE MATRIX TO BE FACTORED.

LDA INTEGER
 THE LEADING DIMENSION OF THE ARRAY A .

N INTEGER
 THE ORDER OF THE MATRIX A .

ON RETURN

A AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
 WHICH WERE USED TO OBTAIN IT.
 THE FACTORIZATION CAN BE WRITTEN A = L*U WHERE

```

C           L IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C           TRIANGULAR MATRICES AND U IS UPPER TRIANGULAR.
C
C           IPVT  INTEGER(N)
C           AN INTEGER VECTOR OF PIVOT INDICES.
C
C           INFO  INTEGER
C           = 0  NORMAL VALUE.
C           = K  IF U(K,K) .EQ. 0.0 . THIS IS NOT AN ERROR
C           CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C           INDICATE THAT CGESL OR CGEDI WILL DIVIDE BY ZERO
C           IF CALLED. USE RCOND IN CGECO FOR A RELIABLE
C           INDICATION OF SINGULARITY.
C
C           LINPACK. THIS VERSION DATED 07/14/77 .
C           CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C           SUBROUTINES AND FUNCTIONS
C
C           BLAS CAXPY, SCAL, ICAMAX
C           FORTRAN ABS, AIMAG, CMPLX, REAL
C
C           INTERNAL VARIABLES
C
C           COMPLEX T
C           INTEGER ICAMAX, J, K, KP1, L, NM1
C
C           COMPLEX ZDUM
C           REAL CABS1
C           CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))
C
C           GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C           INFO = 0
C           NM1 = N - 1
C           IF (NM1 .LT. 1) GO TO 70
C           DO 60 K = 1, NM1
C             KP1 = K + 1
C
C           FIND L = PIVOT INDEX
C
C           L = ICAMAX(N-K+1, A(K,K), 1) + K - 1
C           IPVT(K) = L
C
C           ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
C           IF (CABS1(A(L,K)) .EQ. 0.0E0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
C           IF (L .EQ. K) GO TO 10
C             T = A(L,K)
C             A(L,K) = A(K,K)
C             A(K,K) = T
C           10  CONTINUE
C
C           COMPUTE MULTIPLIERS
C
C           T = -CMPLX(1.0E0, 0.0E0) / A(K,K)
C           CALL CSCAL(N-K, T, A(K+1, K), 1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
C           DO 30 J = KP1, N
C             T = A(L, J)
C             IF (L .EQ. K) GO TO 20

```



```

C          DO 10 J = 1, P
C          CALL CGESL(A,LDA,N,IPVT,C(1,J),0)
C          10 CONTINUE
C
C LINPACK. THIS VERSION DATED 07/14/77 .
C CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C SUBROUTINES AND FUNCTIONS
C
C BLAS CAXPY,CDOTC
C FORTRAN CONJG
C
C INTERNAL VARIABLES
C
C COMPLEX CDOTC,T
C INTEGER K,KB,L,NM1
C
C NM1 = N - 1
C IF (JOB .NE. 0) GO TO 50
C
C JOB = 0 , SOLVE A * X = B
C FIRST SOLVE L*Y = B
C
C IF (NM1 .LT. 1) GO TO 30
C DO 20 K = 1, NM1
C   L = IPVT(K)
C   T = B(L)
C   IF (L .EQ. K) GO TO 10
C     B(L) = B(K)
C     B(K) = T
10   CONTINUE
C     CALL CAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
20   CONTINUE
30   CONTINUE
C
C NOW SOLVE U*X = Y
C
C DO 40 KB = 1, N
C   K = N + 1 - KB
C   B(K) = B(K)/A(K,K)
C   T = -B(K)
C   CALL CAXPY(K-1,T,A(1,K),1,B(1),1)
40   CONTINUE
C GO TO 100
50 CONTINUE
C
C JOB = NONZERO, SOLVE CTRANS(A) * X = B
C FIRST SOLVE CTRANS(U)*Y = B
C
C DO 60 K = 1, N
C   T = CDOTC(K-1,A(1,K),1,B(1),1)
C   B(K) = (B(K) - T)/CONJG(A(K,K))
60   CONTINUE
C
C NOW SOLVE CTRANS(L)*X = Y
C
C IF (NM1 .LT. 1) GO TO 90
C DO 80 KB = 1, NM1
C   K = N - KB
C   B(K) = B(K) + CDOTC(N-K,A(K+1,K),1,B(K+1),1)
C   L = IPVT(K)
C   IF (L .EQ. K) GO TO 70
C     T = B(L)
C     B(L) = B(K)
C     B(K) = T
70   CONTINUE

```

```

                A(L,J) = A(K,J)
                A(K,J) = T
20             CONTINUE
                CALL CAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
30             CONTINUE
             GO TO 50
40             CONTINUE
                INFO = K
50             CONTINUE
60             CONTINUE
70             CONTINUE
             IPVT(N) = N
             IF (CABS1(A(N,N)) .EQ. 0.0E0) INFO = N
             RETURN
             END

```

```

SUBROUTINE CGESL(A, LDA, N, IPVT, B, JOB)

```

```

INTEGER LDA, N, IPVT(1), JOB
COMPLEX A(LDA, 1), B(1)

```

CGESL SOLVES THE COMPLEX SYSTEM
 $A * X = B$ OR $CTRANS(A) * X = B$
 USING THE FACTORS COMPUTED BY CGECO OR CGEFA.

ON ENTRY

A COMPLEX(LDA, N)
 THE OUTPUT FROM CGECO OR CGEFA.

LDA INTEGER
 THE LEADING DIMENSION OF THE ARRAY A .

N INTEGER
 THE ORDER OF THE MATRIX A .

IPVT INTEGER(N)
 THE PIVOT VECTOR FROM CGECO OR CGEFA.

B COMPLEX(N)
 THE RIGHT HAND SIDE VECTOR.

JOB INTEGER
 = 0 TO SOLVE $A * X = B$,
 = NONZERO TO SOLVE $CTRANS(A) * X = B$ WHERE
 $CTRANS(A)$ IS THE CONJUGATE TRANSPOSE.

B THE SOLUTION VECTOR X .

ERROR CONDITION

A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS A ZERO ON THE DIAGONAL. TECHNICALLY THIS INDICATES SINGULARITY BUT IT IS OFTEN CAUSED BY IMPROPER ARGUMENTS OR IMPROPER SETTING OF LDA . IT WILL NOT OCCUR IF THE SUBROUTINES ARE CALLED CORRECTLY AND IF CGECO HAS SET RCOND .GT. 0.0 OR CGEFA HAS SET INFO .EQ. 0 .

TO COMPUTE $INVERSE(A) * C$ WHERE C IS A MATRIX
 WITH P COLUMNS
 CALL CGECO(A, LDA, N, IPVT, RCOND, Z)
 IF (RCOND IS TOO SMALL) GO TO ...

```

C          DO 10 J = 1, P
C          CALL CGESL(A,LDA,N,IPVT,C(1,J),0)
C          10 CONTINUE
C
C          LINPACK. THIS VERSION DATED 07/14/77 .
C          CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C          SUBROUTINES AND FUNCTIONS
C
C          BLAS CAXPY,CDOTC
C          FORTRAN CONJG
C
C          INTERNAL VARIABLES
C
C          COMPLEX CDOTC,T
C          INTEGER K,KB,L,NM1
C
C          NM1 = N - 1
C          IF (JOB .NE. 0) GO TO 50
C
C          JOB = 0 , SOLVE  A * X = B
C          FIRST SOLVE  L*Y = B
C
C          IF (NM1 .LT. 1) GO TO 30
C          DO 20 K = 1, NM1
C            L = IPVT(K)
C            T = B(L)
C            IF (L .EQ. K) GO TO 10
C            B(L) = B(K)
C            B(K) = T
10          CONTINUE
C            CALL CAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
20          CONTINUE
30          CONTINUE
C
C          NOW SOLVE  U*X = Y
C
C          DO 40 KB = 1, N
C            K = N + 1 - KB
C            B(K) = B(K)/A(K,K)
C            T = -B(K)
C            CALL CAXPY(K-1,T,A(1,K),1,B(1),1)
40          CONTINUE
C          GO TO 100
50          CONTINUE
C
C          JOB = NONZERO, SOLVE  CTRANS(A) * X = B
C          FIRST SOLVE  CTRANS(U)*Y = B
C
C          DO 60 K = 1, N
C            T = CDOTC(K-1,A(1,K),1,B(1),1)
C            B(K) = (B(K) - T)/CONJG(A(K,K))
60          CONTINUE
C
C          NOW SOLVE CTRANS(L)*X = Y
C
C          IF (NM1 .LT. 1) GO TO 90
C          DO 80 KB = 1, NM1
C            K = N - KB
C            B(K) = B(K) + CDOTC(N-K,A(K+1,K),1,B(K+1),1)
C            L = IPVT(K)
C            IF (L .EQ. K) GO TO 70
C            T = B(L)
C            B(L) = B(K)
C            B(K) = T
70          CONTINUE

```

```

80    CONTINUE
90    CONTINUE
100  CONTINUE
      RETURN
      END

```

```

*****
      INTEGER FUNCTION ICAMAX(N,CX, INCX)
*****

```

FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
 JACK DONGARRA, LINPACK, 3/11/78.

```

COMPLEX CX(1)
REAL SMAX
INTEGER I, INCX, IX, N
COMPLEX ZDUM
REAL CABS1
CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))

```

```

ICAMAX = 0
IF( N .LT. 1 ) RETURN
ICAMAX = 1
IF(N.EQ.1)RETURN
IF(INCX.EQ.1)GO TO 20

```

CODE FOR INCREMENT NOT EQUAL TO 1

```

IX = 1
SMAX = CABS1(CX(1))
IX = IX + INCX
DO 10 I = 2, N
  IF(CABS1(CX(IX)).LE.SMAX) GO TO 5
  ICAMAX = I
  SMAX = CABS1(CX(IX))
5   IX = IX + INCX
10  CONTINUE
    RETURN

```

CODE FOR INCREMENT EQUAL TO 1

```

20  SMAX = CABS1(CX(1))
    DO 30 I = 2, N
      IF(CABS1(CX(I)).LE.SMAX) GO TO 30
      ICAMAX = I
      SMAX = CABS1(CX(I))
30  CONTINUE
    RETURN
    END

```

```

*****
      REAL FUNCTION SCASUM(N,CX, INCX)
*****

```

TAKES THE SUM OF THE ABSOLUTE VALUES OF A COMPLEX VECTOR AND
 RETURNS A SINGLE PRECISION RESULT.
 JACK DONGARRA, LINPACK, 3/11/78.

```

COMPLEX CX(1)
REAL STEMP
INTEGER I, INCX, N, NINCX

```

```

SCASUM = 0.0E0
STEMP = 0.0E0
IF(N.LE.0)RETURN
IF(INCX.EQ.1)GO TO 20

```

```

C
C      CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        STEMP = STEMP + ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))
10    CONTINUE
      SCASUM = STEMP
      RETURN

C
C      CODE FOR INCREMENT EQUAL TO 1
C
      DO 30 I = 1,N
        STEMP = STEMP + ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))
30    CONTINUE
      SCASUM = STEMP
      RETURN
      END

C*****
      SUBROUTINE CAXPY(N,CA,CX,INCX,CY,INCY)
C*****
C      CONSTANT TIMES A VECTOR PLUS A VECTOR.
C      JACK DONGARRA, LINPACK, 3/11/78.
C
      COMPLEX CX(1),CY(1),CA
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF (ABS(REAL(CA)) + ABS(AIMAG(CA)) .EQ. 0.0 ) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20

C      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C      NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CY(IY) = CY(IY) + CA*CX(IX)
        IX = IX + INCX
        IY = IY + INCY
10    CONTINUE
      RETURN

C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
      DO 30 I = 1,N
        CY(I) = CY(I) + CA*CX(I)
30    CONTINUE
      RETURN
      END

C*****
      COMPLEX FUNCTION CDOTC(N,CX,INCX,CY,INCY)
C*****
C      FORMS THE DOT PRODUCT OF TWO VECTORS, CONJUGATING THE FIRST
C      VECTOR.
C      JACK DONGARRA, LINPACK, 3/11/78.
C
      COMPLEX CX(1),CY(1),CTEMP
      INTEGER I,INCX,INCY,IX,IY,N
C

```

```

CTEMP = (0.0,0.0)
CDOTC = (0.0,0.0)
IF(N.LE.0)RETURN
IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20

```

```

CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
NOT EQUAL TO 1

```

```

IX = 1
IY = 1
IF(INCX.LT.0)IX = (-N+1)*INCX + 1
IF(INCY.LT.0)IY = (-N+1)*INCY + 1
DO 10 I = 1,N
  CTEMP = CTEMP + CONJG(CX(IX))*CY(IY)
  IX = IX + INCX
  IY = IY + INCY
10 CONTINUE
CDOTC = CTEMP
RETURN

```

```

CODE FOR BOTH INCREMENTS EQUAL TO 1

```

```

20 DO 30 I = 1,N
  CTEMP = CTEMP + CONJG(CX(I))*CY(I)
30 CONTINUE
CDOTC = CTEMP
RETURN
END

```

```

*****
SUBROUTINE CSSCAL(N,SA,CX,INCX)
*****

```

```

SCALES A COMPLEX VECTOR BY A REAL CONSTANT.
JACK DONGARRA, LINPACK, 3/11/78.

```

```

COMPLEX CX(1)
REAL SA
INTEGER I, INCX, N, NINCX

```

```

IF(N.LE.0)RETURN
IF(INCX.EQ.1)GO TO 20

```

```

CODE FOR INCREMENT NOT EQUAL TO 1

```

```

NINCX = N*INCX
DO 10 I = 1,NINCX, INCX
  CX(I) = CMPLX(SA*REAL(CX(I)), SA*AIMAG(CX(I)))
10 CONTINUE
RETURN

```

```

CODE FOR INCREMENT EQUAL TO 1

```

```

20 DO 30 I = 1,N
  CX(I) = CMPLX(SA*REAL(CX(I)), SA*AIMAG(CX(I)))
30 CONTINUE
RETURN
END
SUBROUTINE CSCAL(N,CA,CX,INCX)

```

```

SCALES A VECTOR BY A CONSTANT.
JACK DONGARRA, LINPACK, 3/11/78.

```

```

COMPLEX CA,CX(1)
INTEGER I, INCX, N, NINCX

```

```
IF (N.LE.0) RETURN
IF (INCX.EQ.1) GO TO 20
```

```
C
C     CODE FOR INCREMENT NOT EQUAL TO 1
```

```
C
C     NINCX = N*INCX
C     DO 10 I = 1, NINCX, INCX
C         CX(I) = CA*CX(I)
10 CONTINUE
RETURN
```

```
C
C     CODE FOR INCREMENT EQUAL TO 1
```

```
C
C     20 DO 30 I = 1, N
C         CX(I) = CA*CX(I)
30 CONTINUE
RETURN
END
```

```
C=====
C     THIS SUBROUTINE GENERATES THE FEM MATRIX
C=====
```

```
1 SUBROUTINE FEM(NED, NES, EDST, A)
   PARAMETER (MAXA=2000)
   INTEGER EDNA(15000), GNN(15000, 2), TAB(6, 3), SED(2000), EDST(MAXA, 2)
   CHARACTER BIT(6)*1
   REAL XYZ(3000, 3), A1(6, 6), X(500), Y(500), Z(500), B1(6, 6), EPS(10)
   COMPLEX A(MAXA, MAXA), B(MAXA, MAXA)
   LOGICAL MATZ
   COMMON /BANK/EDNA, GNN, XYZ, EPS, MAT
   COMMON /MESS/X, Y, Z
   PI=3.14159265359
   NL=1
   OPEN(1, FILE='EGLOB')
   OPEN(2, FILE='EDGY')
   OPEN(3, FILE='ENODDY')
   OPEN(7, FILE='ESURFC')
   READ(1, *) (EPS(I), I=1, NL)
   READ(1, *) NN
   DO I=1, NN
      READ(3, *) K, X(I), Y(I), Z(I)
   ENDDO
   READ(1, *) NEL
   DO I=1, 6*NEL
      ITEMP=1+(I/6.)
      READ(1, *) ELM, EDNA(I), GNN(I, 1), GNN(I, 2), MAT(ITEMP)
   ENDDO
   DO I=1, NED
      READ(2, *) K, XYZ(I, 1), XYZ(I, 2), XYZ(I, 3)
   ENDDO
   READ(7, *) (SED(I), I=1, NES)
   CLOSE(1)
   CLOSE(2)
   CLOSE(3)
   CLOSE(7)
   NET=NED-NES
   DO I=1, NET
      DO J=1, NET
         A(I, J)=CMPLX(0., 0.)
         B(I, J)=CMPLX(0., 0.)
      ENDDO
   ENDDO
C WRITE(6, *) 'GENERATING FEM MATRIX'
DO I=1, MAXA
   EDST(I, 1)=0
```

```

EDST(I,2)=0
ENDDO
NPTRX=1
DO I=1,NEL
  CALL CRUX(I,A1,B1,TAB)
  DO IJ=1,6
    BIT(IJ)='0'
  ENDDO
  DO 100 J=1,6
    DO ICHK=1,NES
      IF (TAB(J,3).EQ.SED(ICHK)) THEN
        BIT(J)='1'
        GO TO 100
      ENDIF
    ENDDO
  ENDDO
DO J=1,6
  IF (BIT(J).EQ.'0') THEN
    IF (EDST(TAB(J,3),2).NE.TAB(J,3)) THEN
      EDST(TAB(J,3),1)=NPTRX
      EDST(TAB(J,3),2)=TAB(J,3)
      NPTRX=NPTRX+1
    ENDIF
  ENDIF
ENDDO
DO J=1,6
  DO K=1,6
    IF ((BIT(J).EQ.'0').AND.(BIT(K).EQ.'0')) THEN
      MMM=EDST(TAB(J,3),1)
      NNN=EDST(TAB(K,3),1)
      A(MMM,NNN)=A(MMM,NNN)+(A1(J,K)-4.*PI*PI*B1(J,K))
    ENDIF
  ENDDO
ENDDO
ENDDO
WRITE(6,*)'FINISHED FEM MATRIX GENERATION'
IF (NET.NE.(NPTRX-1)) THEN
  WRITE(6,*)'ERROR IN MATRIX ASSEMBLY'
  STOP
ENDIF
IR=0
DO I=1,NPTRX-1
  DO J=1,NPTRX-1
    IF (REAL(A(I,J)-A(J,I)).GT..000001) THEN
      WRITE(6,*)A(I,J),A(J,I),I,J
      IR=1
    ENDIF
  ENDDO
ENDDO
IF (IR.EQ.0) THEN
  WRITE(6,*)'SYMMETRY TEST PASSED'
ELSE
  WRITE(6,*)'SYMMETRY TEST FAILED'
ENDIF

```

END

```

=====
SUBROUTINE CRUX(L,A,B,TAB)
INTEGER EDNA(15000),GNN(15000,2),TAB(6,3),MAT(2500)
REAL XYZ(3000,3),X(500),Y(500),Z(500),A(6,6),B(6,6),F(6,3),
1   G(6,3),TMP(3),EPS(10)
COMMON /BANK/EDNA,GNN,XYZ,EPS,MAT
COMMON /MESS/X,Y,Z
COMMON /LOCAL/SUMX,SUMY,SUMZ,XX,YY,ZZ,XY,YZ,ZX
COMMON /FGS/F,G
LV=6*(L-1)

```



```

DO J=1,6
  TAB(J,1)=GNN(LV+J,1)
  TAB(J,2)=GNN(LV+J,2)
  TAB(J,3)=EDNA(LV+J)
ENDDO
C....SORTING THE ARRAY 'TAB' ARRANGES IT ACCORDING TO LOCAL NODE NOS. SO
C   THAT THE ARRAY LOOKS LIKE THE ONE IN FILE 'INPUT'.
  CALL SORT(TAB)
C....'CALC' CALCULATES SOME DATA CORRESPONDING TO THE ELEMENT
  CALL CALC(TAB(1,1),TAB(1,2),TAB(2,2),TAB(3,2))
C....'VOLUME' COMPUTES SIX TIMES THE VOLUME OF THE TETRAHEDRAL ELEMENT
  CALL VOLUME(TAB(1,3),TAB(2,3),TAB(3,3),VOL)
DO J=1,6
  CALL FCROSS(X(TAB(7-J,1)),Y(TAB(7-J,1)),Z(TAB(7-J,1)),X(TAB(
1     7-J,2)),Y(TAB(7-J,2)),Z(TAB(7-J,2)),TMP)
C....'BJ' STORES THE LENGTH OF THE 'J' TH EDGE.
  BJ=SQRT((XYZ(TAB(J,3),1)**2)+(XYZ(TAB(J,3),2)**2)+(XYZ
1     (TAB(J,3),3)**2))
  DO K=1,3
    F(J,K)=BJ*TMP(K)/VOL
    G(J,K)=BJ*XYZ(TAB(7-J,3),K)/VOL
  ENDDO
ENDDO
CALL FGSGN(TAB)
VOL=VOL/6.
DO J=1,6
  DO K=1,6
    A(J,K)=4.*DOT(J,K,G)*VOL
    B(J,K)=F1(J,K,F,G)*EPS(MAT(L))*VOL
  ENDDO
ENDDO
RETURN
END

```

```

C=====
SUBROUTINE SORT (B)
INTEGER TAB(6,3)
NC=1
DO IK=4,2,-1
  IF (NC.EQ.1) THEN
    IJ=6
  ELSE
    IJ=IK
  ENDIF
  DO II=1,IJ-1
    DO J=IJ,II+1,-1
      IF (TAB(J,NC).LT.TAB(J-1,NC)) THEN
        DO JK=1,3
          CALL EXCHG(TAB(J,JK),TAB(J-1,JK))
        ENDDO
      ENDIF
    ENDDO
  ENDDO
  NC=2
ENDDO
IF (TAB(5,2).LT.TAB(4,2)) THEN
  DO JK=1,3
    CALL EXCHG(TAB(5,JK),TAB(4,JK))
  ENDDO
ENDIF
CALL EXCHG(TAB(5,1),TAB(5,2))
RETURN
END

```

```

C=====
SUBROUTINE VOLUME(J1,J2,J3,V)
INTEGER EDNA(15000),GNN(15000,2),MAT(2500)
REAL XYZ(3000,3),B1(3),B2(3),B3(3),EPS(10)

```

```

COMMON /BANK/EDNA,GNN,XYZ,EPS,MAT
DO IT=1,3
  B1(IT)=XYZ(J1,IT)
  B2(IT)=XYZ(J2,IT)
  B3(IT)=XYZ(J3,IT)
ENDDO
V=ABS(B1(1)*((B2(2)*B3(3))-(B2(3)*B3(2)))-B1(2)*((B2(1)*B3
1  (3))-(B2(3)*B3(1)))+B1(3)*((B2(1)*B3(2))-(B2(2)*B3(1))))
RETURN
END

```

```

=====
SUBROUTINE FCROSS(X1,Y1,Z1,X2,Y2,Z2,TEMPO)
REAL TEMPO(3)
TEMPO(1)=Y1*Z2-Y2*Z1
TEMPO(2)=Z1*X2-Z2*X1
TEMPO(3)=X1*Y2-X2*Y1
RETURN
END

```

```

=====
SUBROUTINE CALC(J1,J2,J3,J4)
REAL X(500),Y(500),Z(500)
COMMON /MESS/X,Y,Z
COMMON /LOCAL/SUMX,SUMY,SUMZ,XX,YY,ZZ,XY,YZ,ZX
SUMX=X(J1)+X(J2)+X(J3)+X(J4)
SUMY=Y(J1)+Y(J2)+Y(J3)+Y(J4)
SUMZ=Z(J1)+Z(J2)+Z(J3)+Z(J4)
XX=SUMX*SUMX+X(J1)*X(J1)+X(J2)*X(J2)+X(J3)*X(J3)+X(J4)*X(J4)
YY=SUMY*SUMY+Y(J1)*Y(J1)+Y(J2)*Y(J2)+Y(J3)*Y(J3)+Y(J4)*Y(J4)
ZZ=SUMZ*SUMZ+Z(J1)*Z(J1)+Z(J2)*Z(J2)+Z(J3)*Z(J3)+Z(J4)*Z(J4)
XY=SUMX*SUMY+X(J1)*Y(J1)+X(J2)*Y(J2)+X(J3)*Y(J3)+X(J4)*Y(J4)
YZ=SUMY*SUMZ+Y(J1)*Z(J1)+Y(J2)*Z(J2)+Y(J3)*Z(J3)+Y(J4)*Z(J4)
ZX=SUMZ*SUMX+Z(J1)*X(J1)+Z(J2)*X(J2)+Z(J3)*X(J3)+Z(J4)*X(J4)
RETURN
END

```

```

=====
REAL FUNCTION DOT(J1,J2,VEC)
REAL VEC(6,3)
DOT=(VEC(J1,1)*VEC(J2,1))+(VEC(J1,2)*VEC(J2,2))+(VEC(J1,3)*VEC
1  (J2,3))
RETURN
END

```

```

=====
REAL FUNCTION F1(J1,J2,F,G)
REAL F(6,3),G(6,3),TMP1(3),TMP2(3)
COMMON /LOCAL/SUMX,SUMY,SUMZ,XX,YY,ZZ,XY,YZ,ZX
CALL FCROSS(F(J1,1),F(J1,2),F(J1,3),G(J2,1),G(J2,2),G(J2,3),TMP1)
CALL FCROSS(F(J2,1),F(J2,2),F(J2,3),G(J1,1),G(J1,2),G(J1,3),TMP2)
TERM1=DOT(J1,J2,F)
TERM2=((TMP1(1)+TMP2(1))*SUMX+(TMP1(2)+TMP2(2))*SUMY+
1  (TMP1(3)+TMP2(3))*SUMZ)/4.
TERM3=(G(J1,2)*G(J2,2)+G(J1,3)*G(J2,3))*XX+(G(J1,1)*G(J2,1)+
1  G(J1,3)*G(J2,3))*YY+(G(J1,1)*G(J2,1)+G(J1,2)*G(J2,2))*ZZ
2  -(G(J1,1)*G(J2,2)+G(J1,2)*G(J2,1))*XY-(G(J1,1)*G(J2,3)+
3  G(J1,3)*G(J2,1))*ZX-(G(J1,2)*G(J2,3)+G(J1,3)*G(J2,2))*YZ
F1=TERM1+TERM2+(TERM3/20.)
RETURN
END

```

```

=====
SUBROUTINE EXCHG(J1,J2)
NTMP=J1
J1=J2
J2=NTMP
RETURN
END

```

```

+++++C
SUBROUTINE FGSGN(TAB)

```

```

      INTEGER EDNA(15000),GNN(15000,2),TAB(6,3),MAT(2500)
      REAL XYZ(3000,3),X(500),Y(500),Z(500),A(6,6),B(6,6),F(6,3),
1      G(6,3),TMP(3),EPS(10),V1(3),V2(3),GXEL(3),GXEL2(3),EHAT(3)
1      ,E0(3)
      COMMON /BANK/EDNA,GNN,XYZ,EPS,MAT
      COMMON /MESS/X,Y,Z
      COMMON /FGS/F,G
      DO 100 I=1,6
      CALL FCROSS(G(I,1),G(I,2),G(I,3),X(TAB(I,1)),Y(TAB(I,1)),
1      Z(TAB(I,1)),GXEL)
      DO K=1,3
      V1(K)=F(I,K)+GXEL(K)
      ENDDO
      IF(TAB(I,2).GT.TAB(I,1))THEN
      DO K=1,3
      EHAT(K)=-XYZ(TAB(I,3),K)
      ENDDO
      ELSE
      DO K=1,3
      EHAT(K)=XYZ(TAB(I,3),K)
      ENDDO
      ENDIF
      S=0.
      AEHAT=SQRT(EHAT(1)**2+EHAT(2)**2+EHAT(3)**2)
      DO K=1,3
      S=S+V1(K)*EHAT(K)/AEHAT
      ENDDO
      IF(S.GT.0.)THEN
      GOTO 100
      ELSE
      DO K=1,3
      F(I,K)=-F(I,K)
      G(I,K)=-G(I,K)
      ENDDO
      ENDIF
100  CONTINUE
      RETURN
      END

```

```

SUBROUTINE COMB1(MESH_FILE,TRISIGN,NNOD,TAB3,EDGSGN)
C      THE CODE TO COMBINE THE MATRICES FROM FEM SUBROUTINE AND BI SUBROUTINE
C      SINCE THE TWO METHODS HAVE DIFFERENT NUMBERING SYSTEM, THIS CODE RELIES
C      ON THE INFORMATION PROVIDED BY PREP1.FTN AND PREP2.FTN TO CONVERT THE
C      SIGN CONVENTION OF THE FEM FOR THE ON-DIELECT-SURFACE EDGES TO MATCH
C      THE BI SIGN CONVENTION.
      PARAMETER (MAXEDGE=2000,MAXNODE=500,MAXTRI=650,MAXZ=2000)
      INTEGER P0,P1,P2,P3,EG0,EG1,EG2,EG3,EDGSGN(MAXZ),TAB2(MAXNODE)
      INTEGER TRISIGN(MAXTRI,3),EDGNODS(MAXEDGE,2),TAB3(MAXEDGE)
      INTEGER TRIEDGS(MAXTRI,3)
      REAL NODCRDS(MAXNODE,3)
      CHARACTER*40 LINE
      CHARACTER*40 MESH_FILE

      OPEN (UNIT=4,FILE=MESH_FILE)
100  CONTINUE
      READ(4,'(A)',END=1000) LINE
      IF (LINE(2:6).EQ.'FLAG1') THEN
      READ(4,'(A)') LINE
      I=1
200  READ(4,*) NODE,NODCRDS(I,1),NODCRDS(I,2),NODCRDS(I,3)
      IF (NODE.EQ.-1) THEN
      GOTO 300
      END IF
      I=I+1
      GOTO 200
300  CONTINUE

```

```

ELSE IF (LINE(2:6).EQ.'FLAG2') THEN
  READ(4,'(A)') LINE
  READ(4,*) NINTEDG
  READ(4,'(A)') LINE
  READ(4,*) NEXTEDG
  READ(4,'(A)') LINE
  NEDG=NINTEDG+NEXTEDG
  DO 400 J=1,NEDG
    READ(4,*) NDUM,EDGNODS(J,1),EDGNODS(J,2)
400  CONTINUE

ELSE IF (LINE(2:6).EQ.'FLAG3') THEN
  READ(4,'(A)') LINE
  READ(4,*) NTRIS
  READ(4,'(A)') LINE
  DO 500 K=1,NTRIS
    READ(4,*) NDUM,TRIEDGS(K,1),TRIEDGS(K,2),TRIEDGS(K,3)
500  CONTINUE

END IF
GOTO 100
.000 CONTINUE
CLOSE(4)

OPEN(4,FILE='TAB2')
DO I=1,NNOD
  READ(4,*) NDUM,TAB2(I)
ENDDO
OPEN(4,FILE='TAB3')
DO I=1,NEDG
  READ(4,*) NDUM,TAB3(I)
ENDDO

DO 1500 I=1,NTRIS
  EG1=TRIEDGS(I,1)
  EG2=TRIEDGS(I,2)
  EG3=TRIEDGS(I,3)
  DY1=ABS(NODCRDS(EDGNODS(EG1,2),2)-NODCRDS(EDGNODS(EG1,1),2))
  DY2=ABS(NODCRDS(EDGNODS(EG2,2),2)-NODCRDS(EDGNODS(EG2,1),2))
  DY3=ABS(NODCRDS(EDGNODS(EG3,2),2)-NODCRDS(EDGNODS(EG3,1),2))
  FIND THE MINIMUM DY'S AND MARK THE CORRESPONDING EDGES
  IF(DY1.LE.DY2) THEN
    IF(DY1.LE.DY3) THEN
      MK=1
    ELSEIF(DY1.GT.DY3) THEN
      MK=3
    ENDIF
  ELSE
    IF(DY2.LE.DY3) THEN
      MK=2
    ELSEIF(DY2.GT.DY3) THEN
      MK=3
    ENDIF
  ENDIF
  FIND TWO NODE #'S (P2,P3) CORRESPONDING TO THE EDGE (EG1) WITH MINIMUM 'DY'
  IF(MK.EQ.1) THEN
    P2=EDGNODS(EG1,1)
    P3=EDGNODS(EG1,2)
  ENDIF
  IF(MK.EQ.2) THEN
    P2=EDGNODS(EG2,1)
    P3=EDGNODS(EG2,2)
  ENDIF
  EG0=EG1
  EG1=EG2
  EG2=EG0

```

```

ENDIF
IF (MK.EQ.3) THEN
P2=EDGNODS (EG3,1)
P3=EDGNODS (EG3,2)
EG0=EG1
EG1=EG3
EG3=EG0
ENDIF
C FIND THE 3RD NODE #
P0=EDGNODS (EG2,1)
IF (P0.NE.P2.AND.P0.NE.P3) THEN
P1=P0
ELSE
P1=EDGNODS (EG2,2)
ENDIF
C ORDER P1,P2,P3 COUNTERCLOCKWISELY
Y1=NODCRDS (P1,2)
Y2=NODCRDS (P2,2)
Y3=NODCRDS (P3,2)
X2=NODCRDS (P2,1)
X3=NODCRDS (P3,1)
YY=AMAX1 (Y2,Y3)
IF (Y1.GT.YY) THEN
C NUP=1
IF (X3.GT.X2) THEN
GOTO 1110
ELSE
P0=P2
P2=P3
P3=P0
ENDIF
ELSE
C NUP=2
IF (X3.LT.X2) THEN
GOTO 1110
ELSE
P0=P2
P2=P3
P3=P0
ENDIF
ENDIF
1110 CONTINUE
C FIND EG2: (P1,P3); EG3: (P1,P2)
N21=EDGNODS (EG2,1)
N22=EDGNODS (EG2,2)
IF (P2.NE.N21.AND.P2.NE.N22) THEN
GOTO 1120
ELSE
EG0=EG2
EG2=EG3
EG3=EG0
ENDIF
1120 CONTINUE
C COMPARE THE SIGNS (ACCORDING TO TRISIGN FROM MOM),
C BY CHECKING THE EDGES IN EACH TRIANGLES ONE BY ONE
DO II=1,3
IF (EG1.EQ.TRIEDGS (I,II)) NE1=II
IF (EG2.EQ.TRIEDGS (I,II)) NE2=II
IF (EG3.EQ.TRIEDGS (I,II)) NE3=II
ENDDO
IF (TRISIGN (I,NE1).NE.0) THEN
IF (TRISIGN (I,NE1).EQ.1) THEN
IF (TAB2 (P2).LT.TAB2 (P3)) THEN
EDGSGN (TRIEDGS (I,NE1))=1
ELSE

```

```

        EDGSGN (TRIEDGS (I, NE1) )=-1
    ENDIF
ELSE
    IF (TAB2 (P2) .GT. TAB2 (P3) ) THEN
        EDGSGN (TRIEDGS (I, NE1) )=1
    ELSE
        EDGSGN (TRIEDGS (I, NE1) )=-1
    ENDIF
ENDIF
ENDIF
IF (TRISIGN (I, NE2) .NE. 0) THEN
    IF (TRISIGN (I, NE2) .EQ. 1) THEN
        IF (TAB2 (P3) .LT. TAB2 (P1) ) THEN
            EDGSGN (TRIEDGS (I, NE2) )=1
        ELSE
            EDGSGN (TRIEDGS (I, NE2) )=-1
        ENDIF
    ELSE
        IF (TAB2 (P3) .GT. TAB2 (P1) ) THEN
            EDGSGN (TRIEDGS (I, NE2) )=1
        ELSE
            EDGSGN (TRIEDGS (I, NE2) )=-1
        ENDIF
    ENDIF
ENDIF
ENDIF
IF (TRISIGN (I, NE3) .NE. 0) THEN
    IF (TRISIGN (I, NE3) .EQ. 1) THEN
        IF (TAB2 (P1) .LT. TAB2 (P2) ) THEN
            EDGSGN (TRIEDGS (I, NE3) )=1
        ELSE
            EDGSGN (TRIEDGS (I, NE3) )=-1
        ENDIF
    ELSE
        IF (TAB2 (P1) .GT. TAB2 (P2) ) THEN
            EDGSGN (TRIEDGS (I, NE3) )=1
        ELSE
            EDGSGN (TRIEDGS (I, NE3) )=-1
        ENDIF
    ENDIF
ENDIF
1500 CONTINUE
CLOSE (4)
END

```

```

SUBROUTINE COMB2 (NINTEDG, EDST, TAB3, EDGSGN, Z, A)
GIVEN FEM: MATRIX A, NEW/OLD EDGE # CONTRAST TABLE EDST
GIVEN BI: MATRIX Z, NEW/OLD EDGE # CONTRAST TABLE TAB3
GIVEN SIGN OF EDGES (ON DIE-SURFACE) CONTRAST TABLE EDGSGN
PARAMETER (MAXEDGE=2000, MAXA=2000, MAXZ=2000)
INTEGER TAB3 (MAXEDGE), EDST (MAXA, 2), EDGSGN (MAXZ)
COMPLEX A (MAXA, MAXA), Z (MAXZ, MAXZ)
DO I=1, NINTEDG
M=TAB3 (I)
L=EDST (M, 1)
DO J=1, NINTEDG
N=TAB3 (J)
K=EDST (N, 1)
A (L, K) =A (L, K) *EDGSGN (J) +Z (I, J)
ENDDO
ENDDO
RETURN
END

```



THE UNIVERSITY OF MICHIGAN

DATE DUE

7/12/98 7:06 PM
