National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Far-field Predictions from Near-field Measurements using the Three Dimensional Finite Element and Boundary Integral Methods

by

Rédouane Laroussi, B.A.Sc.

A thesis
submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

ISBN 0-315-93593-6

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this document. I authorize the University of Ottawa to lend this document to other institutions or individuals for the purpose of scholarly research.

Rédouane Laroussi

I further authorize the University of Ottawa to reproduce this document by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Rédouane Laroussi

Abstract

Characterizing antenna designs in their operational modes has been a classic problem for electromagnetic engineers for many years. The far-field pattern is of great importance in antenna characterization, but in many cases, it is impossible or at least impractical to measure directly.

This thesis presents a numerical technique whereby near-field measurements can be transformed into far-field solutions. The finite element method(FEM), and an integral formulation, solution to the scattering problem, are used separately to solve the open boundary wave propagation problem in three dimensions. The near-field of a known source is sampled over a mesh of points and used as data for both methods. The integral formulation yields the far-field solution for homogeneous problems without the need for space discretization. The finite element method uses tetrahedral elements for discretizing the domain of interest and second order interpolation functions. It is complemented by the application of an absorbing boundary condition(ABC) which allows truncation of the domain. Classical ABCs of the local type are applied to the three dimensional problem. A new non-local ABC is obtained from the integral formulation and is coupled to the finite element approach. Computed field strengths obtained with the different approaches are compared with the analytical results obtained from the excitation source's analytical expression. The agreement was very good with maximum error percentages of about 10% for the FEM method and 2.5% for the integral formulation.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor G. Costache for his valuable advices and guidance and for his unfailing patience with me during the course of this research. I would also like to thank my professors and colleagues for the valuable discussions we had and for enduring my persistent questioning. Finally, I would like to acknowledge the patience and understanding of my family and friends during my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The problem of characterizing antenna designs in their operational modes has been a classical problem in electromagnetics for quite some years. Measurement of the far-field pattern directly in the far-field is in many cases impractical, if not impossible to obtain. It may be that the far-field distance is too great and unachievable, or that the antenna cannot be moved about easily.

In the scope of controlling electromagnetic interference (EMI), a quick and efficient way of estimating radiated emissions of electrical equipment is needed. To comply with radiation regulations for electromagnetic interference and compatibility (EMI/EMC), signal levels at different distances and frequencies have to be specified for all electronic equipment.

Many algorithms have been developed to obtain far-field information from near-field measurements. These have been generally based on simplified schemes that take advantage of problem symmetries, or make assumptions that limit the range of applicability of the results. The amount of work done to model the phenomenon of electromagnetic(EM) wave propagation in 3-D is very limited and has been possible today only with the advent of powerful computers.

The motivation of the work that follows was to provide a way of transforming near-field data to far-field information by exact emulation of the wave propagation in space without introducing any simplifications. The FEM is not generally considered a suitable technique for antenna or radiation problems because they are unbounded problems; however, the FEM has many positive benefits over other techniques. The problem of using the finite element method on open problems and the errors due

to domain truncation present an added difficulty, the handling of which, still needs much improvement. What is known in the literature as an absorbing boundary condition(ABC) has been used quite extensively as a remedy to this problem. The ABC simulates the far-field response at the boundary of the truncated mesh. Applications of ABCs in 3-D are relatively new and publications in scientific journals presenting quantitative results to demonstrate the efficiency or inefficiency of the implementation of ABCs in numerical analysis are hard to find.

1.2 Contribution

The results presented in this thesis offers two approaches to tackle the problem of far-field predictions from near-field measurements using the finite element method and an integral equation solution to the scattering problem.

A meshing program was developed to discretize the 3-D domain of the problem solved. The meshing program MESHMAIN creates a mesh made up of tetrahedral elements. It allows finite elements to have different electrical characteristics (permittivity/conductivity). Boundaries of the domain can be identified as any one of three boundary types used in EM modeling: Dirichlet, Neumann or open boundary. The meshing program can be executed independently of the electromagnetic solver, and the mesh created can be modified without the need for re-executing the mesher with the use of other auxiliary programs.

The electromagnetic solver, the program MAINCOMPSG, solves the wave equation using the FEM based on a Galerkin formulation for the three Cartesian components of the magnetic field. For the open boundary sections, there is a choice of three ABCs to be applied: i) the Sommerfeld ABC, ii) the first order Bayliss-Turkel ABC, and iii) an integral type ABC derived from the boundary integral method (BIM) and adapted to the FE approach. For a homogeneous domain, an integral equation solution to the EM propagation (BIM) is utilized to obtain the field values anywhere in space. The program BEMRAD uses the mesh created by MESHMAIN to solve the wave equation with a boundary integral. BIM results are faster to obtain and the program itself requires much less memory than MAINCOMPSG for a similar configuration. Results are given for the two methods, BIM and FEM with different ABCs, and comparisons are made between the numerical approaches and analytical values.

Chapter 2

Literature Review

2.1 Introduction

Numerical methods have become important tools in solving practical electromagnetic problems. Problems that can be handled analytically have simple geometries or use assumptions that impose restrictions on generality and in most cases do not represent real or practical problems. The main issue to using this wide range of numerical tools is choosing a good numerical algorithm. The selection criteria will be those that deal with feasibility and flexibility considerations. An algorithm that is inflexible and solves only one problem configuration may be too difficult to modify for different configurations, and thus has low productivity. Another consideration is that every numerical algorithm must make use of a computer. Machine requirements in terms of memory and execution time weigh heavily when deciding which method to use. Reaching a decision which takes into account all aspects of the problem is an impossible task because there is never a unique solution and there will always be room for new arguments and improvements as long as scientific and technologic advancements continue.

2.2 3-D EM Propagation and FEM

Electromagnetic field propagation problems can be classified into two groups: i) bounded and ii) unbounded problems. These two situations have the same approach around the source region and inhomogeneities; however for the unbounded situation, an extra complication arises from the need to deal with an infinite space. The methods of dealing with this situation will be discussed in detail in the next section.

The finite element method despite it's own shortcomings, remains the most appropriate method when the problem has discrete inhomogeneities and complex geometries. Integral equation methods are difficult to formulate for inhomogeneous problems, but are very easily adapted to solve unbounded problems. They can be used to solve large problems very quickly, but they lose their advantage over differential methods such as FEM when the inhomogeneities are not centralized. The Green functions involved become very complicated to find and integrate[1].

Several approaches have been developed specifically to solve 3-D EM problems with the finite element method. Many algorithms have been elaborated and tested on 2-D situations and parallels drawn to their use in 3-D. Webb[2] used a variational approach with FEM to solve a vector equation for either the magnetic field $\vec{H}$ or electric field $\vec{E}$, and derived the scattering parameters from the field values in waveguide structures. The results obtained were very good, but at the expense of sophisticated trial functions. Some of the problems solved were inherently 2-D problems. Webb did not mention any difficulties with spurious modes.

The occurrence of non-physical solutions (spurious modes) arises very frequently in the solution of 3-D EM problems[3, 4, 5]. Many approaches have been taken to eliminate this shortcoming of the FEM. In another paper[4], Webb promoted the use of a penalty factor associated with the divergence of the field quantity to be added to the wave equation in order to eliminate spurious modes based on the assumption that spurious modes are not divergence-free. Among the advantages mentioned for the penalty factor, was the preservation of the sparsity of the matrix system produced with FEM. The selection of the factor itself, however, was not systemized. In his paper, Webb implemented a test where the divergence is repeatedly checked until satisfactory levels of error were reached.

In [6], Konrad substituted the system of equations obtained from the divergence directly into the system of equations derived from the wave equation, and thus reduced its order. The final system is smaller, and quicker to solve, but the substitution itself is not straightforward[4]. In [3], Svedin enforced explicitly the boundary conditions between adjacent elements. The approach was tested on several situations including anisotropic media with magnetic and dielectric losses. The results obtained compare very well with other methods and did not present any spurious behaviour.

MacNeal[5] replaced the polynomial shape functions used in the classical FEM approach by modal basis functions. The modal functions were the eigenfunctions of

the associated eigenvalue problem. The field was rewritten as an expansion of modal functions and the modal amplitudes were solved for. The idea was that if the basis functions used were divergence free, the solution would be divergence free.

In [7], G.Mur was more concerned with the continuity of the tangential components and normal flux of the electromagnetic quantities than with spurious modes. The work relied on basis functions that provide linear interpolation on the whole finite element(FE) and not only in the interior, as was the case with Nedelec[8] in his work on continuity of the same quantities between finite elements. Mur reported excellent results for 3-D situations but did not mention any spurious mode problems.

The most complete work done on the subject of spurious modes has been conducted by Lynch and Paulsen[9, 10]. Lynch and Paulsen studied the dispersion relation for four versions of the Maxwell equation:

- The double curl equation: $\nabla \times \nabla \times \vec{E} - k^2 \vec{E} = 0$
- The Helmholtz equation: $\nabla^2 \vec{E} + k^2 \vec{E} = 0$
- The penalty equation: $\nabla \times \nabla \times \vec{E} - p\nabla(\nabla \cdot \vec{E}) - k^2 \vec{E} = 0$
- The primitive coupled Maxwell equations: $\nabla \times \vec{H} = -j\omega\epsilon\vec{E}$; $\nabla \times \vec{E} = j\omega\mu\vec{H}$

The findings were that the double curl, the penalty equation and the primitive coupled equations supported spurious modes. Some remedies are possible, but not always and not for all three approaches. The Helmholtz scheme supports physical modes that are divergence free. The enforcement of divergence free boundary conditions eliminates the potential for spurious modes.

This quick overview of what has been done with FE in 3-D illustrates some of the shortcomings that the finite element method has. There is a consensus, however, that it is one of the most promising methods in terms of versatility and generality for covering a wide range of problems. The ability of the finite element method to model any geometry or electrical property distribution, currently, gives it an edge over some of the other popular numerical techniques.

2.3 3-D Space Discretization

A finite element mesh lends itself very efficiently to modelling of all kinds of inhomogeneities in electrical characteristics and geometries. The cost is a huge amount

of mesh data that has to be stored and manipulated. The result is slower execution time and the need for large storage capabilities. This can be remedied by using a coarse mesh.

The finite element mesh, if any value is to be given to results, has to satisfy two requirements. The distance between two nodes of the finite element mesh has to be at most smaller than $\lambda/2$. λ is the wavelength of the frequency of investigation. Although the $\lambda/2$ limit stems from Nyquist's sampling theorem and theoretically should work; good results with reasonable accuracy require finer meshes in the order of $\lambda/5$[11]. The second requirement is that the finite element mesh has to be fine enough so as to describe the smallest electrical or geometrial features in the structure. An adaptive algorithm which could create a very fine mesh around the detailed features and a coarse mesh in the other areas would optimize the number of finite elements needed for a problem.

In [12], J.D. Zhou proposed to use a function over the solution domain that returns a value proportional to the size of the finite element to be created. The same author in [13], proposed, after first creating a coarse mesh, to go through all the finite elements and subdivide them repeatedly until the mesh is fine enough. The disadvantage of such an approach is that the details of the structure are lost in the first triangularization given that the smaller finite elements are confined in the larger ones.

These schemes for mesh density are easily applied to 2-D as well as 3-D problems. The actual triangularization in 2-D and 3-D is not so straightforward. For the 2-D case, any improvised algorithm will work when enough overlapping tests are included. In [14] Nelson gives an algorithm that will triangularize any planar surface without the need to check for overlapping of finite elements during meshing. It is based on the selection of a third point to be associated with another two, such that the three points are the ones best spread on the circumference of a circle. This algorithm could not be easily extended to 3-D meshing.

Most commercially available packages that solve 3-D problems are still limited to structures with axial symmetries, or ones that use a uniform mesh. The approach of discretizing with symmetrical finite elements such as cubes which fit together nicely is easy, but does not provide much flexibility in following curved surfaces. The natural extension of the 2-D triangular finite element is the four faced tetrahedron, which allows an accurate modelling of the structure features and provides the ability to use

simplex coordinates(to be explained), thus simplifying the bases functions and the evaluation of volume and surface integrals[15].

2.4 Open Boundary Problem and the Absorbing Boundary Condition (ABC)

2.4.1 Introduction

The finite element method cannot be used to solve open boundary problems, unless a means of simulating the condition at infinity is achieved. The simple approach is to place the outer boundary very far away and mesh the interior. However, the demand on computer storage will be too great if a large domain is discretized, especially for a 3-D problem. With most problems, the area of interest is limited to a small region and solutions for the surrounding regions is not necessary. One approach has been to terminate the mesh before the fields go to zero and impose an absorbing boundary condition to represent the response of an infinite space. Another approach has been to combine the finite element method with an integral method to take care of the infinite space. This has resulted in methods called hybrid methods.

2.4.2 Local Type ABC

Local type absorbing boundary conditions are the ones that impose a value for the variation of the field quantity, in the direction normal to the outer boundary, evaluated from field values at the position of interest. These types are relatively simple to derive and implement, however, they have some limitations with regards to accuracy and general purpose usage.

Most of these ABCs are derived from the expression given by the expansion theorem for electromagnetic fields found in [16]. The expression is given here for convenience:

$$\vec{A}(r) = \frac{e^{jkr}}{r} \sum_{n=0}^{\infty} \frac{\vec{A}_n(\theta,\phi)}{r^n}, \tag{2.1}$$

where $\vec{A}$ is a vector radiating function, and the expression is valid in a region $r > constant$. Based on this equation, many authors have arrived at absorbing boundary conditions of varying accuracy and complexity using derivations and algebraic manipulations.

The most basic ABC is the one known as Sommerfeld's[17] given by:

$$\lim_{r\to\infty} \frac{\partial \vec{A}}{\partial r} = -jk\vec{A}, \tag{2.2}$$

where r is the direction of propagation.

This boundary condition is valid only when imposed very far from the sources, and hence defeats the purpose of early truncation of the domain to reduce the mesh size. J.J. Stoker[18] states that unique solutions of radiation problems cannot be determined by Sommerfeld type ABCs. The Sommerfeld condition is obtained by dropping all, except the first term of the derivative of Eq. (2.1) with respect to r. Bayliss and Turkel[19] derived a higher order ABC by using Eq. (2.1) and the wave equation. For one component u of $\vec{A}$, they obtained the expressions:

$$\frac{\partial u}{\partial r} = \alpha_2(r)u + \beta_2(r)\frac{\partial^2 u}{\partial \phi^2} \quad \text{in 2 dimensions,} \tag{2.3}$$

and

$$\frac{\partial u}{\partial r} = -jk[\alpha_3(r)u + \beta_3(r)Du] \quad \text{in 3 dimensions,} \tag{2.4}$$

where $\alpha_2, \alpha_3, \beta_2$ and β_3 are polynomial functions of r, and D is the operator defined as:

$$Df = \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial f}{\partial\theta}) + \frac{1}{\sin^2\theta}\frac{\partial^2 f}{\partial\phi^2}. \tag{2.5}$$

There are three sources of error in expressions of the type of Eqs. (2.2), (2.3) and (2.4): i) the truncation of the series, ii) the direction of the normal to the outer boundary is not the direction of the coordinate r[20], and iii) the difference between r and $r - r_0$ is not negligible, r_0 being the position of the source. In the far-field, all sources behave as source points and furthermore, when the source is centered about the origin of the coordinate system, $r - r_0$ tends towards r. The closer the observation point is to the source, the more innacurate this assumption becomes. In case (ii), many authors have used circular or spherical outer boundaries to be able to replace $\frac{\partial u}{\partial n}$ by $\frac{\partial u}{\partial r}$ where n is the variable in the direction of the normal, and r is the radial component. This assumption's validity depends on how close to a circular shape is the outer boundary.

Many attempts have been made to minimize the effects of these sources of error. In [21], Khebir, Kouki and Mittra derived an ABC from a modified version of Eq. (2.3). They used the progression $n = 0, kr/2, kr, ...$ in Eq.(2.1), instead of $n = 1, 2, 3...$, to obtain a higher order ABC equivalent to using more terms in the series. To simplify

the derivation, the artificial boundary was chosen such that the normal coincided with the spherical coordinate r. Some researchers have attempted to apply these differential ABCs to general boundaries without any clear advantages[22] over spherical boundaries.

An ABC that can be classified in either category, local or non-local, is the ABC developed by Ramahi[23]. Ramahi represented the field by a summation of Hankel functions in 2-D as:

$$u(r,\phi) = \sum a_n H_n^{(2)}(kr) e^{jn\phi}, \tag{2.6}$$

and a normal derivative of the field on the outer boundary written as a weighted combination of field values lying in the interior of the problem as:

$$\frac{\partial u}{\partial n}\Big|_{r_0} = c_0 u_0 + c_1 u_1 + c_2 u_2 + \ldots \tag{2.7}$$

where: 0 refers to the position where the derivative is to be evaluated,
and $0, 1, 2$ refer to the positions of the nodes chosen for the evaluation.

Equation (2.7) was rewritten using (2.6) and enforced at as many harmonics as there were c coefficients in (2.7). This ABC could thus evaluate the normal variation with values of the field that are distributed all around the boundary and thus give a non-local estimate. The major drawback was the selection of the harmonics since there was no a priori knowledge of the dominant ones.

2.4.3 Non-Local Type ABC

The non-local ABC is based on Eq. (2.6). In [24], Keller and Givoli computed the coefficients a_n with a Fourier integral. Eq. (2.6) is then used to estimate the field everywhere outside an outer boundary of radius r on which the Fourier analysis was performed. A circular boundary was chosen to simplify the integration with respect to r. The derivative of the final expression was taken to obtain $\frac{\partial u}{\partial n}$. Keller and Givoli obtained some promising results in 2-D, however, a potential source of inaccuracies is that the field varies in space and the Fourier representation is based only on the surface variation of the field. Only surface field values are used to compute the Fourier coefficients and the variation of the field with respect to the normal is not considered.

2.4.4 Hybrid Methods

Hybrid methods with the FEM are often used to solve open boundary problems. Although, they cannot be called non-local ABC, they are closely related to them. The methods hybridized with FEM can be used to solve the homogeneous portion of the problem and at the same time provide an expression to terminate the FEM mesh. The finite element method has been coupled with several other numerical techniques: the multiple multipole method, the moment method and the boundary integral method. Each of these combinations, were an attempt to overcome the weakness of the FEM in tackling unbounded spaces.

Sroka, in [25], was the first to try to couple the generalized multiple multipole technique with the finite element method. The major findings were that the outer boundary did not need to be as far as when other techniques were used. The hybrid worked very effectively but more work needs to be done to improve the time efficiency of the algorithm.

The FE method has also been combined with the moment method. Xingchao[26] used the FEM to solve the interior region of a problem, the part with all the inhomogeneities, and the moment method for the exterior part. The two parts were coupled together using the equivalence principal and the continuity of the tangential components of $\vec{E}$ and $\vec{H}$.

Another hybrid method where extensive work has been done is the FEM/BEM method[1, 27], a combination of finite element and boundary element methods. B.H. McDonald in [1] used the FEM to solve the problem in the bounded region including all inhomogenities and applied an integral constraint on the outer boundary derived from the application of Green's theorem on the boundary values. Paulsen[27] used a similar approach to solve an electromagnetic propagation problem in 3-D.

When all the inhomogeneities are enclosed in a surface on which the field values are known, the boundary integral methods can be used alone to solve the propagation problem. The formulation provided in [1, 10] can be taken up without the FEM part. The only problem is to find the appropriate Green's function. In [28], Poggio explains the derivation of the integral equation solution to the wave equation.

2.5 The Problem of Predicting Far-Field Patterns from Near-Field Measurements

Much research has been done in the attempt to solve the problem of determining far-field radiation levels and patterns from near-field information, for EMI/EMC compliance tests and antenna radiation pattern measurements. In both these cases, it is sometimes impossible or impractical to provide an open range, or obtain the far-field measurements directly in the far-field, either due to the electrical size of the apparatus or its physical size. One way to bypass this problem has been to collect near-field measurements and transform them to the far-field.

In [29], the field was expanded into a series expression and the coefficients are calculated via a Fourier analysis of the measured near-field data. The results obtained in 2-D situations compared successfully with measured values. In [30], Johnson summarized the three main techniques for obtaining far-field information. The techniques are: i) the transformation of near-field measurements, ii) making measurements on the test antenna irradiated by a plane wave, or iii) focusing the antenna and assuming that the near-field measurements are the same when the antenna is refocused to infinity. Johnson also gives guidelines to conduct measurements.

R.J. Luebbers [31] developed an algorithm to obtain far-field information from a transformation of near-field measurements using the FDTD method. T.K. Sarkar[32] used an algorithm where equivalent currents on the measurements surface are derived from the field values and plugged back into an integral expression to have the solution in the far-field. The disadvantage of this approach is that a matrix equation system has to be solved. In [33], Bucci elaborated on the sampling of the field. He proposed an optimal sampling algorithm where the effect of parameters was studied. Bucci looked at optimal sampling densities and optimal measurement surfaces in terms of shapes and positions in order to improve and simplify the near-field to far-field transformation. He used the Fourier analysis method to transform the measurements to the far-field.

Chapter 3

Mathematical Formulation of the Problem

3.1 The Model

The problem to be solved is of interest to the EMI/EMC community and has extensions into antenna testing applications. The solution is a transformation of field values measured in the vicinity of an electromagnetic source to corresponding far-field values.

In practical situations, the source would be an antenna under test or a piece of electrical equipment with parasitic radiations to be characterized. The apparatus is located above the ground and radiates into open space. This arrangement does not imply any a priori symmetries except that of the ground plane. The source of EMI is represented by an arbitrary distribution of point sources that provide a general and asymmetric field variation.

The magnetic field $\vec{H}$ in spherical coordinates created by a point source with an elementary current directed along the Cartesian $\hat{z}$ component is [33]:

$$\vec{H}(r,\theta,\phi) = \frac{I\Delta z}{4\pi}[\frac{j\beta}{r} + \frac{1}{r^2}]e^{-j\beta r}\sin\theta\,\hat{\phi}, \tag{3.1}$$

where: r, θ and ϕ are the coordinates of the observation point when the point source is located at the origin of the coordinate system, β is the phase velocity in free space, and $\Delta z << \lambda$ is the length of the filament of current amplitude I.

The contributions to the $\vec{H}$ field in Cartesian coordinates for current filaments directed in each of the three directions are given in Table 3.1 with A defined as

$$A = -\frac{I\Delta z e^{-j\beta r}}{4\pi r}[j\beta + \frac{1}{r}]. \tag{3.2}$$

	$\hat{x}$ directed current	$\hat{y}$ directed current	$\hat{z}$ directed current
H_x	0	$-A\cos\theta$	$A\sin\theta\sin\phi$
H_y	$A\cos\theta$	0	$-A\sin\theta\cos\phi$
H_z	$A\sin\theta\cos\phi$	$A\sin\theta\cos\phi$	0

Table 3.1: Contribution to the field H_i from the current filament I_i

The total contribution to the field $\vec{H}_x$ can be written as follows:

$$H_x(r,\theta,\phi) = \sum_{i=0}^{n_x} A(r,r_{oi})0 - \sum_{i=0}^{n_y} A(r,r_{oi})\cos\theta' + \sum_{i=0}^{n_z} A(r,r_{oi})\sin\theta'\sin\phi', \qquad (3.3)$$

where n_x, n_y, n_z are the numbers of point sources directed respectively along $\hat{x}$, $\hat{y}$ and $\hat{z}$ axis. r_{oi} is the position of the i^{th} point source. θ' and ϕ' are functions of θ and θ_{oi}, and ϕ and ϕ_{oi}.

Similar, expressions for H_y and H_z can be written out. The point sources positioned in a cluster above ground are sketched in Fig.3.1. For clarity the sources are regrouped according to their orientation. The point source has been chosen because it has an analytical solution valid for both the near and far-field regions. It allows the construction of an EMI source of any desired degree of variation and the analytical solution can be used as a means for validation of the numerical algorithms. The total source field is obtained by superposition of the analytical solution of the individual point sources. Figure 3.2 identifies the different regions and limiting boundaries associated with Fig.3.1

The surface S in Fig.3.2 enclosing the solution domain V is made up of three segments: S_1, S_2, and S_3.

$$S = S_1 \bigcup S_2 \bigcup S_3. \qquad (3.4)$$

S_1 is the portion of S which encloses all sources in Fig.3.1 and is the surface on which measurements of the field values are made. In the model, these values are obtained by sampling Eq.(3.3) on S_1 for H_x, H_y and H_z. S_2 lies on the ground. In the model the ground is represented by an ideal ground plane and in the mathematical formulation to follow will be accounted for by using image theory. The last segment, S_3, has no physical meaning but is necessary to limit the solution domain.

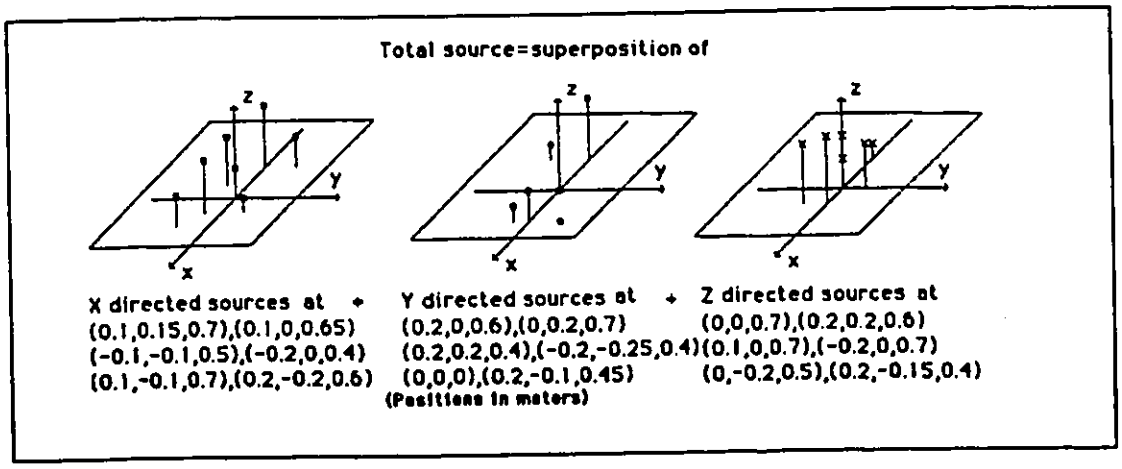

Figure 3.1: Source distribution above ground plane

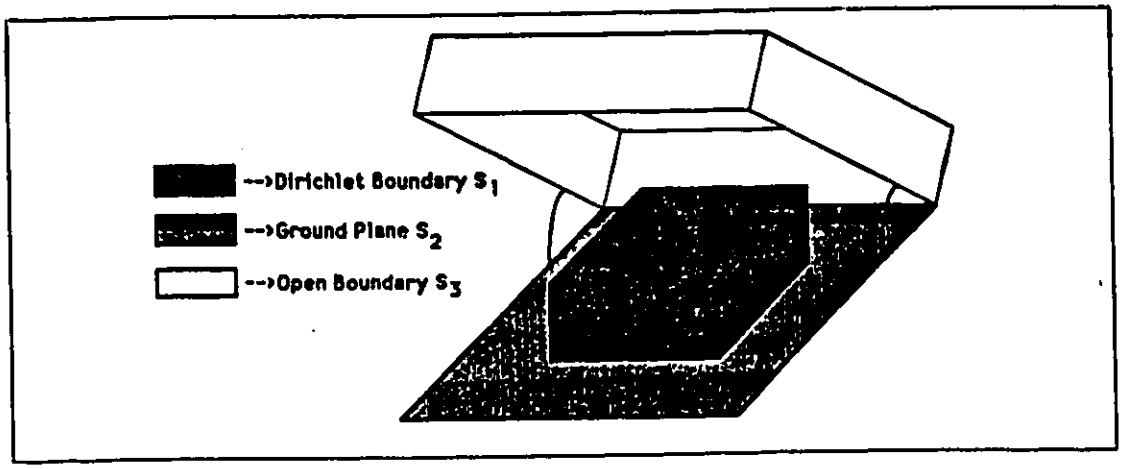

Figure 3.2: Domain and boundaries of the problem

3.2 FEM formulation

3.2.1 The Equations to Solve

In the volume V, the wave propagates in a homogeneous space. In the frequency domain, the Helmholtz wave equation describes the variation of the field components. For the magnetic field, the equation for the phasor $\underline{\vec{H}}$, is given by

$$\nabla^2 \underline{\vec{H}} + \omega^2 \mu_0 \epsilon_0 \underline{\vec{H}} = 0 \text{ in } V, \tag{3.5}$$

where ω is the phase velocity defined as $2*\pi*$frequency and μ_0 and ϵ_0 are respectively, the magnetic permeability and dielectric constants of free space. Equation (3.5) is arrived at by assuming a time variation as $e^{j\omega t}$.

For the special case of Cartesian coordinates, the Laplacian of the magnetic field(or any vector field) can be written as:

$$\nabla^2 \vec{H} = \hat{x}\nabla^2 H_x + \hat{y}\nabla^2 H_y + \hat{z}\nabla^2 H_z, \tag{3.6}$$

where the underlining bar representing phasors has been omitted to simplify notation. Since there are no inhomogeneities in V, Eq.(3.6) permits the uncoupling of the three Cartesian components of the magnetic field and permits the rewritting of Eq.(3.5) as three independent scalar equations:

$$\nabla^2 H_i + \omega^2 \mu_0 \epsilon_0 H_i = 0 \text{ in } V, \quad i = \hat{x}, \hat{y} \text{ or } \hat{z}. \tag{3.7}$$

This will allow solving problems three times larger than if Eq.(3.5) was used directly. However, it will limit the generality of the solution since the field components are decoupled and the domain must remain homogeneous. For more generality, one would have to replace the Laplacian by the double-curl,

$$\nabla^2 \vec{H} = -\nabla \times \nabla \times \vec{H}, \textit{simplified, here, because of the divergence free magnetic field.} \tag{3.8}$$

The second approach, however, is not spurious-mode free because of the cross-derivatives that arise in the double-curl[9].

A differential equation problem is not properly posed until all initial values are all specified and all boundary conditions are fixed. For the model in Fig.3.2, boundary conditions on S_1, S_2 and S_3 have to be specified to obtain the exact solution and unique solution of the problem at hand.

S_1 is the surface on which measurements will be made for the magnetic field components H_x, H_y and H_z. It will thus be a Dirichlet type boundary. S_3 is open with no constraints. To fully define the problem and obtain a unique solution a condition derived from an a priori knowledge of the solution and/or the governing equations of the problem must be applied at S_3. This condition describes the variation of the field in the direction normal to the surface S_3. Mathematically, it is $\frac{\partial H}{\partial n} = f$ where n is the normal to S_3. This expression known as an absorbing boundary condition is discussed in the literature review and the forms implemented in this work are derived in Section 3.4, namely, the Sommerfeld ABC, the Bayliss-Turkel ABC and a integral type ABC. S_2 is on the ground plane; it is assumed to be a perfectly conducting surface. If the ground plane is in the $x-y$ plane, then

$$H_z = 0 \quad \text{since } H_z = \frac{j}{\omega\mu_0}[\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}], \tag{3.9}$$

and E_y, E_x, being the tangential components of the electric field on a metallic plate, are constant on the $x-y$ plane and identically zero.
From

$$\vec{E} = -\frac{j}{\omega\epsilon}\nabla \times \vec{H} = (-\frac{j}{\omega\epsilon})[-\frac{\partial H_y}{\partial z}\hat{x} + \frac{\partial H_x}{\partial z}\hat{y} + (\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y})\hat{z}], \tag{3.10}$$

simplified using the fact that $H_z = 0$, it is deduced that

$$\frac{\partial H_y}{\partial z} = 0 \quad \text{and} \quad \frac{\partial H_x}{\partial z} = 0, \tag{3.11}$$

which for a ground in the x-y plane becomes:

$$\frac{\partial H_y}{\partial n} = 0 \quad \text{and} \quad \frac{\partial H_x}{\partial n} = 0 \tag{3.12}$$

given that the normal $\hat{n} = \hat{z}$. A condition such as the one given in Eq.(3.12) is known as a homogeneous Neumann boundary condition. In table format, the summary of the boundary condition for the whole boundary S is shown in Table 3.2.1.

3.2.2 The Weighted Residual Method

Solving the problem using the Finite Element Method consists of the following: solving Eq.(3.7) on small patches, called finite elements(FE), which when combined form the domain V; and, satisfying the boundary conditions given in Table 3.2.1. This scheme effectively breaks down the problem in a multitude of smaller problems to be

	S_1 Measuring surface	S_2 Ground plane	S_3 Open boundary
H_x	Dirichlet	$\frac{\partial H_x}{\partial n} = 0$	ABC
H_y	Dirichlet	$\frac{\partial H_y}{\partial n} = 0$	ABC
H_z	Dirichlet	$H_z = 0$	ABC

Table 3.2: Boundary conditions for the problem

solved simultaneously and simplifies the search for a solution by reducing the requirements to be met on the whole of V to requirements on the small FE patches. It is easier to find an expression, called interpolation or trial function, that fits the exact solution on a smaller domain, than on a larger one. The fitting process translates itself via an integral formulation into a set of linear algebraic equations for every patch. The variables solved for are parameters in the trial functions that make these functions a best fit. The method through which one can obtain the system of algebraic equations is one of two choices. The variational approach or the weighted residual approach. The variational approach is based on minimizing a functional; an integral expression derived from the governing equations of the problems and its boundary conditions. In some cases, this process turns out to be a minimization of the energy in the system in accordance with the state of minimum energy principle of physics. The disadvantage is that the functional has to be derived and varies from one problem to the other. The second approach is the weighted residual method. Here, the integral over the domain of the weighted error expression defined as the difference between the right hand side of the governing equation and the left hand side when the approximate solution is used, is forced to zero. This procedure usually will yield the same solution as the variational approach.

As it applies to the problem described in Section 3.1; if only one of the components is considered, say H_x, the problem is formulated as:

$$\nabla^2 H_x + k^2 H_x = 0 \quad \text{in } V, \tag{3.13}$$

$$\begin{aligned} H_x &= g \quad \text{on } S_1, \\ \frac{\partial H_x}{\partial n} &= 0 \quad \text{on } S_2, \\ \frac{\partial H_x}{\partial n} &= ABC \quad \text{on } S_3. \end{aligned} \tag{3.14}$$

The function g is the function describing the values of the field H_x on S_1 and $k^2 = \omega^2\mu_0\epsilon_0$.

Assume an approximate solution over a FE i as $\tilde{H}_{xi}$; the expression for the error E is then

$$\nabla^2\tilde{H}_{xi} + k^2\tilde{H}_{xi} = E \quad \text{in} \quad V_i. \tag{3.15}$$

For a weighting function W, the weighted integral becomes

$$\int_{V_i} W(\nabla^2\tilde{H}_{xi} + k^2\tilde{H}_{xi})\, dV_i = \int_{V_i} W\, E\, dV_i \tag{3.16}$$

where V_i is the domain of the i^{th} FE. When $\tilde{H}_{xi}$ tends towards the solution, E tends to zero.

Equation (3.16) is implemented for all the finite elements that form the domain of the solution. The patch work of $\tilde{H}_{xi}$ solutions will make up the global approximate problem solution $\tilde{H}_x$.

In the case of open problems, the domain V is infinite. To limit V to the region of interest and explicitely include the boundary condition defining the problem in Eq.(3.16), Green's first theorem is used. Green's first theorem will also reduce the requirements on the continuity of the derivatives of the patch solutions $\tilde{H}_{xi}$. As it is now, Eq.(3.16) contains a Laplacian term which involves second degree derivatives. To have well defined integrals of the Laplacian, $\tilde{H}_{xi}$ has to have a continuous first derivative; if the Laplacian is reduced to a gradient, $\tilde{H}_{xi}$ need only be continuous. Green's first theorem is as follows[35]:

$$\int_V (\nabla u \cdot \nabla v) dV = \oint_S u\frac{\partial v}{\partial n}\, dS - \int_V u\nabla^2 v\, dV \tag{3.17}$$

where: V is the domain,
S is the boundary enclosing V,
u and v are two continuous functions with continuous second derivatives.

When Eq.(3.17) is substituted in Eq.(3.16) with u replaced by W and v by $\tilde{H}_{xi}$, the weighted residual integral becomes

$$\int_{V_i} (\nabla W_i \cdot \nabla\tilde{H}_{xi})\, dV_i \; - k^2 \int_{V_i} W\tilde{H}_{xi}\, dV_i \; - \int_{S_i} W_i\frac{\partial\tilde{H}_{xi}}{\partial n}\, dS_i = 0 \tag{3.18}$$

where: V_i is the domain of a FE,
S_i is the boundary enclosing V_i,
and E, the error, has been set to zero.

3.2.3 Tetrahedral Element and Interpolation Functions

The tetrahedral element in Fig.3.3 has been chosen to subdivide the domain V into finite subvolumes V_i. This choice, as will be seen in the following developement, is the best one when it comes to representing irregular geometries. The mathematical properties the tetrahedral shape posseses, allow a high degree of automation in performing derivations and integrals analytically.

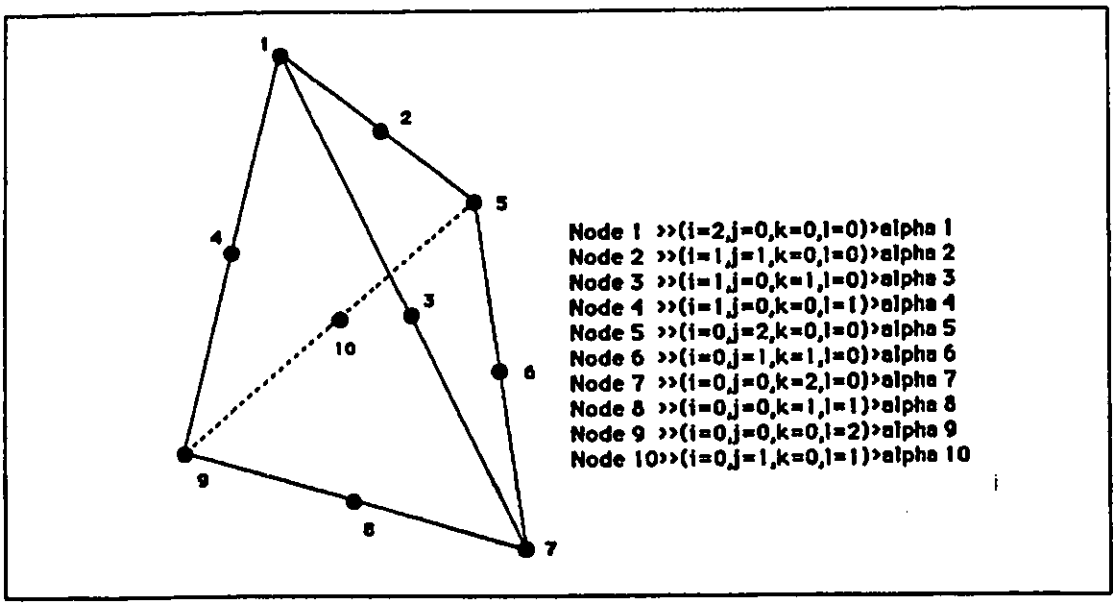

Figure 3.3: The tetrahedral element and its node indexing

The Simplex and Simplex Coordinates

A simplex S in N-dimensional space is the minimal non-trivial geometric figure in that space. In 2-D, it's a triangle defined by 2+1 vertices. In 3-D, it's a tetrahedron defined by 3+1 vertices.

When a point P is positioned inside an N space simplex, it divides it into $N+1$ subsimplexes each of which has P as a vertex and the remaining N chosen from the $N+1$ vertices of the original simplex. See Figure 3.4. The ratio of the size of the $N+1$ subsimplexes to the size of the original simplex define $N+1$ local coordinates called simplex coordinates. These coordinates define exactly the position of the point P in the simplex S. The size of a simplex in 1-D refers to its length, in 2-D to its area and in 3-D to its volume.

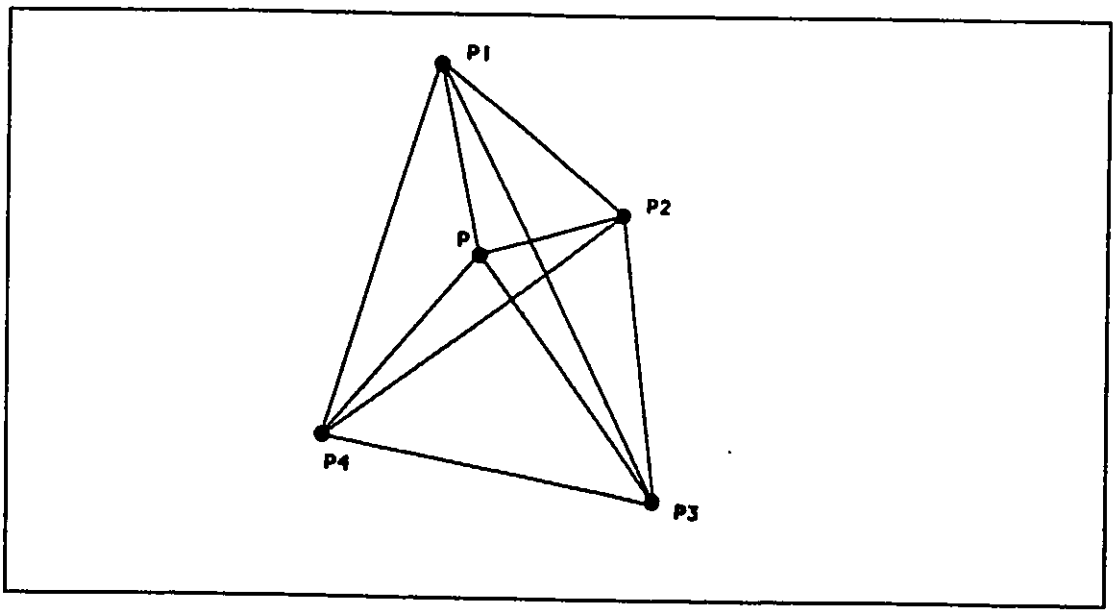

Figure 3.4: One simplex divided in four subsimplexes

Let $\delta_1, \delta_2, \delta_3, \delta_4$ denote the four simplex coordinates and P_1, P_2, P_3, P_4 represent the four vertices of a simplex S. The coordinates of a point P inside S are given by:

$$\begin{aligned} \delta_1 &= \frac{V_1}{V} \quad \delta_2 = \frac{V_2}{V} \\ \delta_3 &= \frac{V_3}{V} \quad \delta_4 = \frac{V_4}{V} \end{aligned} \tag{3.19}$$

where: V is the volume of the simplex S
V_1 is the volume of the subsimplex having P, P_2, P_3, P_4 as vertices,
V_2 is the volume of the subsimplex having P, P_1, P_3, P_4 as vertices,
V_3 is the volume of the subsimplex having P, P_1, P_2, P_4 as vertices,
V_4 is the volume of the subsimplex having P, P_1, P_2, P_3 as vertices.

To note that when P coincides with one of the vertices P_1, P_2, P_3 or P_4, say P_i, then $\delta_i = 1$. When P falls on a face defined by say P_1, P_2, P_3 then $\delta_4 = 0$. A point P on an edge travelling from one vertex P_1 to P_2 will see the simplex coordinate δ_1 vary from 0 to 1. When P is halfway between P_1 and P_2, $\delta_1 = 1/2$.

The volume of the tetrahedron simplex defined by four vertices P_1, P_2, P_3, P_4 such that the coordinates of vertex P_i is

$$P_i \text{ at } (x = x_i, y = y_i, z = z_i) \tag{3.20}$$

is given by

$$V = \frac{1}{3!}\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}. \tag{3.21}$$

When the point P at (x, y, z) is inside S, we can define the coordinate simplexes $\delta_1, \delta_2, \delta_3, \delta_4$ as follows:

$$\delta_1 = \frac{1}{3!V}\begin{vmatrix} 1 & x & y & z \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \quad \delta_2 = \frac{1}{3!V}\begin{vmatrix} 1 & x & y & z \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$

$$\delta_3 = \frac{1}{3!V}\begin{vmatrix} 1 & x & y & z \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \quad \delta_4 = \frac{1}{3!V}\begin{vmatrix} 1 & x & y & z \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{vmatrix}$$

Using the cofactor expansion along the first row, the simplex coordinates can be rewritten as

$$\begin{aligned} \delta_1 &= A_1 + B_1 x + C_1 y + D_1 z \\ \delta_2 &= A_2 + B_2 x + C_2 y + D_2 z \\ \delta_3 &= A_3 + B_3 x + C_3 y + D_3 z \\ \delta_4 &= A_4 + B_4 x + C_4 y + D_4 z \end{aligned} \tag{3.22}$$

where: A_i is the cofactor of entry e_{11} in the definition of δ_i,
B_i that of entry e_{12},
C_i that of entry e_{13} ,
D_i that of entry e_{14}.

Equation (3.22) will define a point P as a function of local simplex coordinates $\delta_1, \delta_2, \delta_3, \delta_4$, themselves being functions of the global Cartesian coordinates.

Interpolation in a Tetrahedron

To be able to approximate the solution $\tilde{H}_{xi}$ inside a tetrahedral FE, one must use interpolation functions that are defined inside the FE. The variation they will provide should represent the variation of the real solution H_{xi} over the FE. The solution $\tilde{H}_{xi}$

will be locally defined by values of the field H_x at special positions in the FE called nodes.

The choice of the order of these interpolation functions depends on many factors. In 2-D, it has been found that it is better to use a coarse meshing, i.e. larger finite elements, and higher order interpolation functions than a fine mesh, and lower (first) order interpolation[14]. For a situation of similar node numbers, the former case will achieve a smoother fit due to the interpolation functions curvatures as opposed to a linear segment fitting. No similar result has been documented for 3-D problems. A complete second degree polynomial interpolation function set defined by 10 nodes values has been chosen for this work. The polynomials, called alpha functions and denoted as α can be determined via the following expression:

$$\begin{aligned} R_m(n_0,\delta) &= \frac{1}{m!}\prod_{k=0}^{m-1}(n_0\delta - k) \quad m>0 \\ \alpha_{ijkl} &= R_i(n_0,\delta_1)R_j(n_0,\delta_2)R_k(n_0,\delta_3)R_l(n_0,\delta_4) \\ R_0(n_0,\delta) &= 1 \end{aligned} \tag{3.23}$$

where: $i+j+k+l=n_0$, n_0 being the order of the polynomial.
The alpha functions are obtained by permutating the values 0,1 and 2 between i,j,k and l and keeping the sum $n_0=2$. The ten functions in sequence are:

$$\begin{aligned} \alpha_1 &= 2\alpha_1^2-\alpha_1 & \alpha_6 &= 4\alpha_2\alpha_3 \\ \alpha_2 &= 4\alpha_1\alpha_2 & \alpha_7 &= \alpha_3^2-\alpha_3 \\ \alpha_3 &= 4\alpha_1\alpha_3 & \alpha_8 &= 4\alpha_3\alpha_4 \\ \alpha_4 &= 4\alpha_1\alpha_4 & \alpha_9 &= 2\alpha_4^2-\alpha_4 \\ \alpha_5 &= 2\alpha_2^2-\alpha_2 & \alpha_{10} &= 4\alpha_2\alpha_4. \end{aligned} \tag{3.24}$$

These functions are associated with their respective node values of $\tilde{H}_{xi}$ as displayed on Fig.3.3. The node positions are such that the alpha function is identicaly one on its corresponding node, zero on all the other nodes and outside the FE, and different then zero everywhere else. The interpolation polynomials have one important advantage. They are expressed in terms of local coordinates and thus do not depend on global coordinate transforms. The work involved in evaluating Eq.(3.16) with these functions can easily be programmed because the integration variables $\delta_1,\delta_2,\delta_3,\delta_4$ always vary from zero to one.

The approximate solution $\tilde{H}_{xi}$ written with the functions given in Eq. (3.24) is:

$$\tilde{H}_{xi} = \sum_{i=0}^{10}\alpha_i(\delta_1,\delta_2,\delta_3,\delta_4)h_{xi} \tag{3.25}$$

where: h_{xi} is the value of H_x at node i.
The solution $\tilde{H}_{xi}$ will match h_{xi} exactly at the node positions and will be interpolated in between.

3.2.4 Single Element Contribution

The single element contribution is obtained from the evaluation of the integral in Eq.(3.16) for one FE and the use of the expanded expression of Eq. (3.25).

When W, the weighting function used in Eq.(3.16) is defined as

$$W = \frac{\partial \tilde{H}_{xi}}{\partial h_{xi}} = \alpha_i, \tag{3.26}$$

there will be ten possible weighting functions consisting of the ten interpolation alpha functions. This manner of defining the weighting functions is known as the Galerkin approach of the weighted residual method.

Substituting the ten W_i's and the expression of Eq.(3.25) for $\tilde{H}_x$, one obtains ten integral equations of the form

$$\int_{V_i} [\nabla\alpha_i \cdot \nabla(\sum_{j=1}^{10} \alpha_j h_{xj}) \; - \; k^2\alpha_i(\sum_{j=1}^{10} \alpha_j h_{xj})] \, dV_i \; - \int_{S_i} \alpha_i \frac{\partial H_x}{\partial n} \, dS_i = 0 \quad i = 1, 2, \ldots \tag{3.27}$$

where V_i and S_i are ,respectively, the volume and surface of the i^{th} FE.
It is possible to take all constants and summation signs outside the integrals because of the linearity property:

$$\sum_{j=1}^{10} h_{xj} \int_{V_i} (\nabla\alpha_i \cdot \nabla\alpha_j \; - k^2\alpha_i\alpha_j) \, dV_i - \int_{S_i} \alpha_i \frac{\partial H_x}{\partial n} \, dS_i = 0 \quad i = 1, 2, \ldots \tag{3.28}$$

If the first integral is denoted by $E1_{ij}$, the second by $E2_{ij}$ and the last by $E3_{ij}$, the full system of linear algebraic equations can be written out in matrix form:

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,10} \\ \vdots & & \\ A_{10,1} & & A_{10,10} \end{bmatrix} \begin{bmatrix} H_{x1} \\ H_{x2} \\ \vdots \\ H_{x10} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.29}$$

where $A_{ij} = E1_{ij} - E2_{ij} - E3_{ij}$.

The evaluation of the E terms can easily be systemized due to the type of interpolation functions used. For the first term $E1_{ij}$, the expression to be evaluated works

out to be:

$$\int_{V_i} (\nabla\alpha_i \cdot \nabla\alpha_j)\, dV_i = \int_{V_i} (\frac{\partial\alpha_i}{\partial x}\frac{\partial\alpha_j}{\partial x} + \frac{\partial\alpha_i}{\partial y}\frac{\partial\alpha_j}{\partial y} + \frac{\partial\alpha_i}{\partial z}\frac{\partial\alpha_j}{\partial z})\, dV_i \tag{3.30}$$

With the use of the chain rule and Eq.(3.22), the derivatives with respect to the global Cartesian coordinates can be expressed in terms of the derivatives with respect to the local simplex coordinates as,.

$$\begin{aligned}\frac{\partial\alpha_i}{\partial x} &= \frac{\partial\alpha_i}{\partial\delta_1}\frac{\partial\delta_1}{\partial x} + \frac{\partial\alpha_i}{\partial\delta_2}\frac{\partial\delta_2}{\partial x} + \frac{\partial\alpha_i}{\partial\delta_3}\frac{\partial\delta_3}{\partial x} + \frac{\partial\alpha_i}{\partial\delta_4}\frac{\partial\delta_4}{\partial x} \\ &= B_1\frac{\partial\alpha_i}{\partial\delta_1} + B_2\frac{\partial\alpha_i}{\partial\delta_2} + B_3\frac{\partial\alpha_i}{\partial\delta_3} + B_4\frac{\partial\alpha_i}{\partial\delta_4} = \sum_{k=1}^{4} B_k\frac{\partial\alpha_i}{\partial\delta_k}.\end{aligned} \tag{3.31}$$

Similarly for the other terms in (Eq.3.30) so that E1 becomes:

$$\begin{aligned}\int_{V_i} (\nabla\alpha_i \cdot \nabla\alpha_j)\, dV_i = \int_{V_i} &(\sum_{k=1}^{4} B_k\frac{\partial\alpha_i}{\partial\delta_k})(\sum_{k=1}^{4} B_k\frac{\partial\alpha_j}{\partial\delta_k}) + \\ &(\sum_{k=1}^{4} C_k\frac{\partial\alpha_i}{\partial\delta_k})(\sum_{k=1}^{4} C_k\frac{\partial\alpha_j}{\partial\delta_k}) + (\sum_{k=1}^{4} D_k\frac{\partial\alpha_i}{\partial\delta_k})(\sum_{k=1}^{4} D_k\frac{\partial\alpha_j}{\partial\delta_k})\, dV_i.\end{aligned} \tag{3.32}$$

The second term $E2_{ij}$ does not need to be manipulated and can be integrated directly.

The last term $E3$ is where the ABC is applied. The different ABC's in this work are derived in Section 3.4 of this chapter. The following section explains how the ABC's are incorporated in the FE method via the term $E3$.

The actual evaluation of the integrals in simplex coordinates is worked out in Appendix B for volume and surface integrals.

3.2.5 Implementation of ABC's

The first ABC to be implemented is the one known as the Sommerfeld's ABC. It approximates the variation of the field in the normal direction to the boundary as

$$\frac{\partial H_x}{\partial n} = -jkH_x. \tag{3.33}$$

When Eq.(3.33) is substituted for the term $\frac{\partial H_x}{\partial n}$ in $E3$, it becomes

$$\int_{S_i} \alpha_i\frac{\partial H_x}{\partial n}\, dS_i = -jk\sum_{j=1}^{10} h_{xi}\int_{S_i} \alpha_i\alpha_j\, dS_i \tag{3.34}$$

This term is now similar to $E2$ and its evaluation is straightforward. Eq.(3.34) differs from $E2$ only by a constant factor jk and by the fact that E2 involves a volume integral and Eq.(3.34) is a surface integral.

A higher order ABC similar to Sommerfeld's is the one derived by Bayliss and Turkel[18]. It defines the normal derivative of the field as

$$\frac{\partial H_x}{\partial n} = -jkH_x - \frac{H_x}{r}. \tag{3.35}$$

With the Bayliss-Turkel ABC, $E3$ becomes

$$\int_{S_i} \alpha_i \frac{\partial H_x}{\partial n}\, dS_i = -\sum_{j=1}^{10} h_{xj} \int_{S_i} (jk + \frac{1}{r})\alpha_i\alpha_j\, dS_i \tag{3.36}$$

The term with r inside the integral can be taken out if the outer boundary of the problem on which S_i falls is chosen spherical. r, then, refers to the distance from the source to the boundary and can be considered constant.

A different ABC, more involved in computation, but more accurate theoretically, is an ABC that defines

$$\frac{\partial H_x(\vec{r})}{\partial n} = \frac{1}{4\pi} \oint_{S'} (H_x(\vec{r'}) \frac{\partial^2 \phi(\vec{r},\vec{r'})}{\partial n \partial n'} - \frac{\partial \phi(\vec{r},\vec{r'})}{\partial n} \frac{\partial H_x(\vec{r'})}{\partial n})\, dS'. \tag{3.37}$$

This ABC is derived in detail in the Section 3.3.2. The surface integral defining $E3$, once (3.37) is used, becomes

$$\int_{S_i} \alpha_i \frac{\partial H_x(\vec{r'})}{\partial n}\, dS_i = \frac{1}{4\pi} \int_{S_i} \oint_{S'} \alpha_i [H_x(\vec{r'}) \frac{\partial^2 \phi(\vec{r},\vec{r'})}{\partial n \partial n'} - \frac{\partial \phi(\vec{r},\vec{r'})}{\partial n} \frac{\partial H_x(\vec{r'})}{\partial n}]\, dS' dS_i. \tag{3.38}$$

The first integral is computed over the face S_i of the tetrahedral element using a Riemman sum with $n = 4$. The second integral is evaluated similarly; the kernel is described in more detail in Section 3.2.2.

The use of this last ABC will reduce the sparsity of the system S obtained with the finite element method. Unlike the two previous ABC's whose contributions are functions of the nodes on the finite element being handled, the new ABC has contributions in terms of all the nodes on a whole layer of finite elements enclosing the source.

3.2.6 Notes on The Global System Matrix S

After computing all the expressions for the single element contribution, the matrix system is incorporated into a larger linear system of equations S representing the whole domain of the problem. The rows in the smaller single element contribution system are added to rows in S corresponding to the same global node numbering.

Rows corresponding to Dirichlet nodes in the problem are all zeroed except for the diagonal entry which is set to 1. The entry on the right hand side is set to the value of the Dirichlet node. This scheme eliminates the need for row and column shifting and node number tracking when a row is deleted from the system S. In matrix form, the system is represented as

$$[S][H_x] = [B] \tag{3.39}$$

where: $[S]$ is the square system of linear algebraic equations describing the whole problem,
$[H_x]$ is the column vector of nodal unknowns,
$[B]$ is the right-hand side filled with the Dirichlet values.

The same procedures are repeated to obtain solutions for H_y and H_z.

3.3 Boundary Integral Formulation(BIM)

3.3.1 Introduction

The FEM, with today's computing capabilities, does not permit a solution for a very large domain to be computed. The limitation encountered when working with open problems may be unacceptable and other methods have to be considered. When the inhomogeneities are enclosed by a surface S', integral methods become more suitable. They are, in these situations easier to implement, less demanding in computing requirements and provide a faster solution. One such method is the Boundary Integral Method or BIM.

The BIM will provide an alternative to the FEM and allow a comparison between two numerical methods. It will also be combined with the FEM to yield an alternative ABC. In the next section, the integrals to be used for BIM are derived and the adaptation of the method to the situation described in Section 3.2.1 is described.

3.3.2 Derivation of the Integral Expression

The expression needed is a solution to a boundary value problem in a three-dimensional domain. Starting with Green's first identity in vector form,

$$\int_V (\vec{Q}\cdot\nabla\times\nabla\times\vec{P} - \vec{P}\cdot\nabla\times\nabla\times\vec{Q})dV = \int_{S'=S'_1\cup S'_2} (\vec{P}\times\nabla\times\vec{Q} - \vec{Q}\times\nabla\times\vec{P})d\vec{S}' \tag{3.40}$$

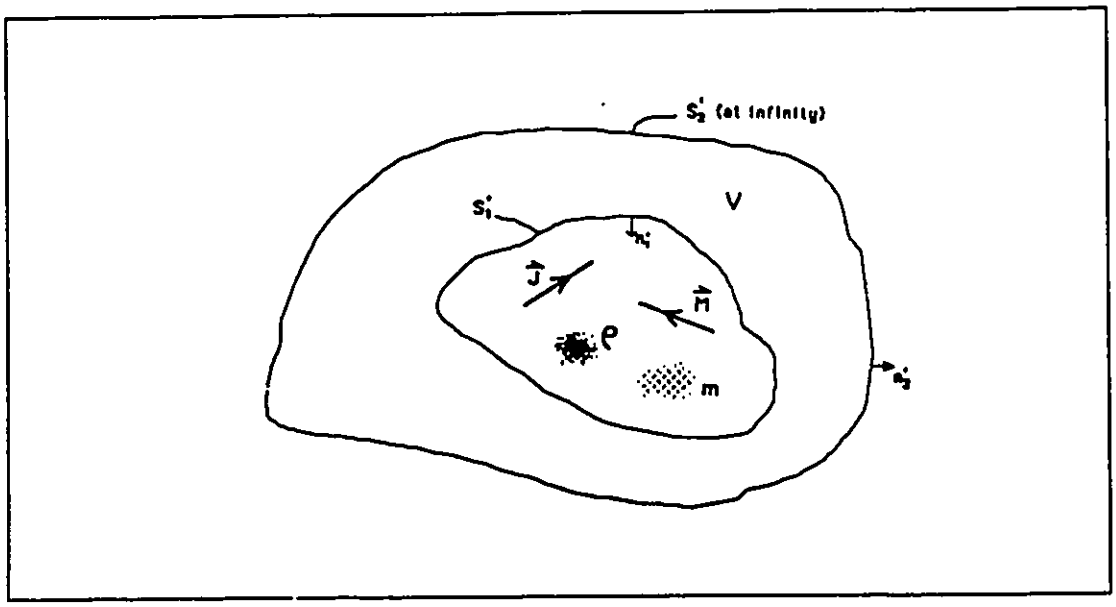

Figure 3.5: General volume enclosed by a surface

where: V is an arbitrary volume,
S' is the surface enclosing V,
$\vec{Q}$ and $\vec{P}$ are two vector functions with continuous 1^{st} and 2^{nd} derivatives.
As it applies to our situation, V is homogeneous and source free. It is also infinite. Figure 3.5 illustrates the extent of V, S'_1 and the terminology used in the equations to follow.

The choice of the vector functions $\vec{P}$ and $\vec{Q}$ is completely arbitrary. Since the problem is solved in a homogeneous open domain, one can replace $\vec{Q}$ by

$$\vec{Q} = \hat{a}\phi(\vec{r},\vec{r}') = \hat{a}\frac{e^{-jk|\vec{r}-\vec{r}'|}}{|\vec{r}-\vec{r}'|} \tag{3.41}$$

where: $\hat{a}$ is a unit vector of arbitrary orientation,
$\vec{r},\vec{r}'$ are respectively the observation and source point position vectors.
$\vec{Q}$ defined as in Eq.(3.41) is recognized to be the Green's function representing the magnetic vector potential due to a point source in a homogeneous domain. Any other $\vec{Q}$ satisfying the particular boundary conditions of a problem can be used. $\vec{P}$ is chosen to be the magnetic field $\vec{H}$.

In the frequency domain the following equations will hold and will be useful for derivation:

$$\nabla \times \vec{E} = -j\omega\mu\vec{H} - \vec{M}; \quad \nabla \cdot \vec{E} = \frac{\rho}{\epsilon}$$
$$\nabla \times \vec{H} = j\omega\epsilon\vec{E} + \vec{J}; \quad \nabla \cdot \vec{H} = \frac{m}{\mu}$$
$$\nabla \cdot \vec{J} = -j\omega\rho; \quad \nabla \cdot \vec{M} = -j\omega m$$

$$\nabla^2\phi - K^2\phi = 0 \tag{3.42}$$

$$\nabla \times \nabla \times \vec{E} - k^2\vec{E} = -j\omega\mu\vec{J} - \nabla \times \vec{M} \tag{3.43}$$

$$\nabla \times \nabla \times \vec{H} - k^2\vec{H} = -j\omega\epsilon\vec{M} + \nabla \times \vec{J} \tag{3.44}$$

where: ω is equal to 2π*frequency
$\vec{J}$ and $\vec{M}$ are the electric and fictitious magnetic current densities
ρ and m are the electric and fictitious magnetic current densities
ϵ and μ are the scalar permitivity and permeability for a homogeneous isotropic media.

If the magnetic field $\vec{H}$ is substituted for $\vec{P}$ and the Green's function $\phi\hat{a}$ for $\vec{Q}$, Eq.(3.40) becomes

$$\int_V (\phi\hat{a}\cdot\nabla\times\nabla\times\vec{H} - \vec{H}\cdot\nabla\times\nabla\times(\phi\hat{a}))dV = \int_{S'=S_1'} (\vec{H}\times\nabla\times\phi\hat{a} - \phi\hat{a}\times\nabla\times\vec{H})d\vec{S}' \tag{3.45}$$

The surface integral over S_2' has been reduced to zero since it extends to infinity and the H-field becomes null.

The first term in the integral can be, with the help of Eq.(3.44),written as

$$\int_V (\phi\hat{a}\cdot\nabla\times\nabla\times\vec{H})\,dV = \int_V \hat{a}\cdot[k^2\vec{H}\phi - j\omega\epsilon\vec{M}\phi + (\nabla\times\vec{J})\phi]\,dV \tag{3.46}$$

By making use of the vector identities A1, A2, A3, A4 and A5 (in Appendix A), the divergence theorem, and the fact that the Green's function satisfies Helmholtz equation as in Eq.(3.42), the second term in Eq.(3.45) can be transformed into

$$\begin{aligned}\int_V (\vec{H}\cdot\nabla\times\nabla\times(\phi\hat{a}))\,dV &= \int_V [\nabla\cdot[(\hat{a}\cdot\nabla\phi)\vec{H}] - (\hat{a}.\nabla\phi)(\nabla\cdot\vec{H}) + \vec{H}\cdot\hat{a}k^2\phi]\,dV \\ &= \int_V [\vec{H}\cdot\hat{a}k^2\phi - (\hat{a}\cdot\nabla\phi)\frac{m}{\mu}]\,dV + \\ &\qquad \oint_{S_1'} (\hat{a}\cdot\nabla\phi)\vec{H}\cdot\hat{n}dS'.\end{aligned} \tag{3.47}$$

The right-hand side of Eq.(3.45), with the identity A6 becomes

$$\oint_{S_1'} (\vec{H}\times\nabla\phi\hat{a} - \phi\hat{a}\times\nabla\vec{H})d\vec{S}' = \oint_{S_1'} \hat{a}\cdot[(\hat{n}_1'\times\vec{H})\times\nabla\phi + \phi\hat{a}\cdot(\hat{n}_1'\times(\nabla\times\vec{H}))]\,dS_1' \tag{3.48}$$

The last terms in Eqs.(3.46) and (3.48) can be rewritten as

$$\int_V (\nabla \times \vec{J})\phi \, dV = \int_V (\vec{J} \times \nabla\phi) \, dV + \oint_{S_1'} (\hat{n}_1' \times \vec{J}\phi) \, dS_1' \tag{3.49}$$

and

$$\oint_{S_1'} \phi\hat{a} \cdot (\hat{n}_1' \times (\nabla \times \vec{H})) \, dS_1' = \oint_{S_1'} \hat{a} \cdot [j\omega\epsilon(\hat{n}_1' \times \vec{E})\phi + \hat{n}_1' \times \vec{J}\phi] \, dS_1' \tag{3.50}$$

with the the help of A6, A7 and Eq.(3.53).

By recombining Eqs.(3.46),(3.47),(3.48) and replacing terms defined by the identities (3.49) and (3.50), Eq.(3.45) becomes:

$$\int_V (-j\omega\epsilon\vec{M}\phi + (\vec{J} \times \nabla\phi) + \frac{m}{\mu}\nabla\phi) \, dV = \oint_{S_1'} [j\omega\epsilon(\hat{n}_1' \times \vec{E})\phi + (\hat{n}_1' \times \vec{H} \times \nabla\phi + (\hat{n}_1' \cdot \vec{H})\nabla\phi] \, dS_1' \tag{3.51}$$

The volume integral will reduce to zero if S_1' encloses all the sources and V becomes a homogeneous domain. Equation (3.51) then simplifies to

$$\oint_{S_1'} [j\omega\epsilon(\hat{n}_1' \times \vec{E})\phi + (\hat{n}_1' \times \vec{H} \times \nabla\phi + (\hat{n}_1' \cdot \vec{H})\nabla\phi] \, dS_1' = 0 \tag{3.52}$$

On the imaginary surface S_1', there will be no physical current density $\vec{J}$ and thus,

$$\nabla \times \vec{H} = j\omega\epsilon\vec{E}. \tag{3.53}$$

Equation (3.53) can be used to write EQ.(3.52) in terms of the magnetic field $\vec{H}$ only;

$$\oint_{S_1'} [\hat{n}_1' \times (\nabla \times \vec{H})\phi + (\hat{n}_1' \times \vec{H} \times \nabla\phi + (\hat{n}_1' \cdot \vec{H})\nabla\phi] \, dS_1' = 0 \tag{3.54}$$

This last equation has a simplified form in the case of Cartesian coordinates and can be reduced to

$$\oint_{S_1'} [\vec{H}\frac{\partial\phi}{\partial n_1'} - \phi\frac{\partial\vec{H}}{\partial n_1'}] \, dS_1' = 0 \tag{3.55}$$

Identities A2, A3, A8 and A9 are used for the transformation.

The definition of the Green's function as in Eq.(3.41) will have a singularity when r', the integration variable, is equal to r. The singularity has to be eliminated since the H-field is differentiable and continuous everywhere and Eq.(3.55) is a finite integral. The discontinuity in ϕ will occur when the observation point r lies on the surface S_1'. To circumvent this problem, the integral over S_1' is partitioned into three integrals

$$\oint_{S_1'} [\vec{H}\frac{\partial\phi}{\partial n_1'} - \phi\frac{\partial\vec{H}}{\partial n_1'}] \, dS_1' + \oint_{S_2} [\ldots] dS_2 + \oint_{S_s} [\ldots] dS_s = 0 \tag{3.56}$$

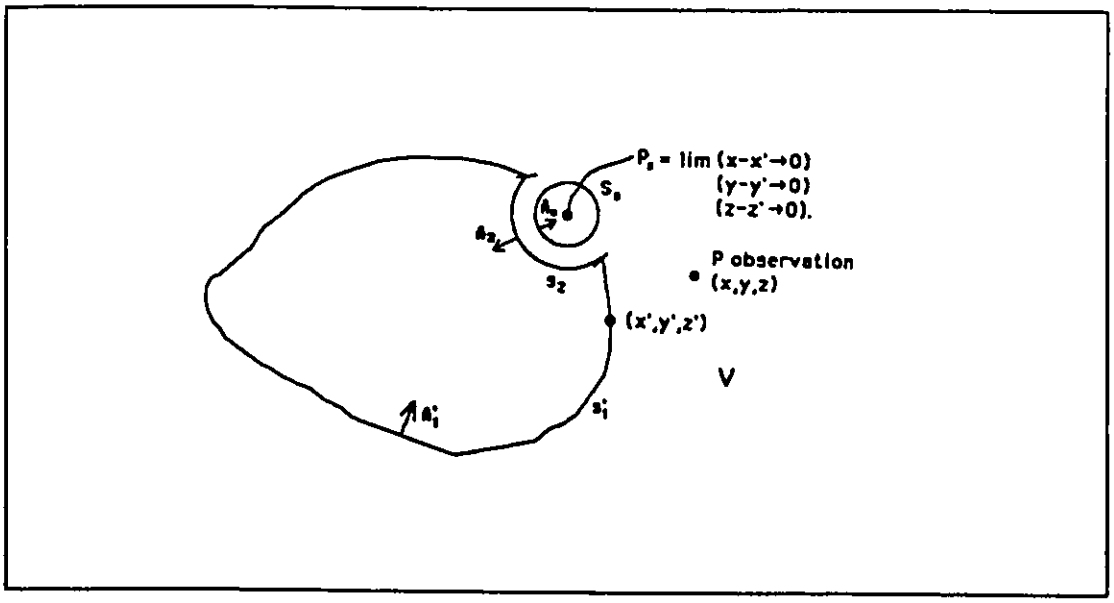

Figure 3.6: Surfaces of integration around the singularity

where S_1, S_2 and S_s are depicted on Fig.3.6.

The center of S_s is at $P_s = |r - r'| = 0$, and because $\hat{n}_s$ is directed inward,

$$\frac{\partial \phi}{\partial n_s} \simeq -\frac{\partial \phi}{\partial r}. \tag{3.57}$$

Over S_s, chosen as a sphere, the variable of integration is constant and is denoted by r_s.

$$\frac{\partial \phi}{\partial n_s} = -\frac{\partial \phi}{\partial r}|_{r=r_s} = -\frac{e^{-jkr_s}}{r_s}[jk + \frac{1}{r_s}] \tag{3.58}$$

The integrals over S_s and S_2, as r_s tends to 0, become two values to be added to the result of the integral over S_1'. If $\bar{H}_x$ and $\frac{\partial \bar{H}_x}{\partial n}$ are defined as the average values, respectively, of the x-component of $\vec{H}$ and $\frac{\partial \vec{H}}{\partial n}$ at r_s, Eq.(3.56) can be written as

$$\lim_{r_s \to 0}[-\bar{H}_x(\frac{jk}{r_s} + \frac{1}{r_s^2}) - \frac{1}{r_s}\frac{\partial \bar{\vec{H}}}{\partial n}]4\pi r_s^2 - [-\bar{H}_x(\frac{jk}{r_s} + \frac{1}{r_s^2}) - \frac{1}{r_s}\frac{\partial \bar{\vec{H}}}{\partial n}]\Omega r_s^2 + \oint_{S_1'}[\ldots]dS_1' = 0 \tag{3.59}$$

with a negative sign before S_2 since $\hat{n}_2 = -\hat{n}_s$. Ω is defined as the solid angle subtented by S_2. The sphere S_s subtends a solid angle of 4π steradians and the half sphere S_2 an angle of 2π steradians. If the limit is taken, Eq.(3.59) is simplified to

$$\bar{H}_x(r_s) = \frac{1}{(4\pi - \Omega)} \oint_{S_1'} [H_x \frac{\partial \phi}{\partial n_1'} - \phi \frac{\partial H_x}{\partial n_1'}]\, dS_1'. \tag{3.60}$$

If the observation point does not lie on S_1 then $\Omega = 0$. The boundary integral equation for all three Cartesian coordinates at an arbitrary observation point r is given by:

$$\vec{H}(\vec{r}) = \frac{1}{4\pi} \oint_{S_1'} [\vec{H}(\vec{r}') \frac{\partial \phi(\vec{r}, \vec{r}')}{\partial n_1'} - \phi(\vec{r}, \vec{r}') \frac{\partial \vec{H}(\vec{r}')}{\partial n_1'}] \, dS_1' \tag{3.61}$$

where $r = \sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2}$. Equation (3.61) appears sometimes in the more popular BEM formulation, however, in this instance, the integral in Eq.(3.61) is simply used, because the kernel is available for computation, to obtain the magnetic field $\vec{H}$ anywhere outside the surface S'. No systems of equations are required to obtain the fields. This same equation is also used in the FEM to evaluate $\partial H/\partial n$ by taking its derivative(see Section 3.2.5).

3.3.3 Adaptation to the Model

Equation (3.61) shows that, if a field on a closed surface is known and the derivative in the direction of the normal to the enclosing surface is known, the field can be computed anywhere outside the surface.

The adaptation of the integral in Eq.(3.61) to the problem depicted in Fig.3.7 starts by defining the surface S' as

$$S' = S_1 \bigcup S_1' \tag{3.62}$$

where S_1' is the image of S_1 with respect to the ground plane. This trick has to be used since the domain, through out the derivation of Eq.(3.61), has been assumed homogeneous. Image theory takes care of the effect of the ground plane and provides the homogeneous domain necessary for Eq.(3.61) to hold. Another approach would have been to derive the Green's function ϕ for an infinite space divided by an infinite perfectly conducting plane and integrate (3.61) over S_1 only. This would have the same result; however, S_1 would have had to be closed by a surface S_g (see Fig.3.8). The surface S_g is just above the ground plane and would extend beneath any apparatus under test resulting in awkard situations for the purpose of measurements.

For the integral kernel in Eq.(3.61), the magnetic field $\vec{H}$ will be measured and is thus known on S_1. To obtain an evaluation of the normal derivative $\frac{\partial H}{\partial n}$, an approximation is made via the first order forward-difference formula

$$\frac{dH}{dn} = \frac{H(x + \Delta n) - H(x)}{\Delta n}. \tag{3.63}$$

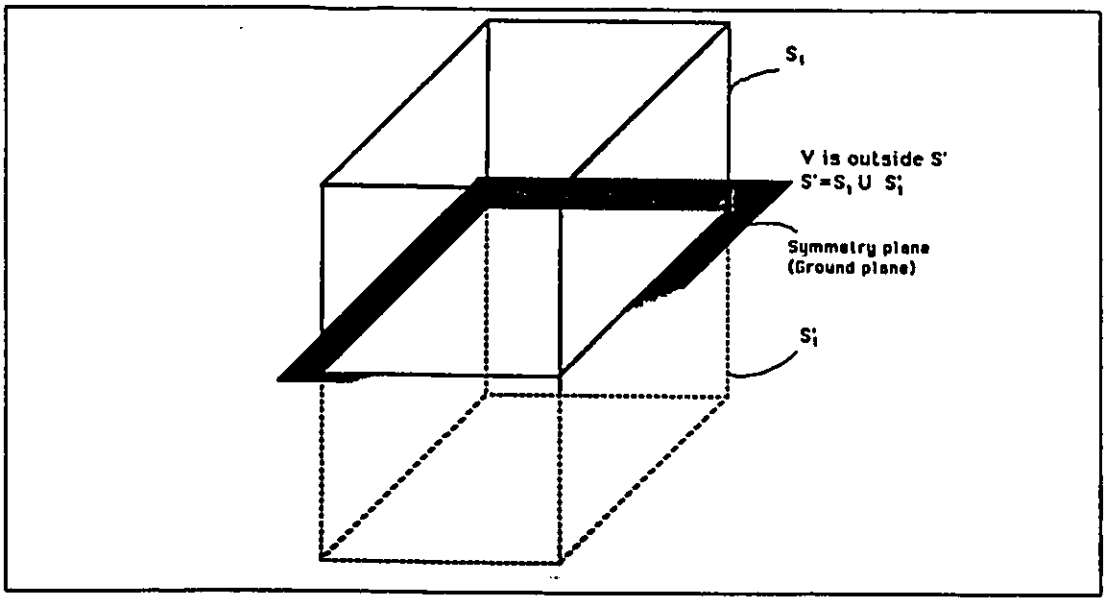

Figure 3.7: Domain and boundaries for BIM solution

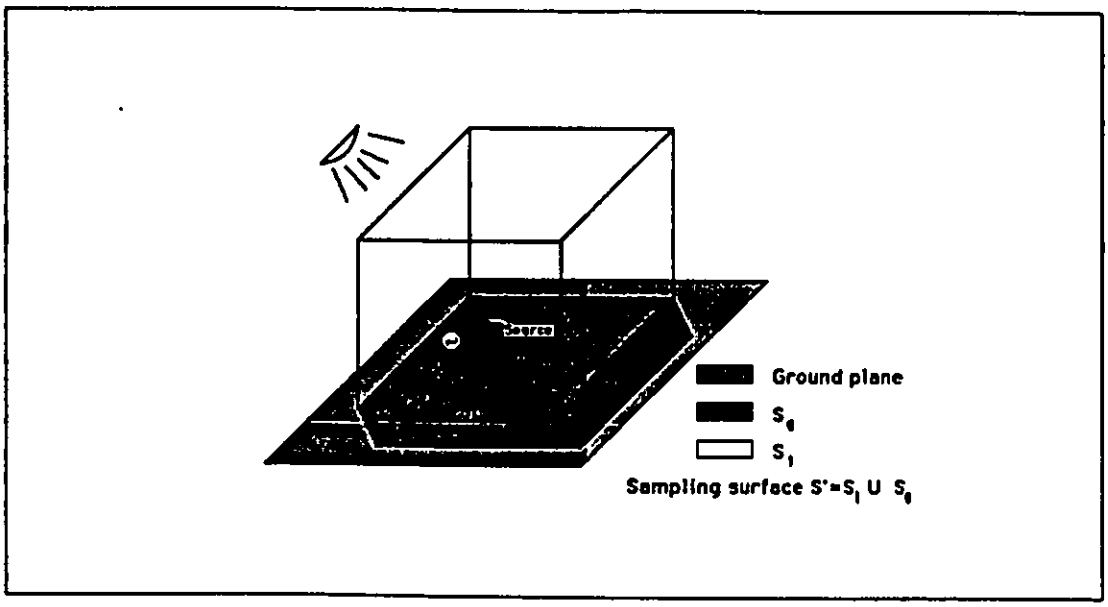

Figure 3.8: Domain and boundaries to be considered when image theory is not used

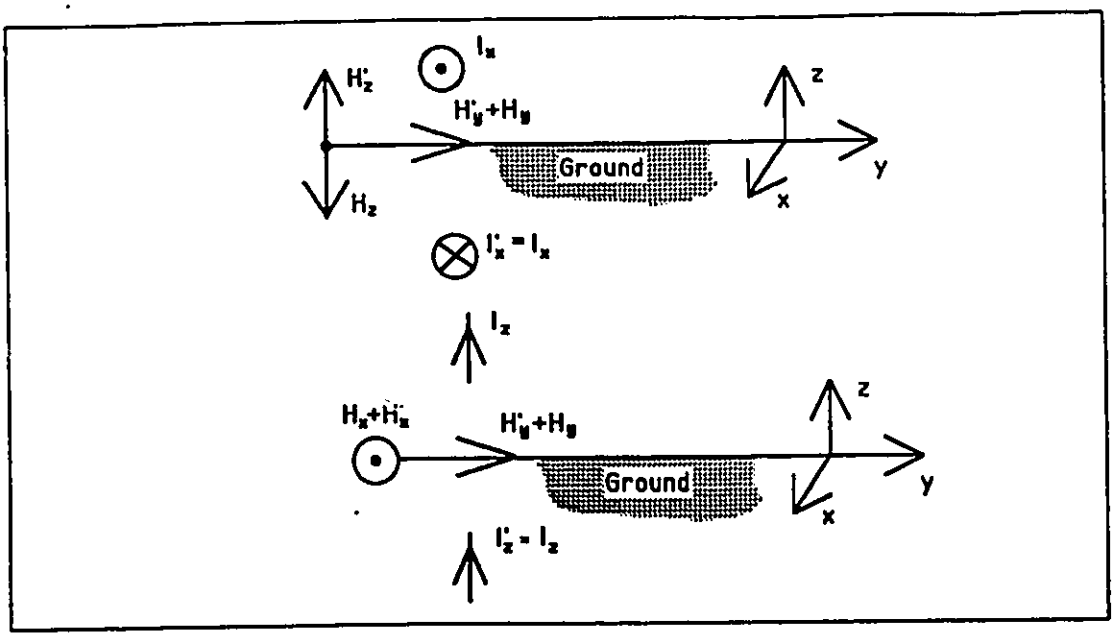

Figure 3.9: Fields generated by a current filament

The step size Δn is defined as a fraction of the wavelength the problem is being solved at. The normal on S_1 as depicted on Fig.3.7 depends on which face of the cubic cover is being considered.

By making use of image theory and analyzing the magnetic field generated by a short current filament above ground, as in Fig.3.9, the relations between $\vec{H}$ and $\vec{H}'$ (the image field), and $\frac{\partial \vec{H}}{\partial n}$ and $\frac{\partial \vec{H}'}{\partial n}$ can be derived from the requirement that the normal component of the magnetic field on a perfect conductor is zero. Table 3.3 summarizes these relations for all possible normal vectors $\hat{n}$ and field components.

To evaluate the integral over S_1, a numerical representation of the surface is needed. This is already available from the mesh that was obtained in the FE method. However, only the first layer of finite elements is used. The mesh does not need to be extended far. The nodes generated by the one layer of finite elements define the positions where measurements are to be made (see Fig.3.10). The basis functions used in FEM are used here to obtain interpolation values for use in the integration. The values of $H(x + \Delta n)$ are obtained in this fashion.

The integral itself is evaluated as a summation of four points on the surface of the bases of every tetrahedral element lying on the Dirichlet boundary. Each triangular base is divided in four equal patches, as in Fig.3.11. It was observed that a finer subdivision of the triangles only slowed the execution without any appreciable gain

	$\hat{n}=\hat{x}$	$\hat{n}=-\hat{x}$	$\hat{n}=\hat{y}$	$\hat{n}=-\hat{y}$	$\hat{n}=\hat{z}$	$\hat{n}=-\hat{z}$
H_x	1	1	1	1	1	1
H_y	1	1	1	1	1	1
H_z	-1	-1	-1	-1	-1	-1
$\frac{\partial H_x}{\partial n}$	1	1	1	1	1	1
$\frac{\partial H_y}{\partial n}$	1	1	1	1	1	1
$\frac{\partial H_z}{\partial n}$	-1	-1	-1	-1	-1	-1

Table 3.3: Correction factor for obtaining the image quantity

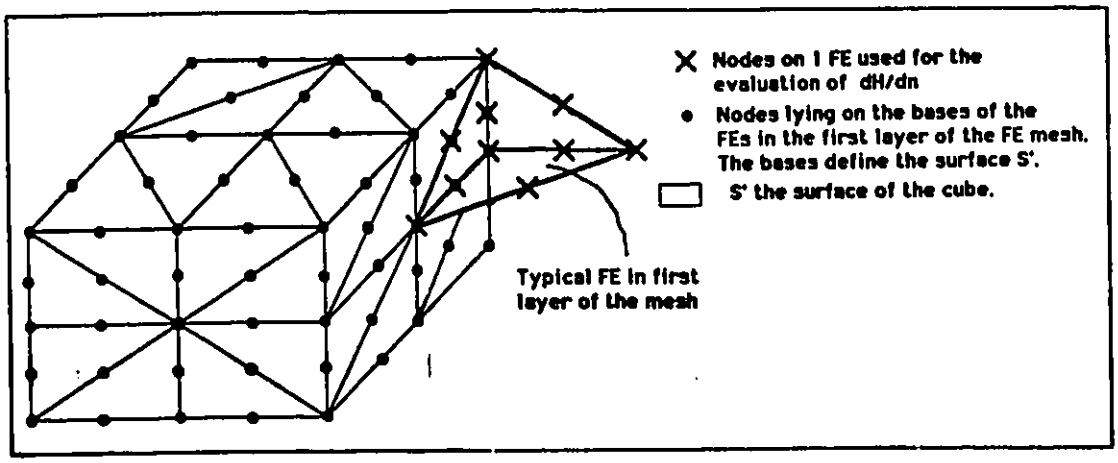

Figure 3.10: Definition of the integration surface with bases of FE's tetrahedrals

in accuracy.

The contribution of every integral over a given triangle is accounted for twice in order to integrate over the image surface as well. The values of the images of H_x and $\frac{\partial H_x}{\partial n}$ were obtained by implementing the sign corrections dictated by Table 3.3 without the need for the involved calculations rising from the interpolation of the field values between nodes.

A clarification about the necessity of using an appropriate evaluation expression for the derivative of the field is necessary. Unlike the finite element approach where derivatives were obtained directly by derivating the interpolation functions, the same cannot be done to obtain a good derivative of the field in the BIM situation. In the finite element method, derivatives at a vertex and in any direction add up to derivatives at the same vertex in parallele direction from other adjacent elements to give a valid final evaluation. This process occurs when single element contributions

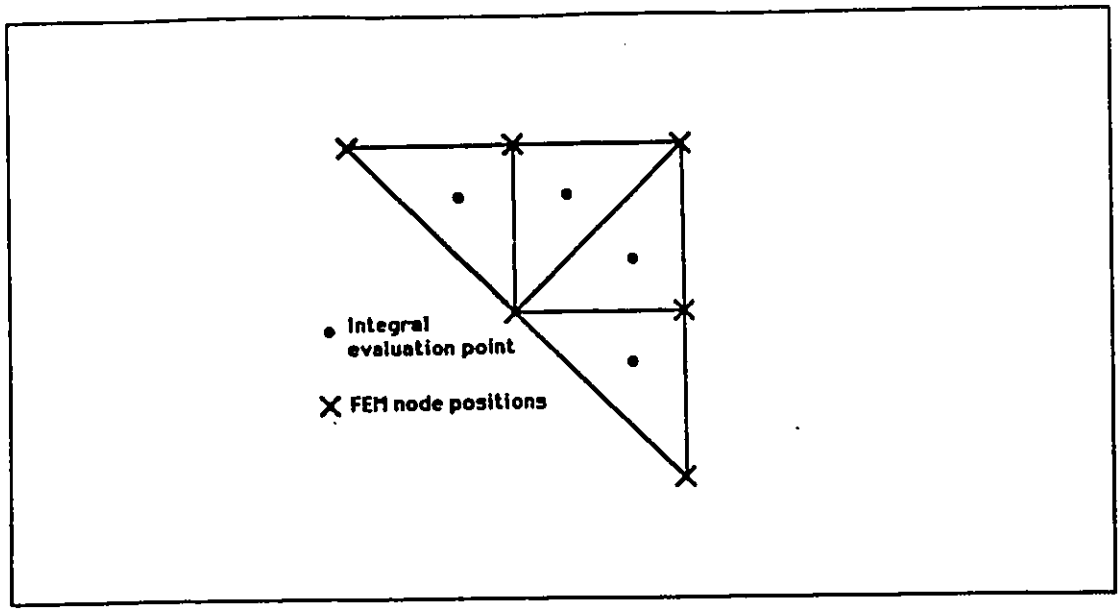

Figure 3.11: Triangle surface subdivision

are added to the global system of equations. Thus, an expression obtained from the derivation on a single element at a given vertex and in a direction parallel to the opposite edge direction will be incomplete because at that position the finite element has no depth. As the point is moved closer to the opposite edge, the derivative becomes more accurate. A diagram describes the situation in Fig.3.12. This means that in the BIM integral a complex algorithm would have to be devised to determine the location of a point in a finite element and catalog all the adjacent finite elements. The alternate route of forward difference formula was chosen to reduce execution time.

3.4 ABC's Derived

3.4.1 Derivation of the Local Type ABC

Two local ABC's are used in this work. They are derived from Wilcox Expansion theorem for electromagnetic fields[16]. The theorem states that:

Theorem 1 *If $\vec{A}(r)$ is a vector radiation function for a given $r > c$ where (r, θ, ϕ) are spherical coordinates, then $\vec{A}(r)$ has an expansion:*

$$\vec{A}(r) = \frac{e^{jkr}}{r} \sum_{n=0}^{\infty} \frac{\vec{A}_n(\theta, \phi)}{r^n} \tag{3.64}$$

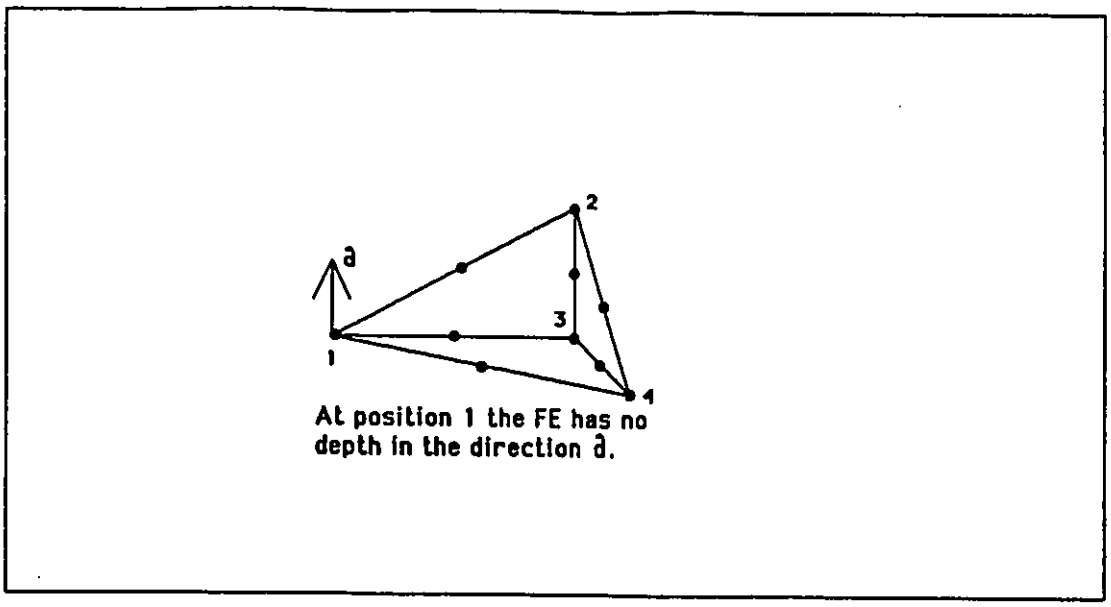

Figure 3.12: Position of points that provide innacurate derivation estimates

which is valid for $r > c$ and which converges absolutely and uniformly in the parameters r, θ and ϕ in any region $r \geq c + \epsilon > c$. The series can be differentiated term by term with respect to r, θ and ϕ any number of times and the resulting series all converge absolutely and uniformly.

A definition of a vector radiating function is given in theorem 2.

Theorem 2 *If $\vec{A}$ is a vector field defined in an exterior domain D. $\vec{A}$ is a vector radiation function for D if and only if the Cartesian components of $\vec{A}$ are scalar radiation function for D and*

$$\lim_{r \to \infty} e^{Im\{kr\}}\, \hat{r} \cdot \vec{A}(r) = 0 \text{ or } \nabla \cdot \vec{A} = 0 \tag{3.65}$$

where $\hat{r} = r\hat{r}$.

From these two theorems, it can be assumed that the Cartesian components of $\vec{H}$ can be expressed as in Eq.(3.64) given that $\vec{H}$ is a vector radiation function satisfying the Helmholtz and radiation condition, and that $\nabla \cdot \vec{H} = 0$. For H_x the series is

$$H_x = \frac{e^{jkr}}{r} \sum_{n=0}^{\infty} \frac{H_{xn}(\theta, \phi)}{r^n}. \tag{3.66}$$

If the normal $\hat{n}$ is chosen in the direction of the spherical coordinate r, as is the case when the outer boundary of the FE mesh is spherical, and if the boundary is far

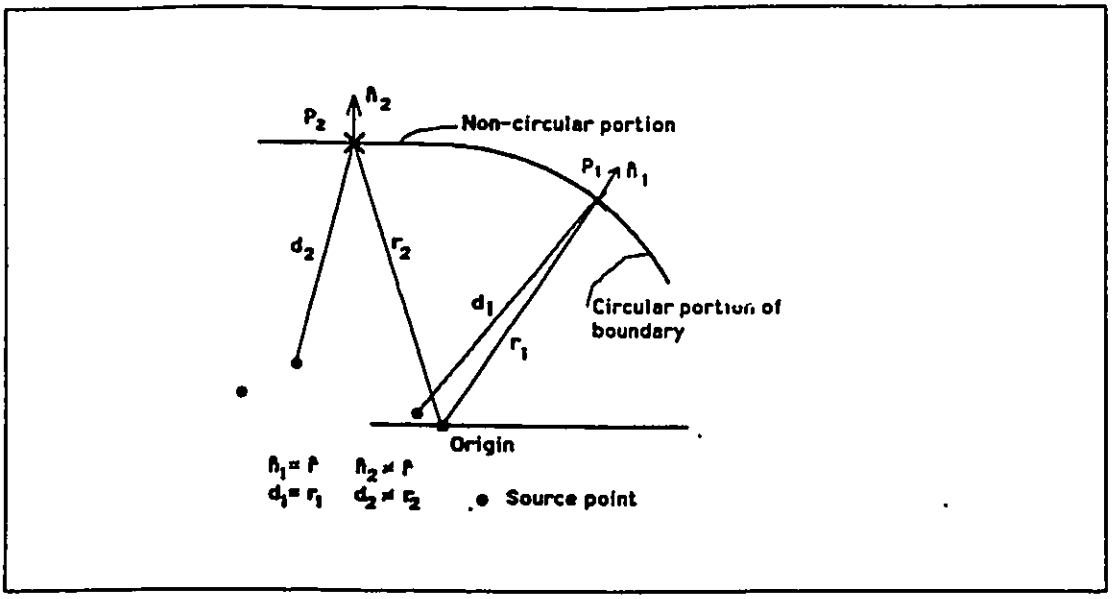

Figure 3.13: Sketch of spherical boundary and the relation between r, r_i and the source position

"enough" (see Fig.3.13 for details of such a configuration), the derivative with respect to n can be replaced by a derivative with respect to r.

$$\frac{\partial H_x}{\partial n} \simeq \frac{\partial H_x}{\partial r} = jkH_x - \frac{1}{r}H_x + Error(O^2). \tag{3.67}$$

The above equation is known as the 1^{st} order Bayliss-Turkel ABC. If only the first term of Eq.(3.67) is retained, a simpler but less accurate expression is obtained.

$$\frac{\partial H_x}{\partial r} = jkH_x \tag{3.68}$$

is known as Sommerfeld's ABC.

3.4.2 Integral Type ABC

In Section 3.3.2, an integral solution to the wave equation was derived. The solution rewritten here for convenience is

$$\vec{H}(\vec{r}) = \frac{1}{4\pi}\oint_{S'}(\vec{H}(\vec{r}')\frac{\partial\phi(\vec{r},\vec{r}')}{\partial n'} - \phi(\vec{r},\vec{r}')\frac{\partial\vec{H}(\vec{r}')}{\partial n'})dS' \tag{3.69}$$

The absorbing boundary condition must provide an evaluation of $\frac{\partial H_x}{\partial n}$. If the derivative is applied to Eq.(3.69), it becomes

$$\frac{\partial\vec{H}(\vec{r})}{\partial n} = \frac{1}{4\pi}\frac{\partial}{\partial n}\oint_{S'}(\vec{H}(\vec{r}')\frac{\partial\phi(\vec{r},\vec{r}')}{\partial n'} - \phi(\vec{r},\vec{r}')\frac{\partial\vec{H}(\vec{r}')}{\partial n'})dS' \tag{3.70}$$

The normal n is defined by the outer boundary of the problem. n' is the normal with respect to the surface where H_x would be already known and can be any interior surface up to the Dirichlet boundary; however, it must still enclose all inhomogeneities. The two surfaces are distinct and are described by two separate coordinate systems; one for the observation point and the other for the Dirichlet surface. By Leibnitz' rule[37], the derivation can be moved inside the integral and performed before the integral.

$$\frac{\partial \vec{H}(\vec{r})}{\partial n} = \frac{1}{4\pi} \oint_{S'} (\vec{H}(\vec{r}') \frac{\partial^2 \phi(\vec{r}, \vec{r}')}{\partial n \partial n'} - \frac{\partial \phi(\vec{r}, \vec{r}')}{\partial n} \frac{\partial \vec{H}(\vec{r}')}{\partial n'}) dS' \tag{3.71}$$

As in the BIM integral of Eq.(3.71), the evaluation is done in the same fashion as in Eq.(3.51). The two equations are similar except for an extra derivative on the Green's function. An analytic expression is easily obtain for these derivatives once the normals are identified.

Chapter 4

Results and Discussions

4.1 The Source Field

The source used to generate the excitation fields in all the approaches has been described in Sec 3.1. Figures 4.1-4.6 illustrate the complexity of the source field achieved with 16 point sources. The real and imaginary part of the Cartesian components of the magnetic fields are given in the near-field at a distance of 1.5m. The signal is a monochromatic signal at 40 MHz with a wavelength of 7.5m. Because the numerical algorithms solve for the real and imaginary parts of the Cartesian components of the magnetic field, the results are plotted in this format to avoid any attenuation of improvements due to the transformation to spherical or magnitude and phase formats.

The expression for the source field has been sampled at positions specified by the mesh generation. In the case of the FEM solution, the positions coincide mainly with nodes at the extremities of, and half-way on all edges of faces of tetrahedrons lying on S_1, the Dirichlet surface. The values of the three Cartesian components of the magnetic field are required. The boundary integral method requires field values for all nodes in the first layer of finite elements adjacent to S_1. Usually, that is the only layer present, however, if the mesh has been generated for use with the FEM, then that may not be the case anymore.

4.2 The Mesher Results

The solution domain described in Fig.3.2 has been discretized using a tetrahedral mesh of finite elements. The discretization required 680 tetrahedrons. The domain mesh was created from 243 primary node points and 1106 secondary nodes defined

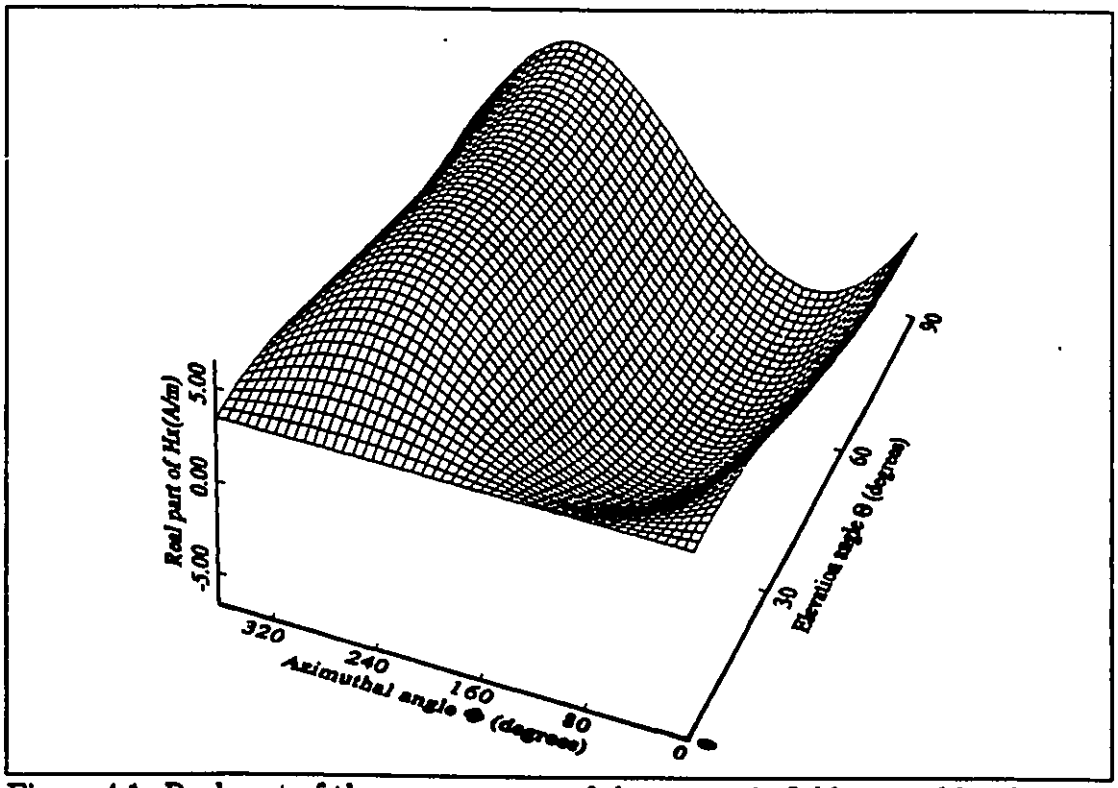

Figure 4.1: Real part of the x-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

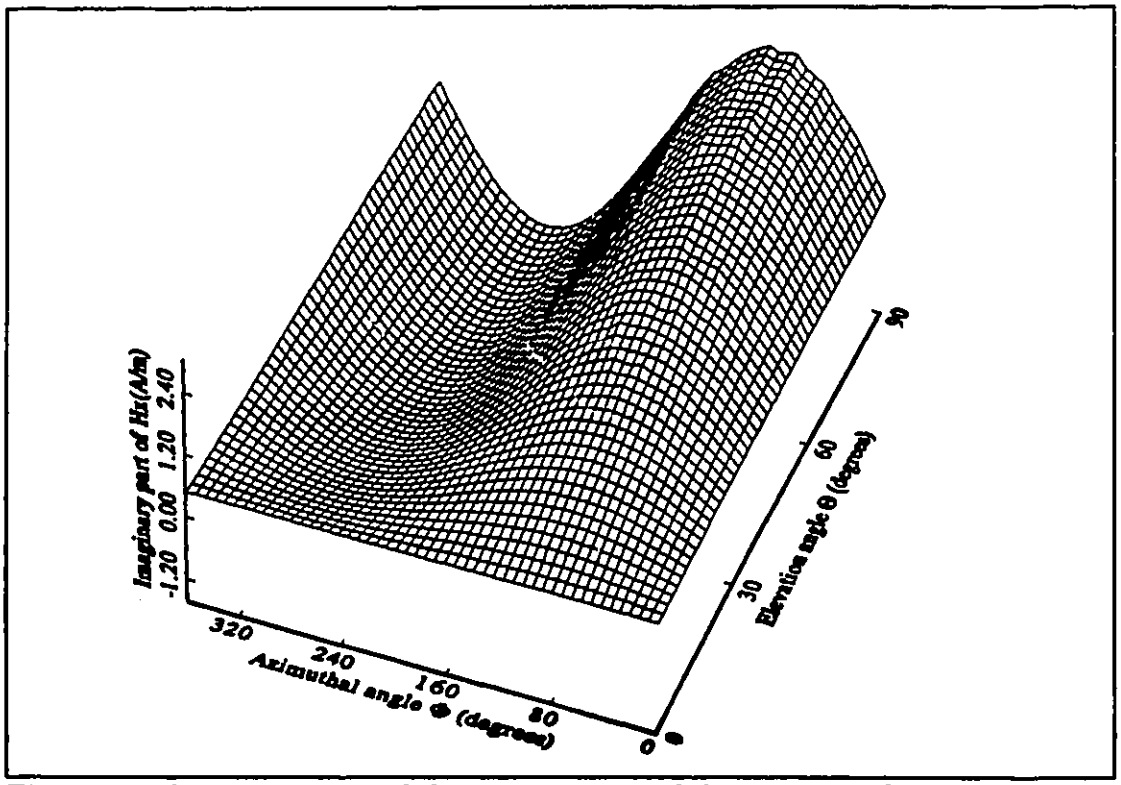

Figure 4.2: Imaginary part of the x-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

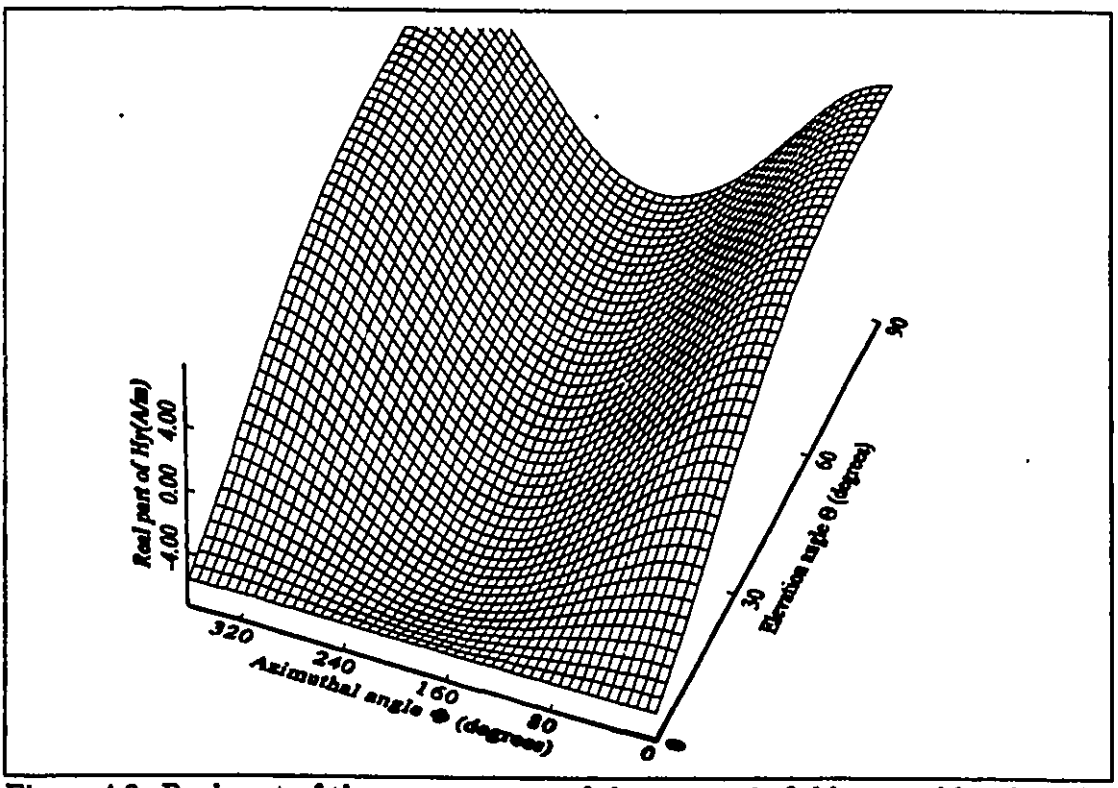

Figure 4.3: Real part of the y-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

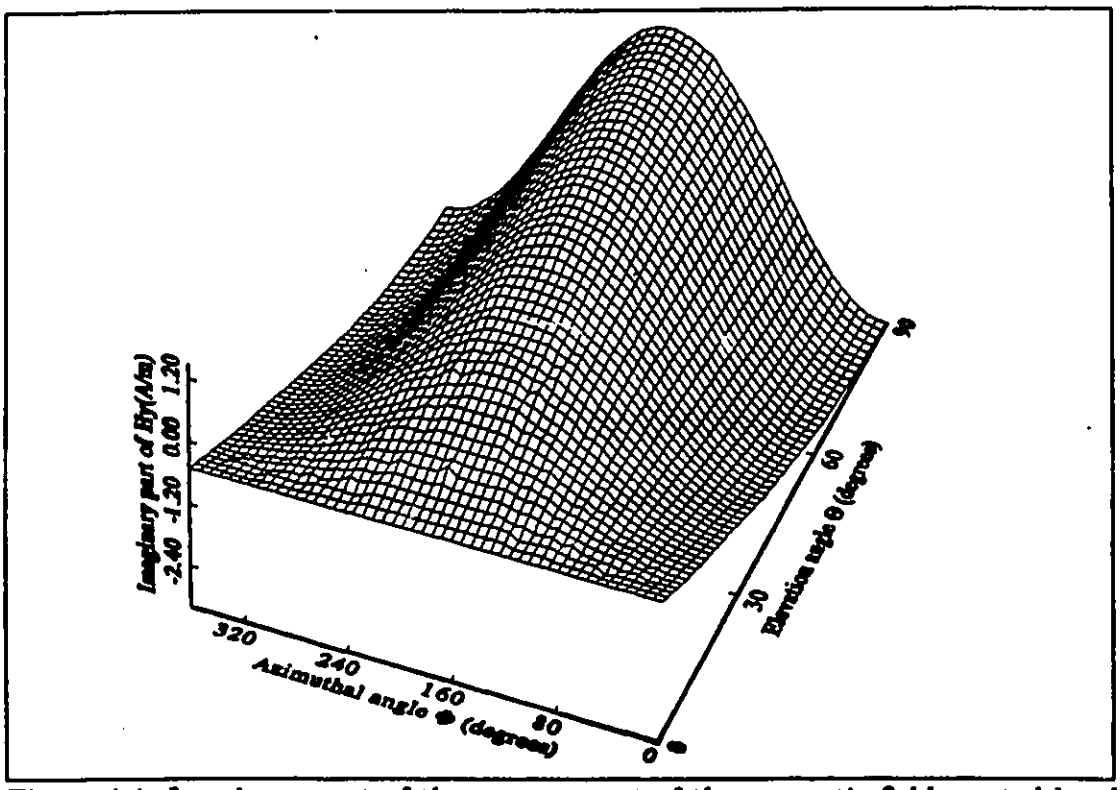

Figure 4.4: Imaginary part of the y-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

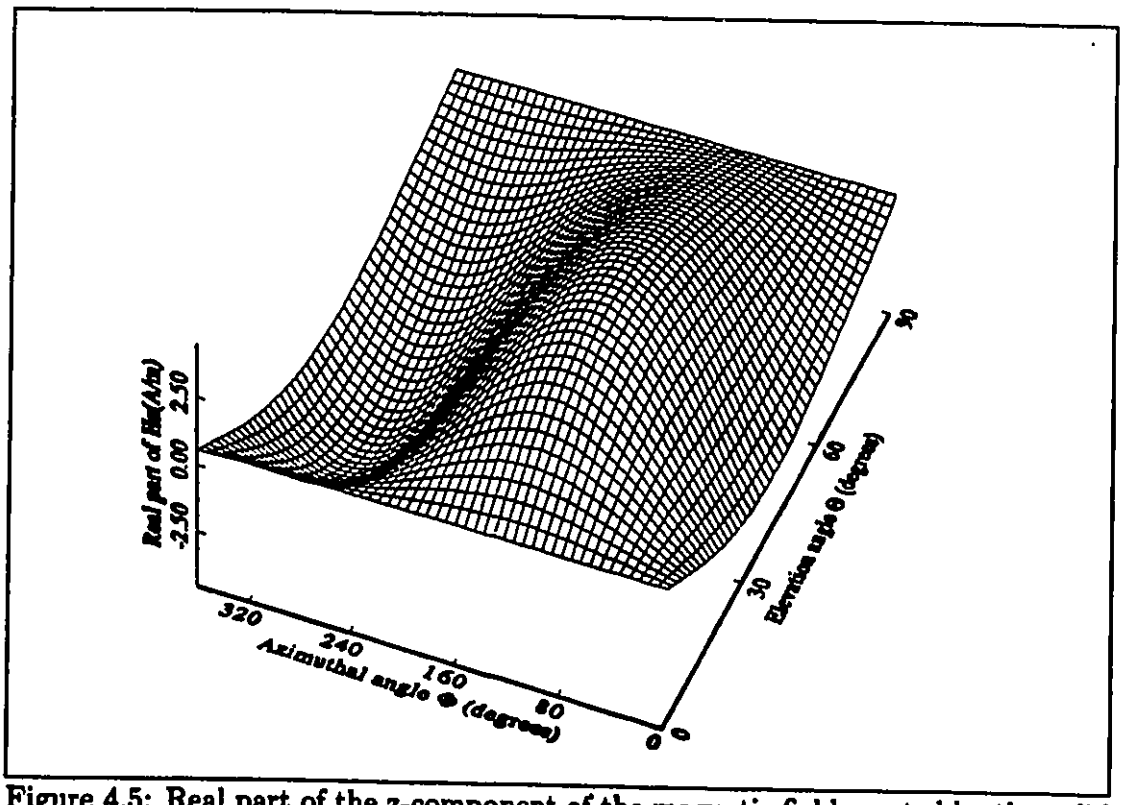

Figure 4.5: Real part of the z-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

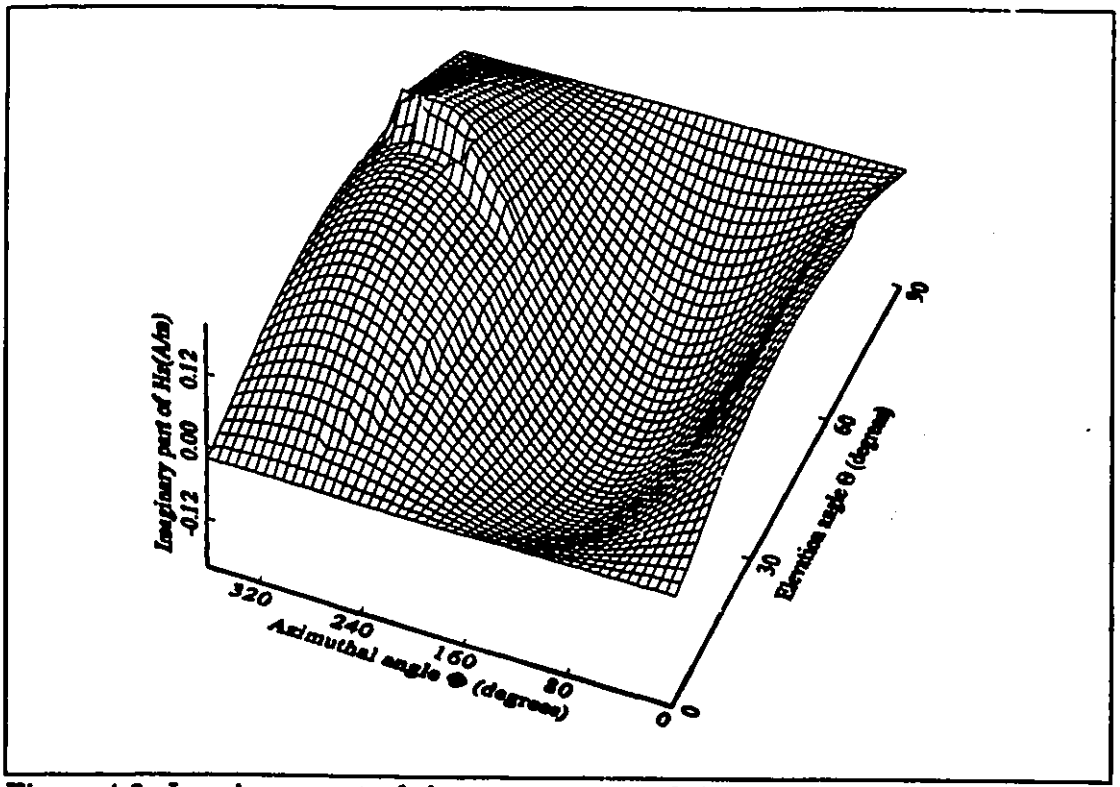

Figure 4.6: Imaginary part of the z-component of the magnetic field created by the validation source at a distance r=1.5m from the origin

halfway on the 1106 edges needed to form the tetrahedrons. These numbers are the ones that set the size of the final system of equations $[S]$ defined in Eq.(3.39). The mesh defines 243+1106 complex unknowns. Figure 4.9 shows the outer tetrahedron faces delimiting the solution domain. It represents S_3, the outer boundary. On Fig.4.8, one can see S_1 and S_2 as defined in Fig.3.2. To be able to apply the first Bayliss-Turkel ABC (BT1), which is more efficient on spherical boundaries, the outer surface S_3 has been mapped onto a spherical dome as in Fig.4.11. The corresponding surfaces S_1 and S_2 are given in Fig.4.10.

The boxes in Figs.4.8 and 4.10 representing S_1 have dimensions 1.5m×1.5m×1.5m and the center of its base is located at the origin of the coordinate system. The plate surrounding the box in Fig.4.8 represents S_2 and has dimensions 3.0m×3.0m. Figure 4.9 shows the cover, S_3, that bounds the solution domain. Its dimensions are 3.0m×3.0m and a heigth of 2.0m. In Figures 4.10 and 4.11, the spherical dome(S_3) and its base(S_2) have a radius $r = \sqrt{0.5^2 + 0.5^2 + 2^2} = 2.91$m. All the faces of the tetrahedral elements making up the surfaces S_1, S_2 and S_3 are identified with respect to which type of boundary they represent for proper computation of the system $[S]$. The tetrahedral shapes are the ones obtained if a cube of .5m×.5m×.5m is divided into five tetrahedrons as in Fig.4.7. The distance between nodes, because the interpolation is a second order interpolation, is 0.25m on two thirds of the edges and $0.25\sqrt{2}$m for the rest or, 21 nodes/wavelength in the worst case. The second layer of FE suffers a deterioration of the number of nodes per wavelength because it has been stretched to map it onto a spherical dome. In some rare case, the ratio can decrease to 15 nodes/wavelength.

Another mesh has been generated for the purpose of executing the boundary integral algorithm. It has the same shape as the mesh in Figs.4.8 and 4.9 but is limited to only one layer of FE with a ratio of nodes per wavelength of 42. This second mesh has 580 tetrahedrons connecting 258 points via 1093 edges.

4.3 ABC Results

The application of the FEM method for an open problem has necessitated the use of absorbing boundary conditions. The choice has been restricted to one local ABC, the first order Bayliss-Turkel(BT1) ABC, and an integral ABC. The accuracy of each of the absorbing boundary conditions has been evaluated directly before being

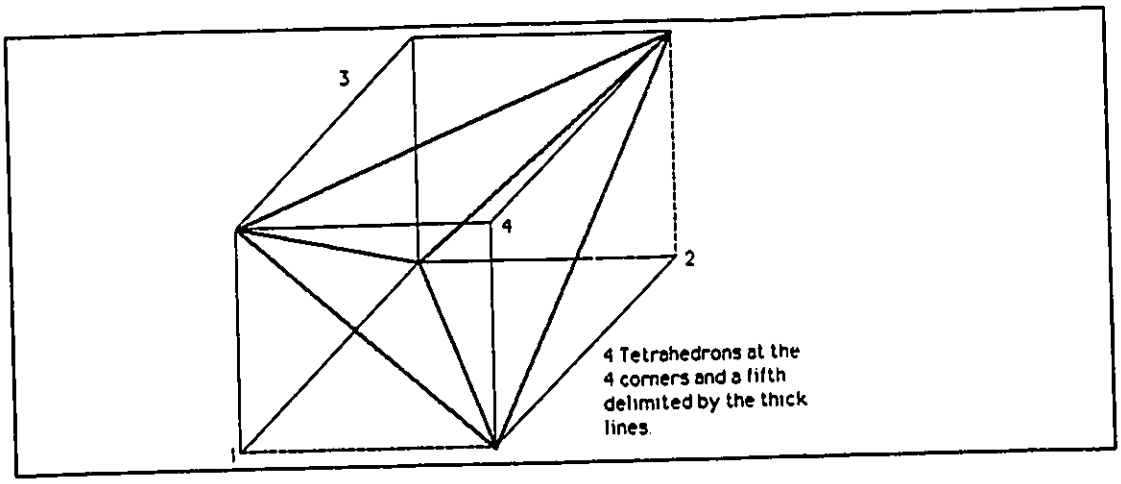

Figure 4.7: Actual shapes of tetrahedrons used in the FEM mesh

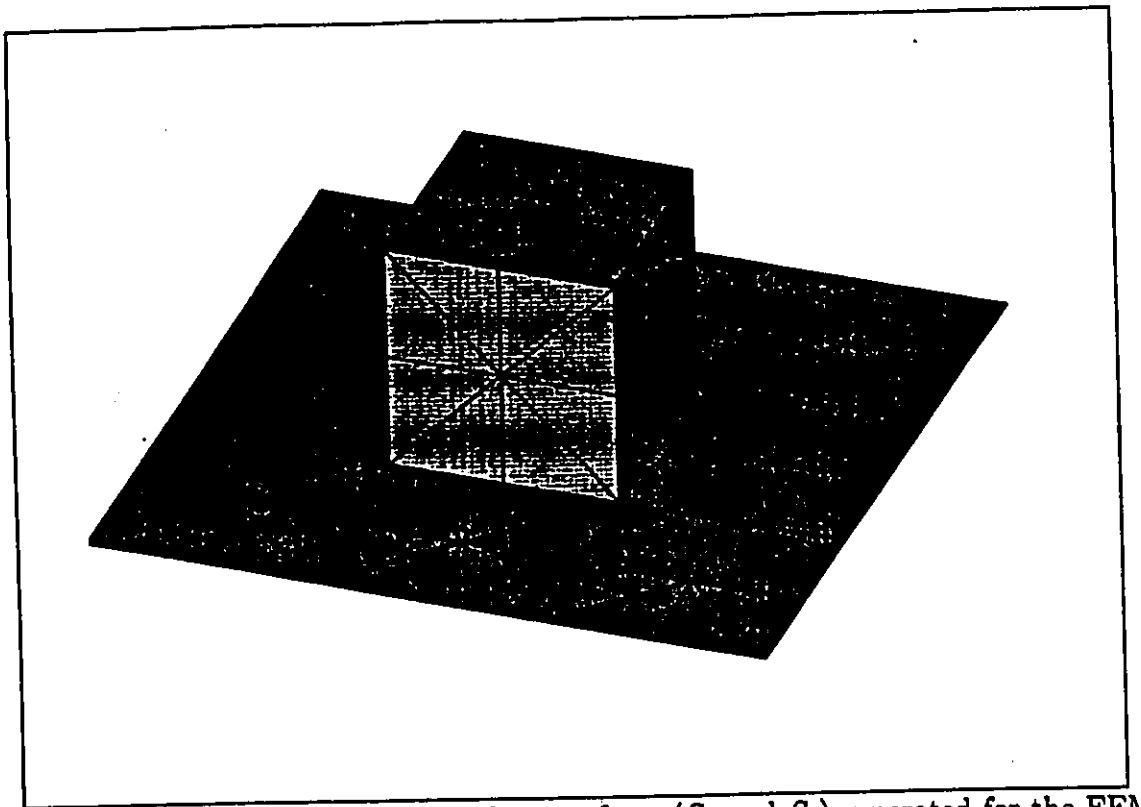

Figure 4.8: Dirichlet and ground plane surfaces (S_1 and S_2) generated for the FEM mesh and integral ABC

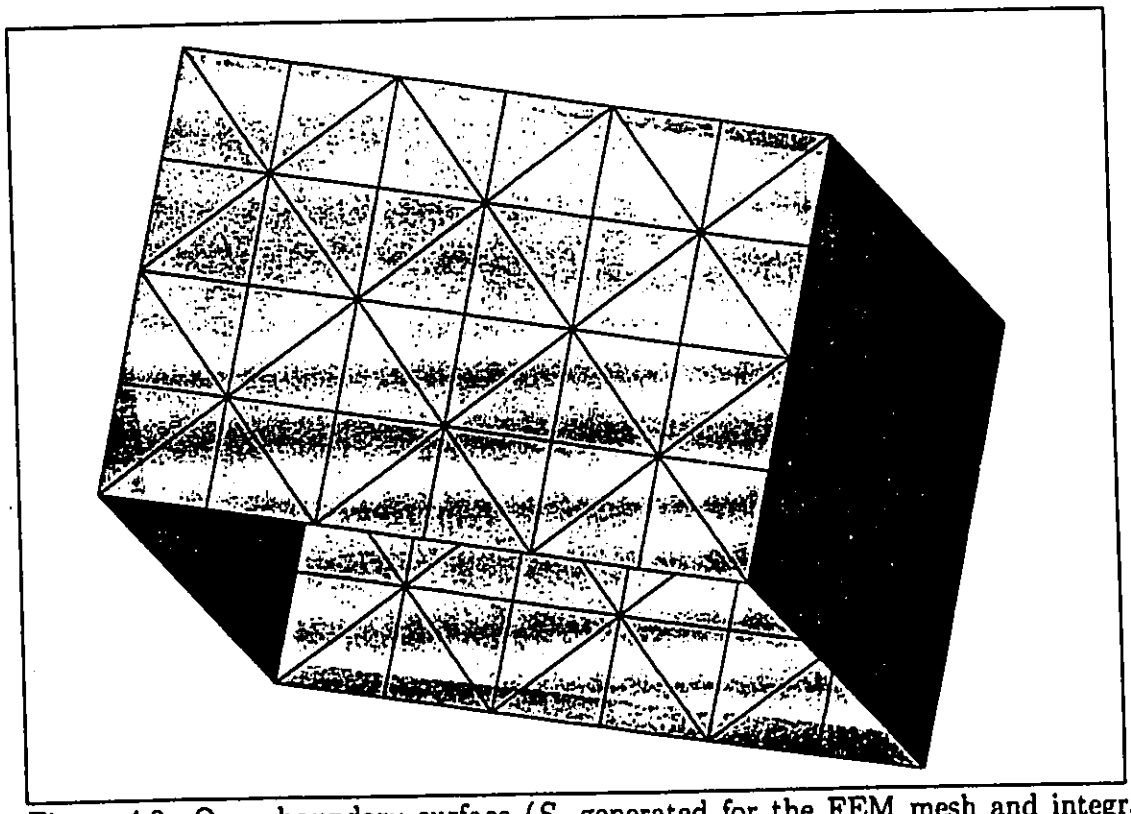

Figure 4.9: Open boundary surface (S_3 generated for the FEM mesh and integral ABC

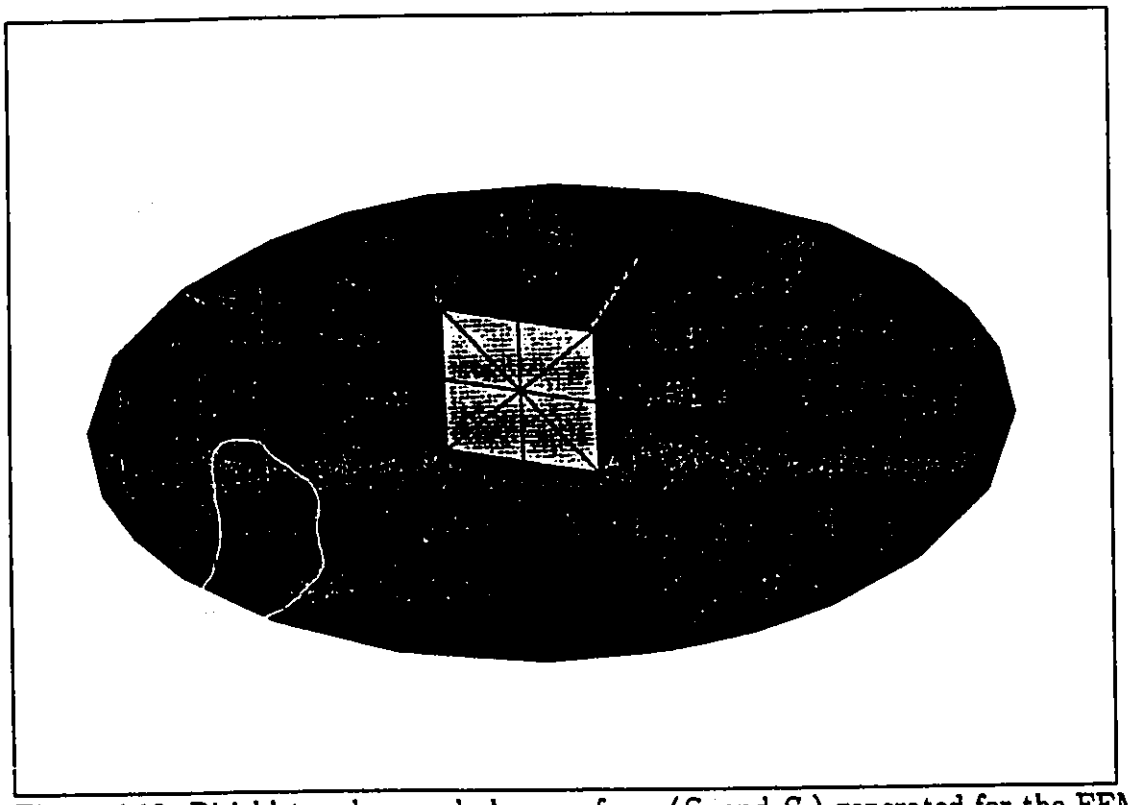

Figure 4.10: Dirichlet and ground plane surfaces (S_1 and S_2) generated for the FEM mesh and BT1 ABC

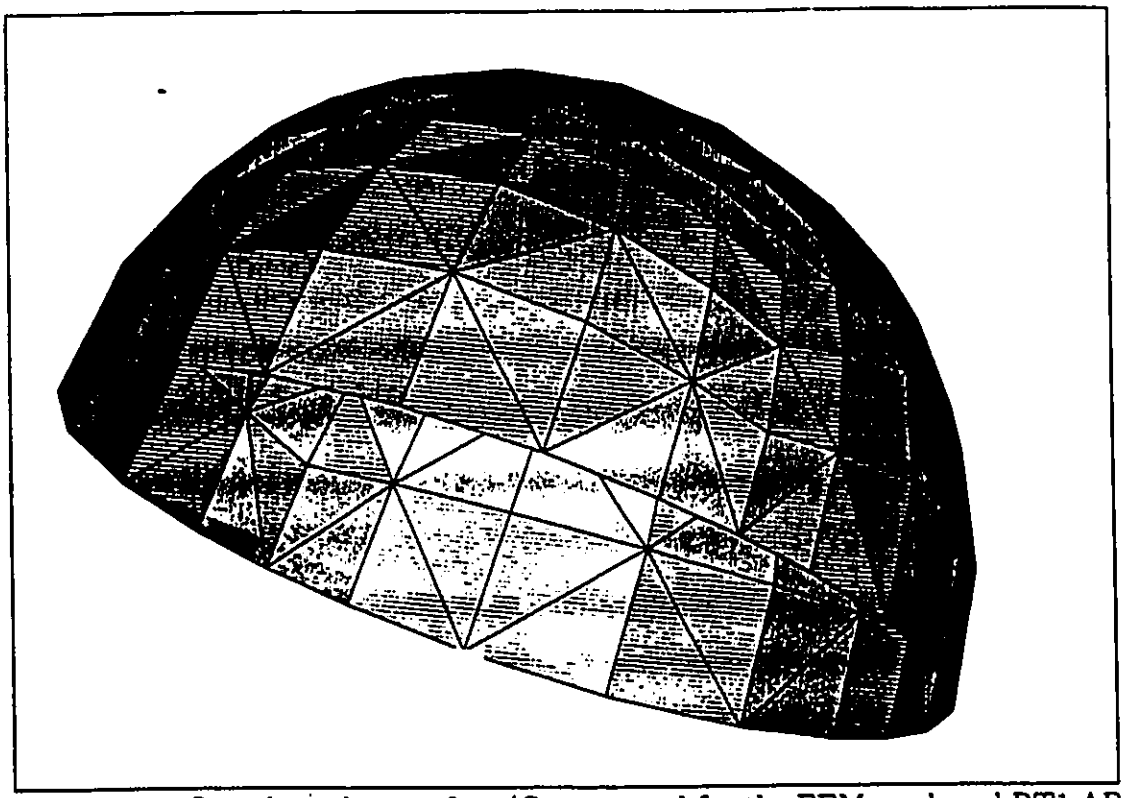

Figure 4.11: Open boundary surface (S_3 generated for the FEM mesh and BT1 ABC

incorporated in the FEM method.

The analytical values of the derivative of the source field in the radial direction were computed using the two point forward formula. The values obtained have been compared with the BT1 and integral ABCs' values. The evaluation was done at a distance of r=2.91m which is the extent of the FEM mesh. A distance of 21m was also used to show the improvements of the local type ABC as one moves away from the source. The elevation angle θ was in both case equal to 37^0.
Figures 4.12 to 4.21 show the real and imaginary part of the derivative with respect to the radial variable for the three Cartesian components of the magnetic field. The improvement is most noticeable in the imaginary part of the ABC. As the observation point is removed further away. The two ABCs converge toward the analytical values. For the 21m distance only the imaginary parts are given since that is where the change occurs.

The improvement with the integral ABC compared to the BT1 ABC can be even more drastic if a more complex source is chosen. In this study, the BT1 has been advantaged by the fact that the derivative with respect to r of the point source analytical field expression is very well approximated by the BT1. One has only to compare the Bayliss-Turkel ABC,

$$\frac{\partial H_z}{\partial r} = -jkH_z - \frac{H_z}{r} = \frac{I\Delta z}{4\pi} e^{-jkr}\left[\frac{k^2}{r} - \frac{2jk}{r^2} - \frac{1}{r^3}\right] \tag{4.1}$$

and the derivative with respect to r of H_z.

$$\frac{\partial H_z}{\partial r} = \frac{I\Delta z}{4\pi} e^{-jkr}\left[\frac{k^2}{r} - \frac{2jk}{r^2} - \frac{2}{r^3}\right] \tag{4.2}$$

In fact the error of approximation decays as O^3. Furthermore, there is no series truncation error in this instance. The derivation with respect to r itself does not introduce much error given that the sources are relatively close to the origin. As the observation point is removed, the convergence of the two ABCs to the analytical values becomes perfect.

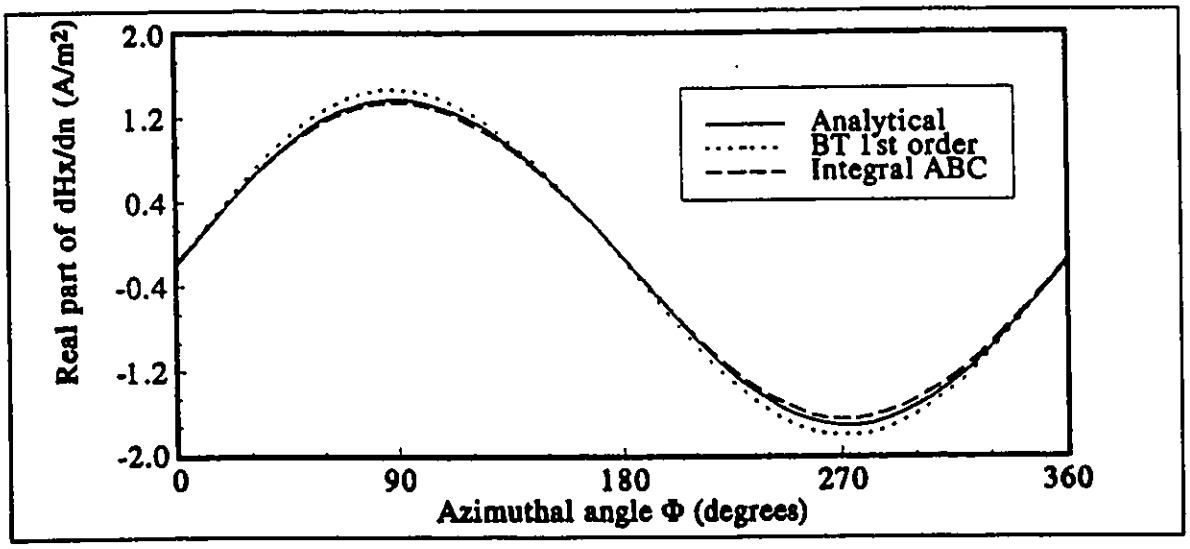

Figure 4.12: Real part of $\partial H_z/\partial r$ at r=2.91m and $\theta = 37^0$

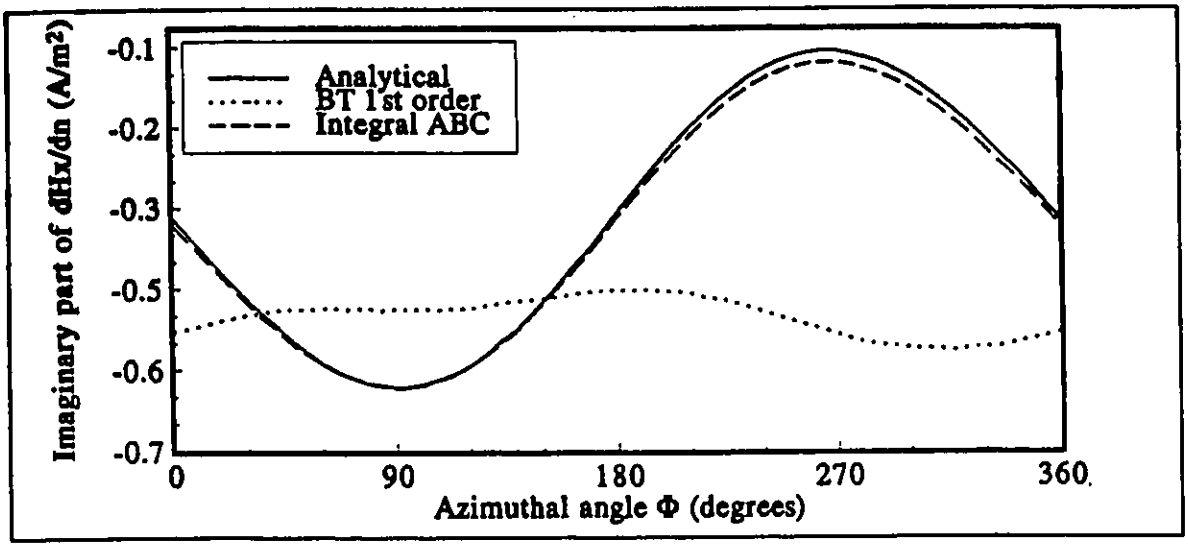

Figure 4.13: Imaginary part of $\partial H_z/\partial r$ at r=2.91m and $\theta = 37^0$

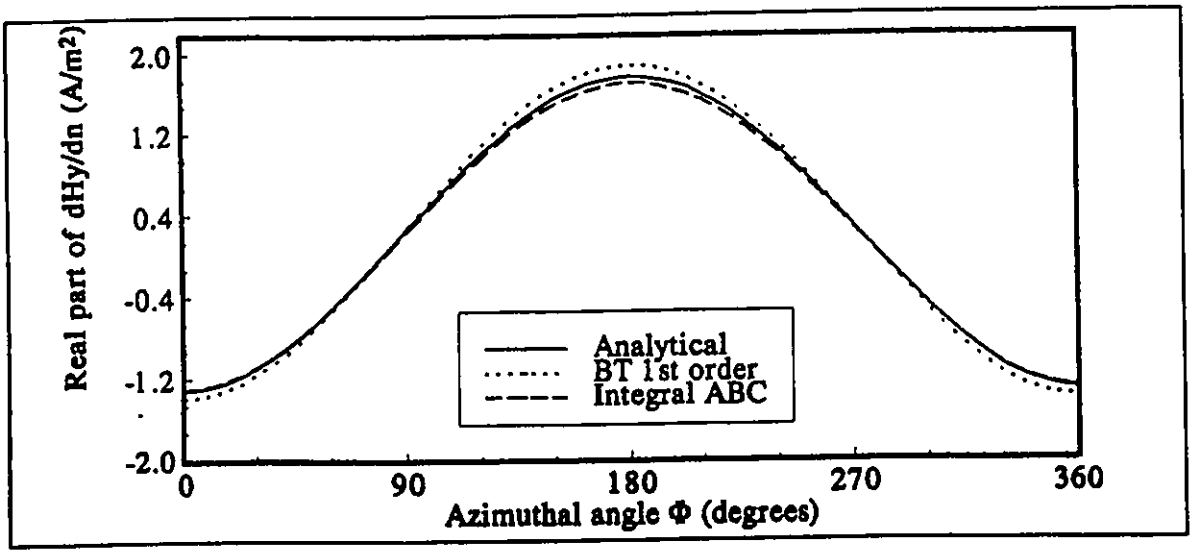

Figure 4.14: Real part of $\partial H_y/\partial r$ at r=2.91m and $\theta = 37^0$

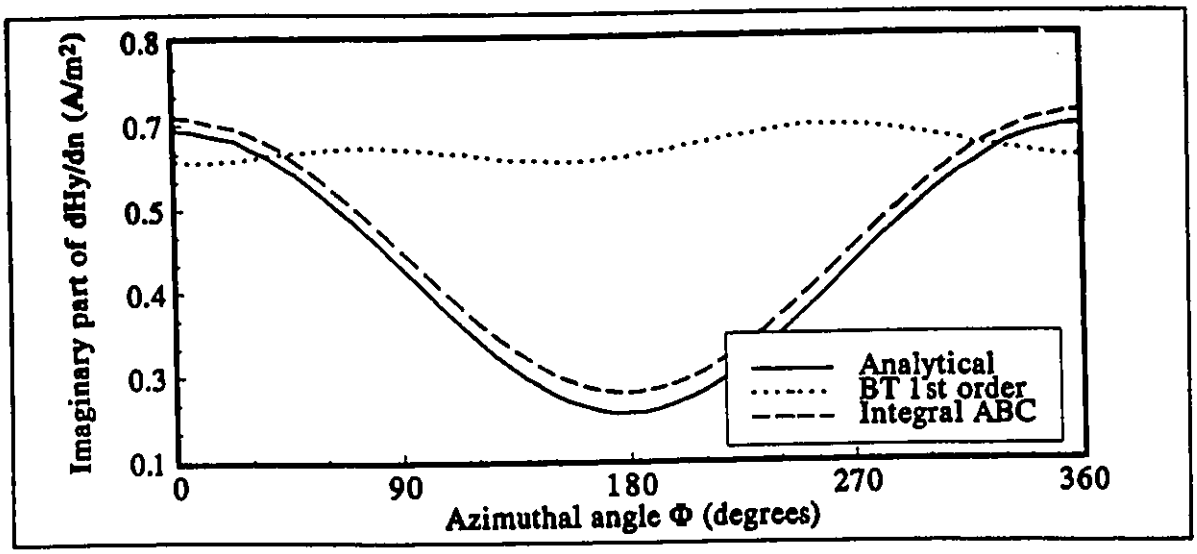

Figure 4.15: Imaginary part of $\partial H_y/\partial r$ at r=2.91m and $\theta = 37^0$

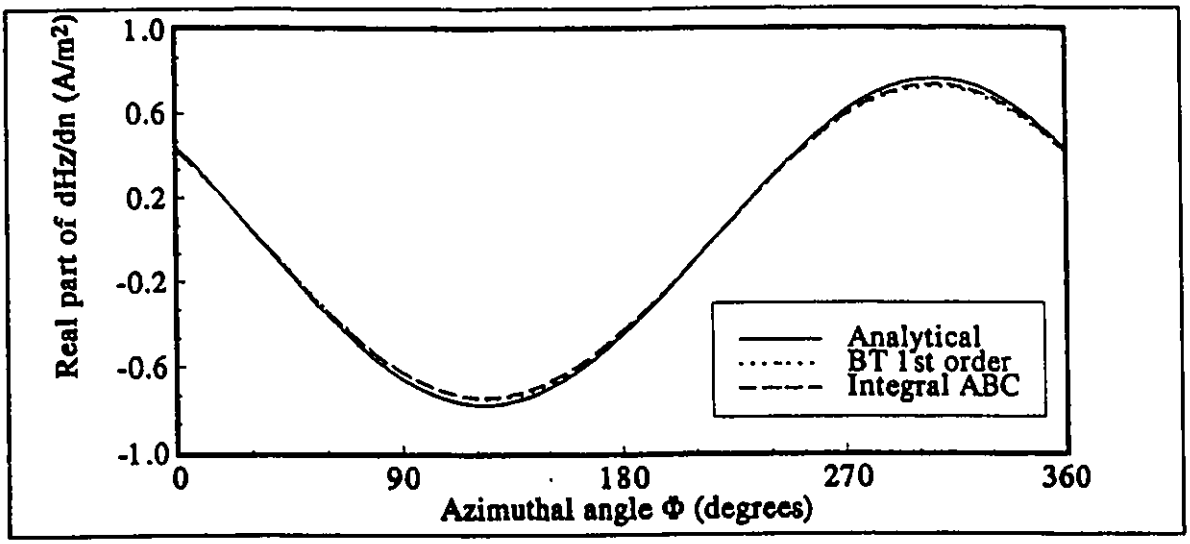

Figure 4.16: Real part of $\partial H_z/\partial r$ at r=2.91m and $\theta = 37^0$

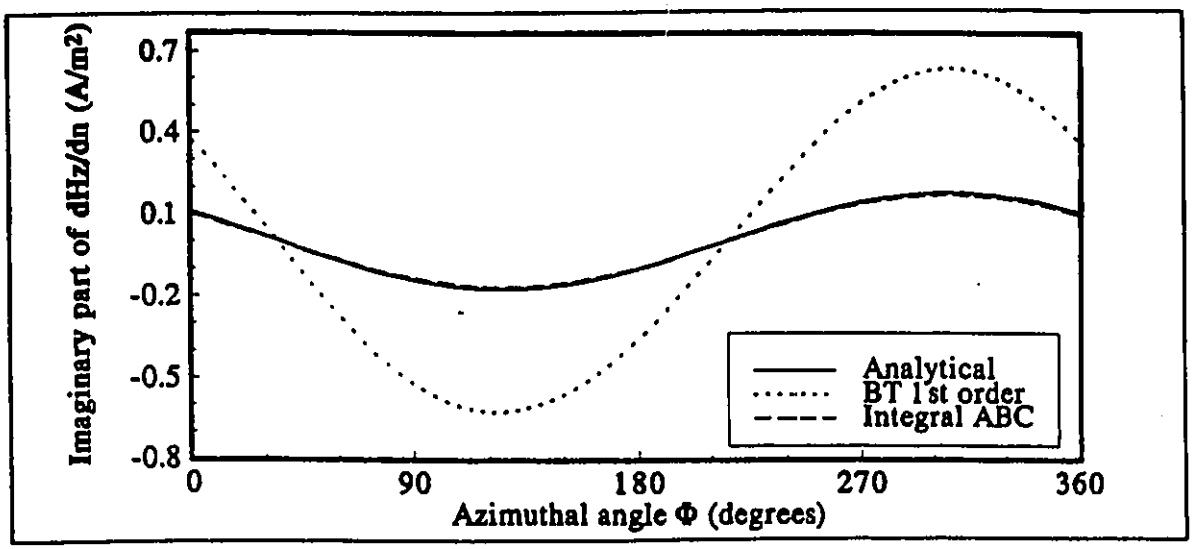

Figure 4.17: Imaginary part of $\partial H_z/\partial r$ at r=2.91m and $\theta = 37^0$

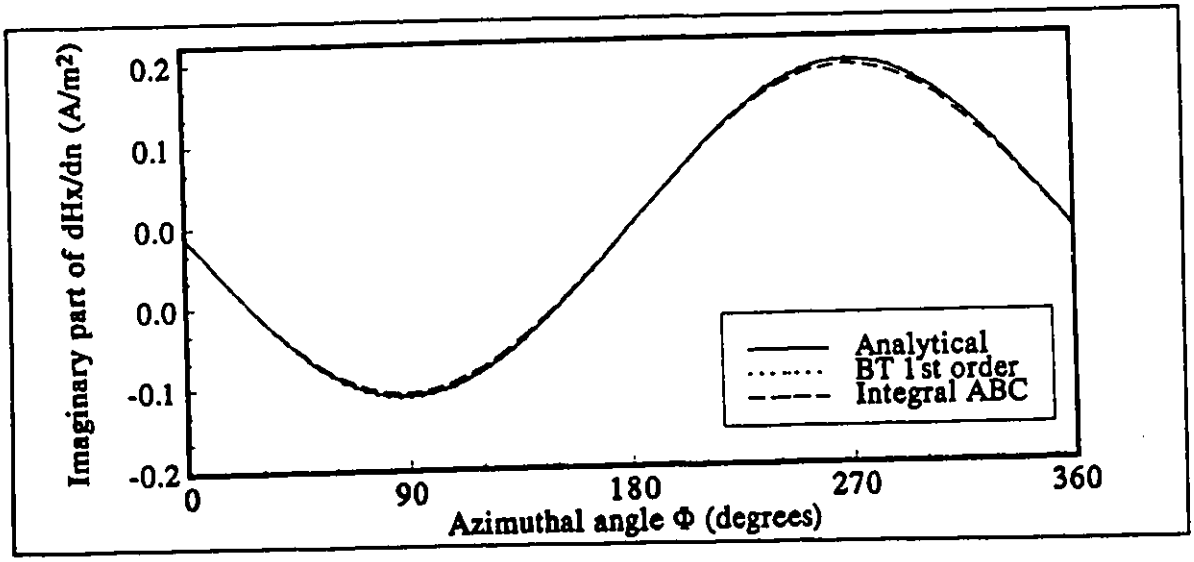

Figure 4.18: Imaginary part of $\partial H_x/\partial r$ at r=21m and $\theta = 37^0$

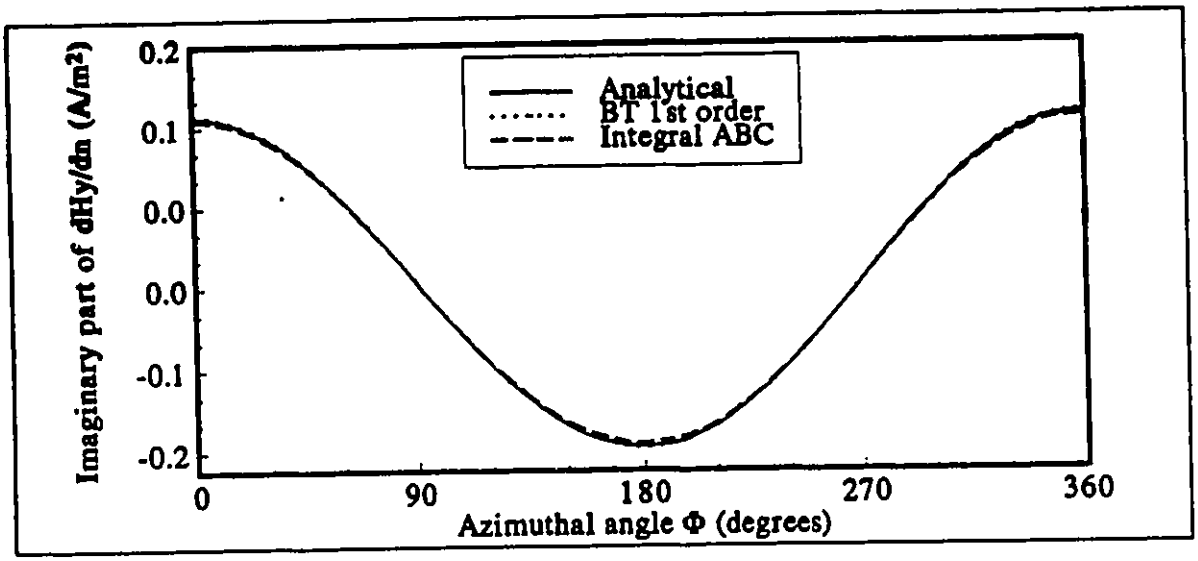

Figure 4.19: Imaginary part of $\partial H_y/\partial r$ at r=21m and $\theta = 37^0$

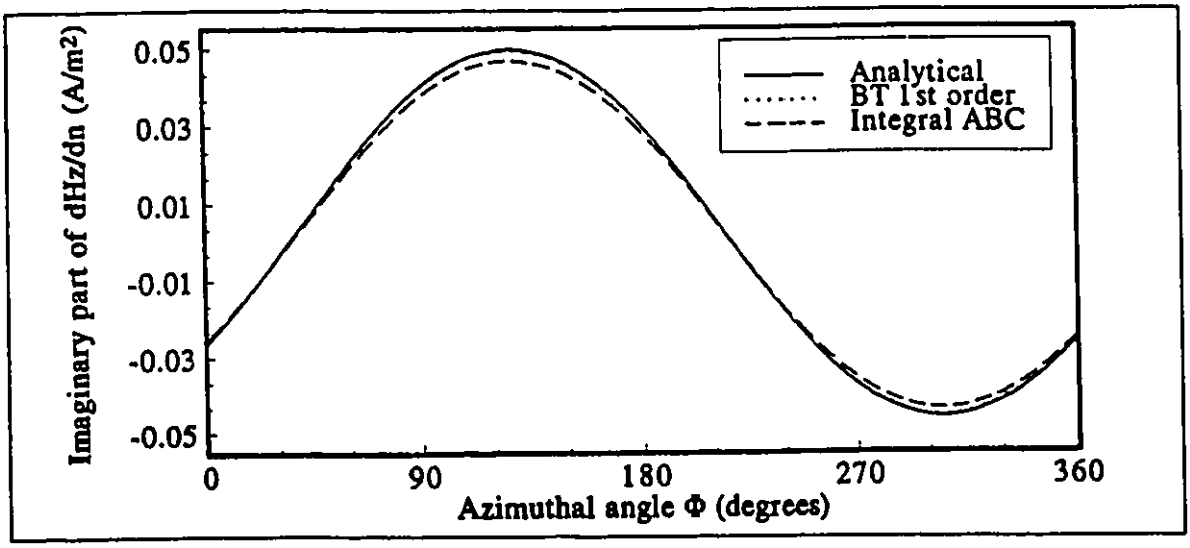

Figure 4.20: Imaginary part of $\partial H_z/\partial r$ at r=21m and $\theta = 37^0$

4.4 FEM Results

The finite element method results have been obtained for the mesh depicted in Figs.4.8 and 4.9. The absorbing boundary conditions have been implemented in the solution to be able to use a truncated mesh. The outer boundary has been placed at one seventh of a wavelength from the Dirichlet boundary. The figures 4.21 to 4.26 display the results obtained with the best combination of mesh shape and ABC. The best results have been obtained with the BT1 local ABC when the outer boundary of the mesh is spherical. The reasons have been covered in Sec.3.3. The integral ABC gave better results when the mesh had a cubic outer boundary. The underlying reason is a simple question of resolution. Because the number of collocation points for the evaluation of the integral in Eq.(3.61) on every facet has been maintained constant and because the facets are stretched to form the spherical boundary, the resolution decreases resulting in better results with the cubic mesh.

Figures 4.21 to 4.26 show the real and imaginary part of the three Cartesian components of the magnetic field at an elevation angle of $\theta = 37^0$ and a distance $r = 2.91m$ (the furthest possible within the cubic mesh). The system of equations obtained with both numerical approaches had 1398 complex unknowns partitioned into 2698 real unkowns (real & imaginary). This was necessary because the package[37] to solve the system was available only for real systems. The system was solved using a di-

agonally scaled biconjugate gradient iterative method. The filling percentage was 0.911%, 0.911% and 0.83% for the three x,y and z components respectively with the local type ABC and 17.17% for the three components when the integral ABC was used. The execution time was of 72 seconds in the first case and 525 seconds in the second case on a Digital 3100 RISC station.

4.5 Discussion of FEM and ABC Results

Although the far-field to near-field transformation has not been achieved with the FEM -only a distance of $\lambda/7$ was possible-, the approach has allowed the study of the local ABC and the derivation of an integral ABC for early mesh truncation. The integral ABC has been shown to work extremely well for a very close outer boundary with the advantage of not being dependent on the source complexity or position. For even higher accuracy, a finer mesh would do the same for both types of ABCs, however a more accurate integral evaluation for the integral ABC can provide better results for the same mesh.

The degree of improvement shown from the integral ABC to the local ABC as shown in Figs.4.12 to 4.20 has been attenuated when the expressions are incorporated to the FEM system. One hypothesis, that can be advanced as an explanation for this is that an averaging of the error or improvement is occuring over all the nodal unknowns.

The mesh size has been limited by the computer memory. The use of the integral ABC puts an extra strain on memory requirements but provides very good results with more flexibility as to the choice of outer boundary shapes and proximity to the source. With the local ABC, the mesh can extend a little further, however the accuracy of the results would still depend on the complexity of the source and the degree of variation of the field.

With the advent of ever larger and faster computers most memory and execution time considerations will become obsolete. But a global ABC such as the integral ABC or any ABC that would provide accurate results for any source with the added choice of being able to truncate the solution domain to the area of interest would always keep the edge.

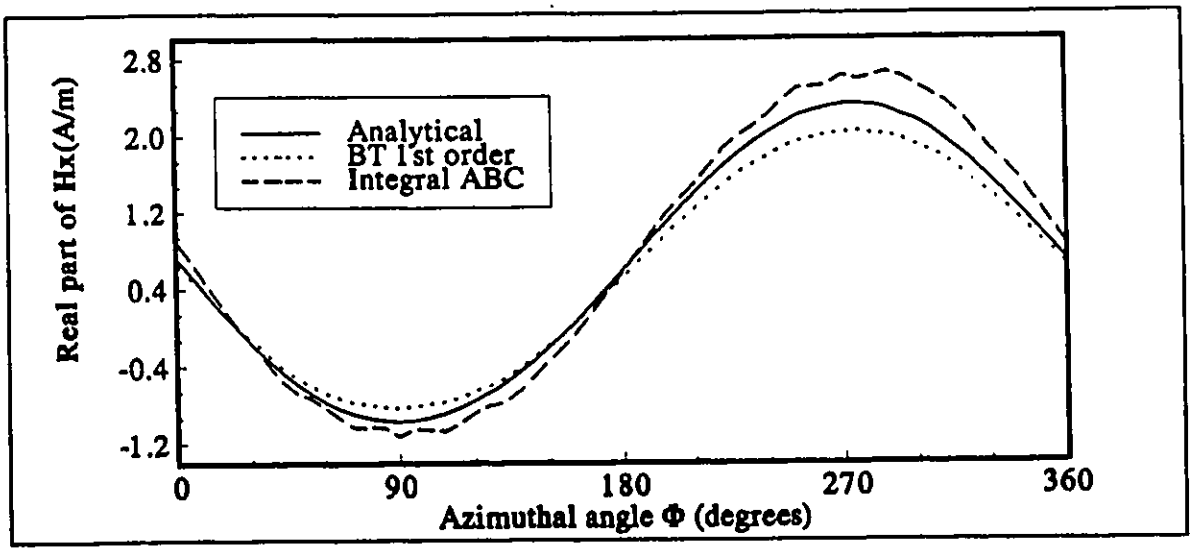

Figure 4.21: Real part of the x-component of the magnetic field at r=2.91m and $\theta = 37^0$

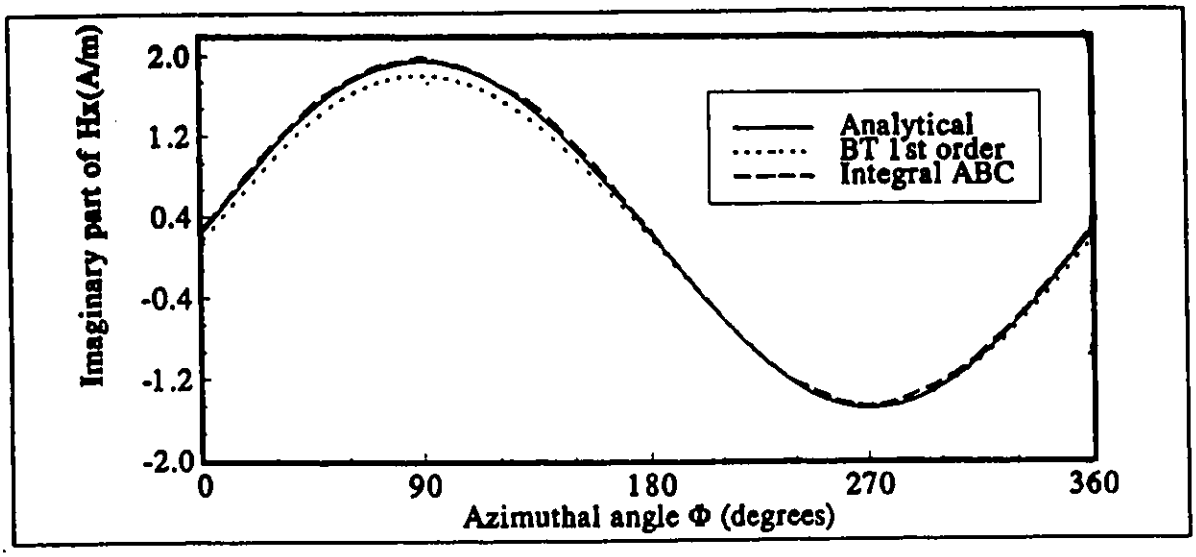

Figure 4.22: Imaginary part of the x-component of the magnetic field at r=2.91m and $\theta = 37^0$

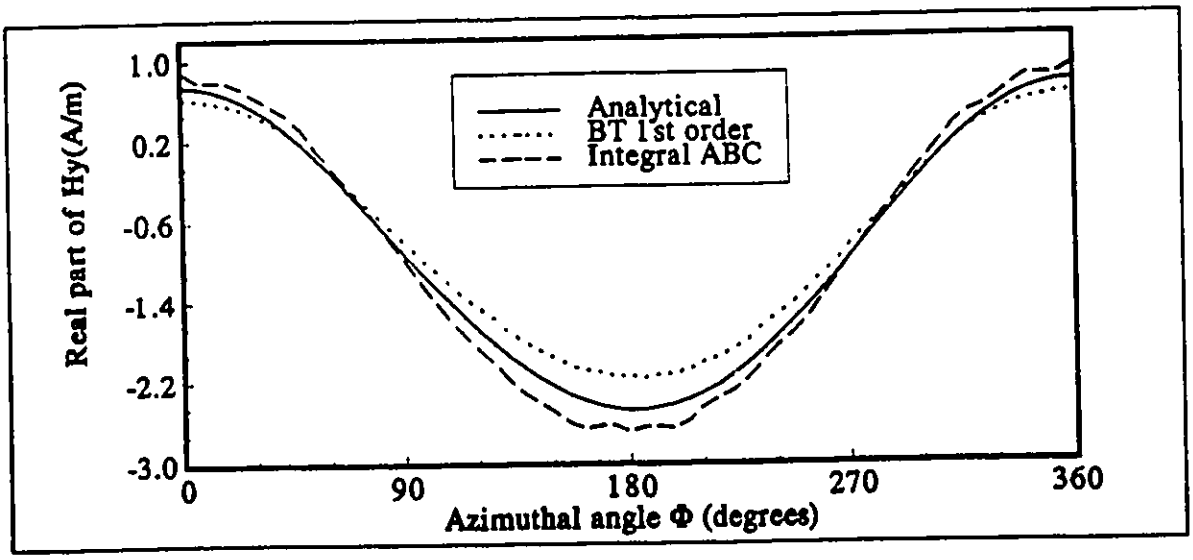

Figure 4.23: Real part of the y-component of the magnetic field at r=2.91m and $\theta = 37^0$

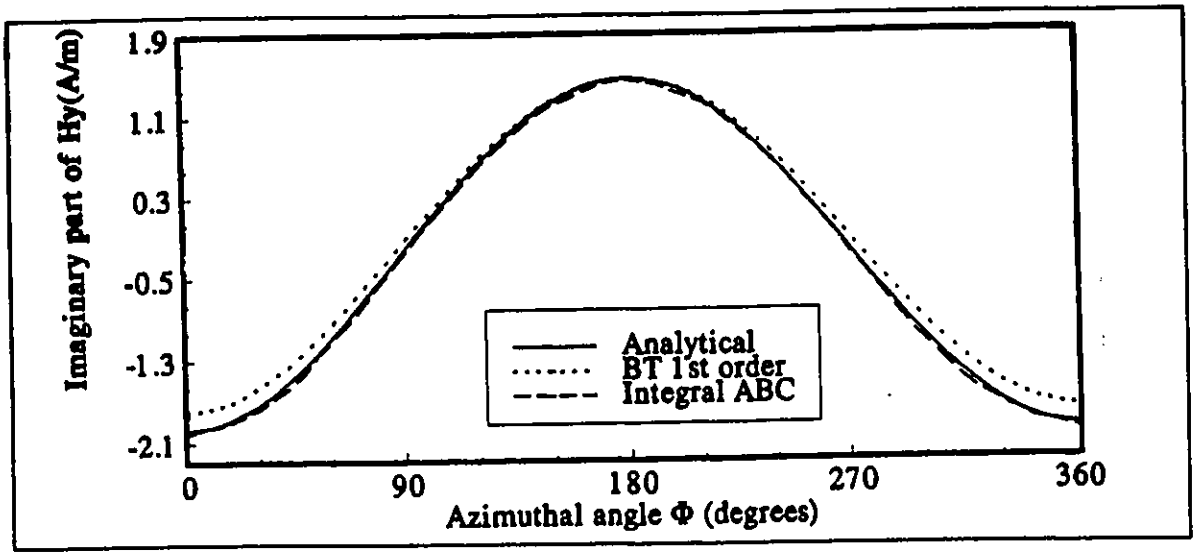

Figure 4.24: Imaginary part of the y-component of the magnetic field at r=2.91m and $\theta = 37^0$

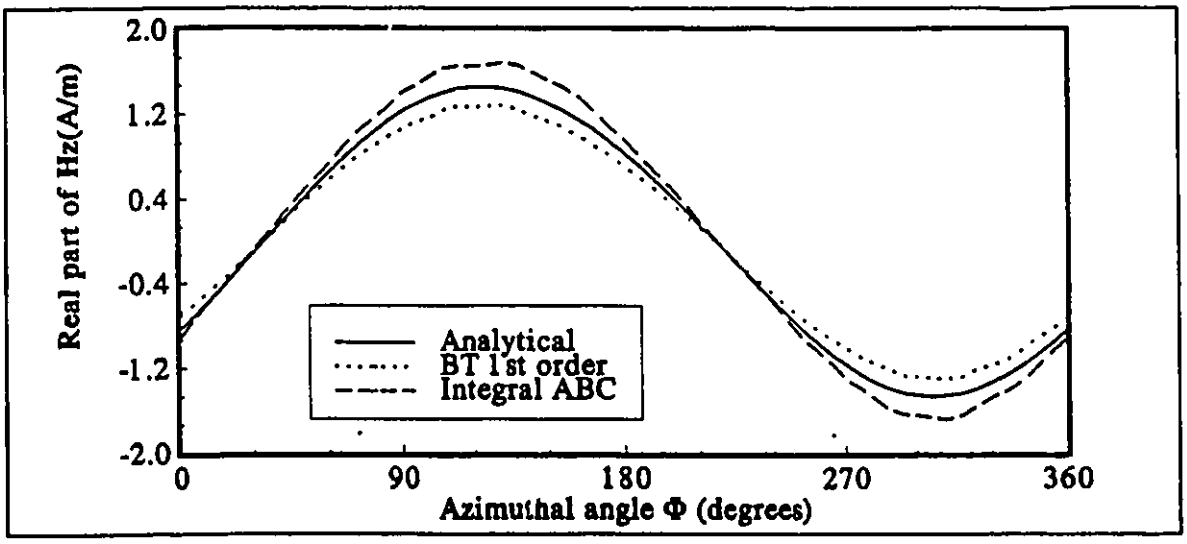

Figure 4.25: Real part of the z-component of the magnetic field at r=2.91m and $\theta = 37^0$

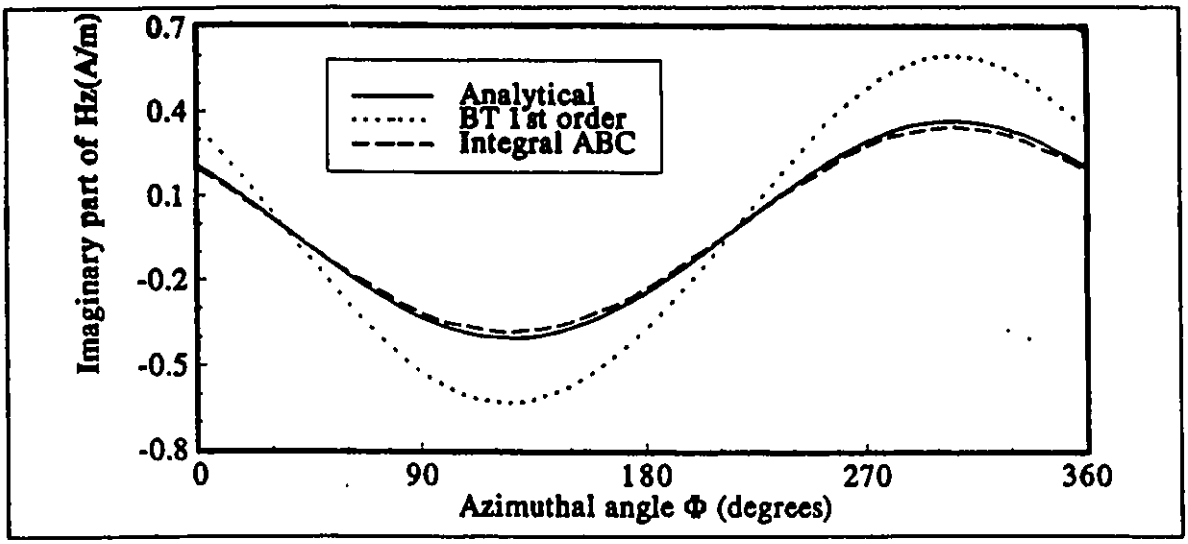

Figure 4.26: Imaginary part of the z-component of the magnetic field at r=2.91m and $\theta = 37^0$

4.6 BIM Results

As an alternative to the FEM use for the near-field to far-field transformation, the application of which faces technical limits, the integral boundary solution has been used to perform the transformation. The method provides field values at any desired range with a very short execution time. Figures 4.27 to 4.38 show the magnitude and phase of the spherical components of the magnetic field at a distance $r = 2.91m$(same as FEM figures) and 30m (far-field). The elevation angle θ was 37^0. The agreement between the analytical results and the boundary integral's is very good with maximum error in the order of 2.5%. A discrepancy in the radial component at $r = 30$m can be noticed however. It can be explained by the fact that the magnitudes involved are very small and that numerical errors due to a low resolution in the integration method over the Dirichlet boundary starts to take over. A remedy would be to increase the number of collocation points in the integral as well as the use of a finer mesh. The mesh used, as mentioned before, is similar in shape to the one depicted in Figures 4.8 and 4.9. It has however a ratio of nodes/wavelength of 42; i.e. twice that used for the FEM. Unlike the FEM, the use of the mesh does not put any restrictions on the size of the problem or the range to which far-field transformation is needed because only a thin volume surrounding the Dirichlet boundary is discretized.

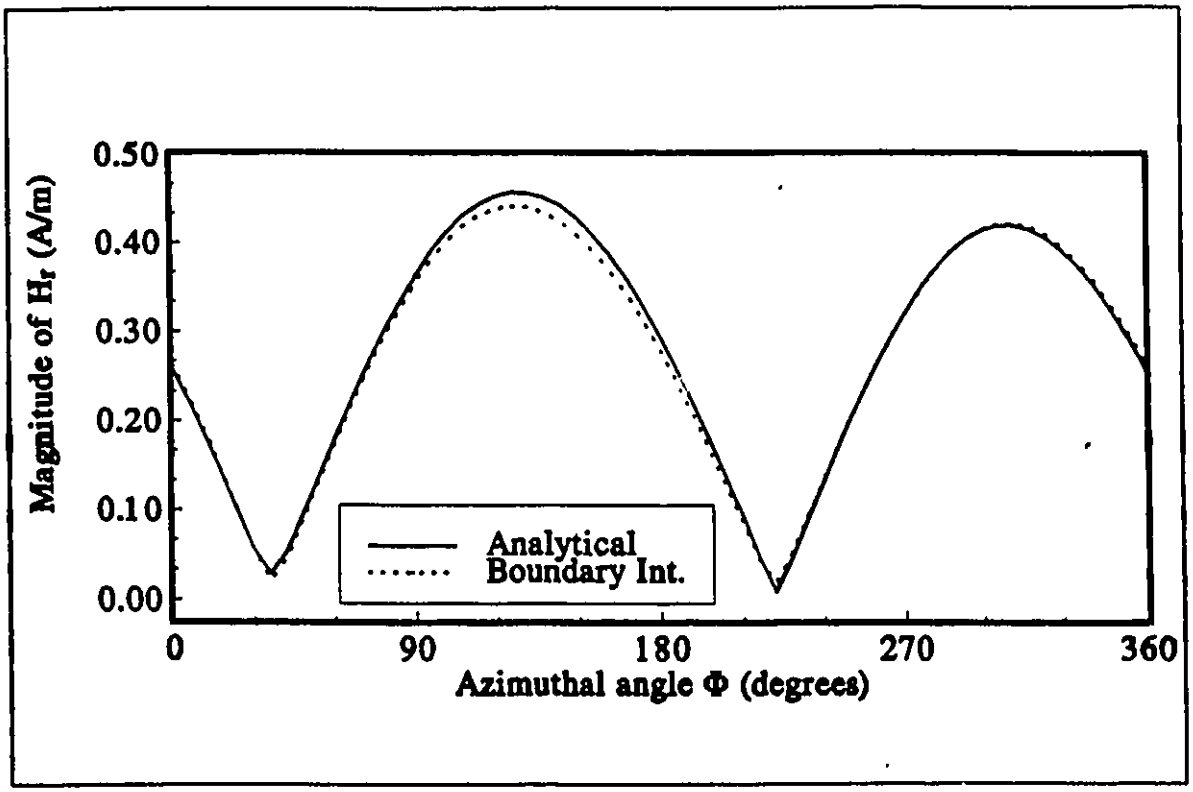

Figure 4.27: Magnitude of the r-component of the magnetic field at r=2.91m and $\theta = 37^0$

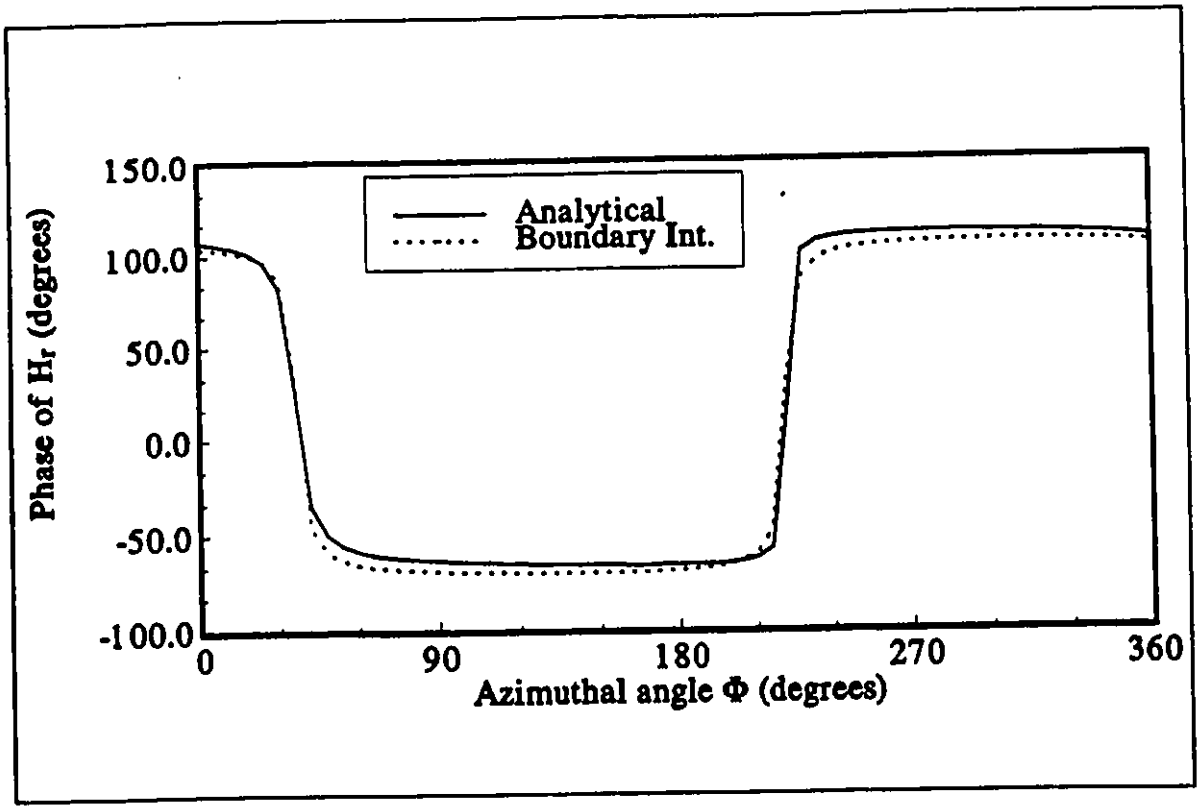

Figure 4.28: Phase of the r-component of the magnetic field at r=2.91m and $\theta = 37^0$

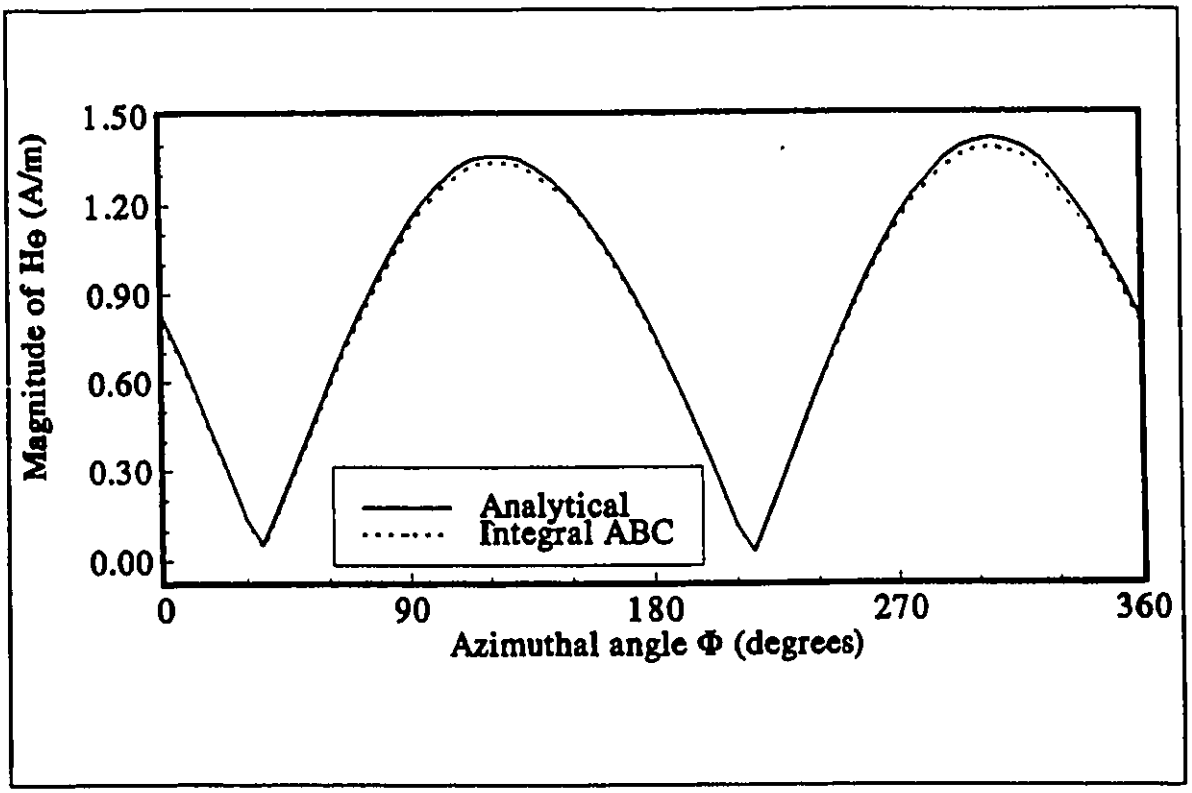

Figure 4.29: Magnitude of the θ-component of the magnetic field at r=2.91m and $\theta = 37^0$

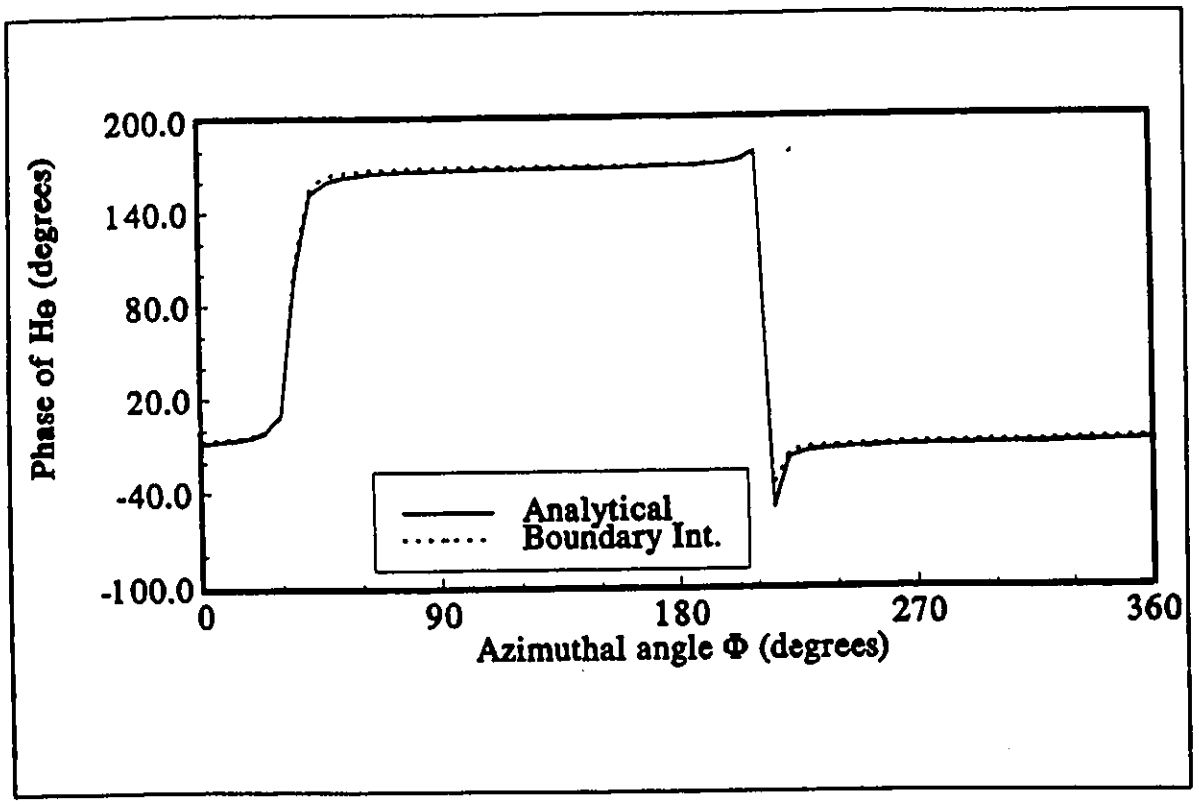

Figure 4.30: Phase of the θ-component of the magnetic field at r=2.91m and $\theta = 37^0$

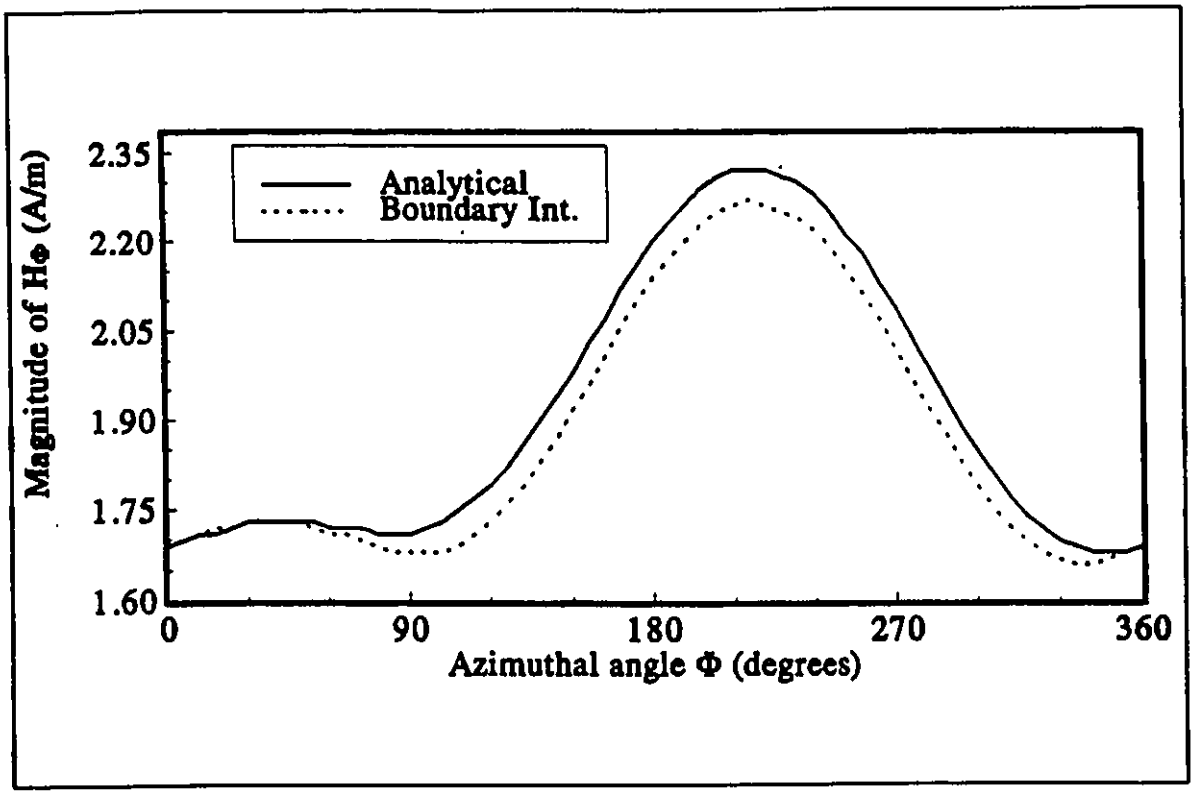

Figure 4.31: Magnitude of the ϕ-component of the magnetic field at r=2.91m and $\theta = 37^0$

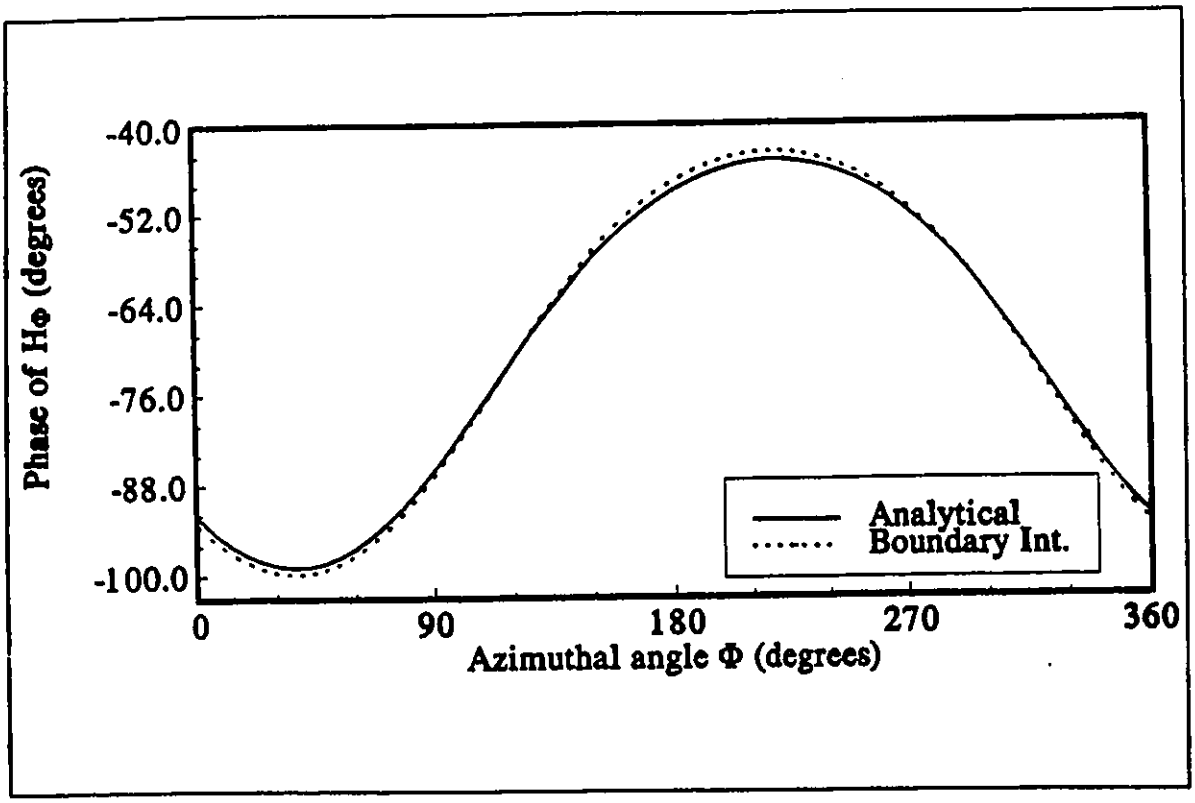

Figure 4.32: Phase of the ϕ-component of the magnetic field at r=2.91m and $\theta = 37^0$

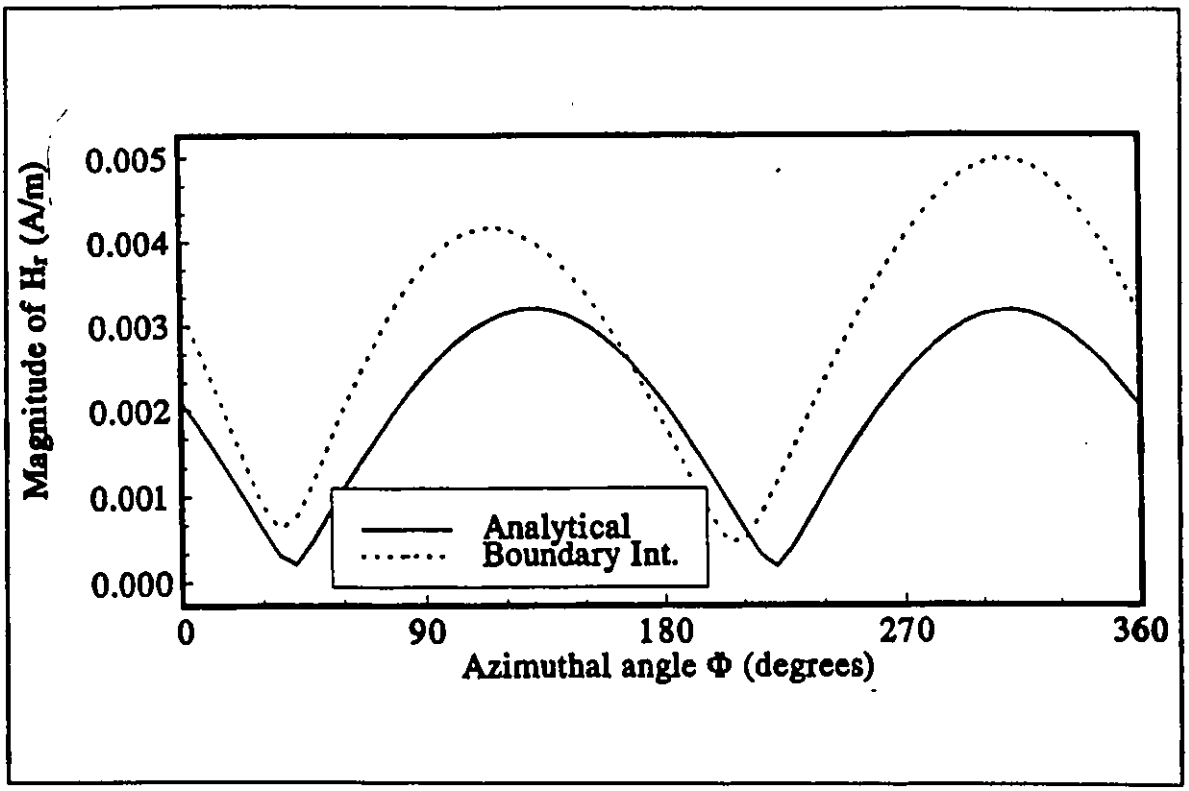

Figure 4.33: Magnitude of the r-component of the magnetic field at r=30m and $\theta = 37^o$

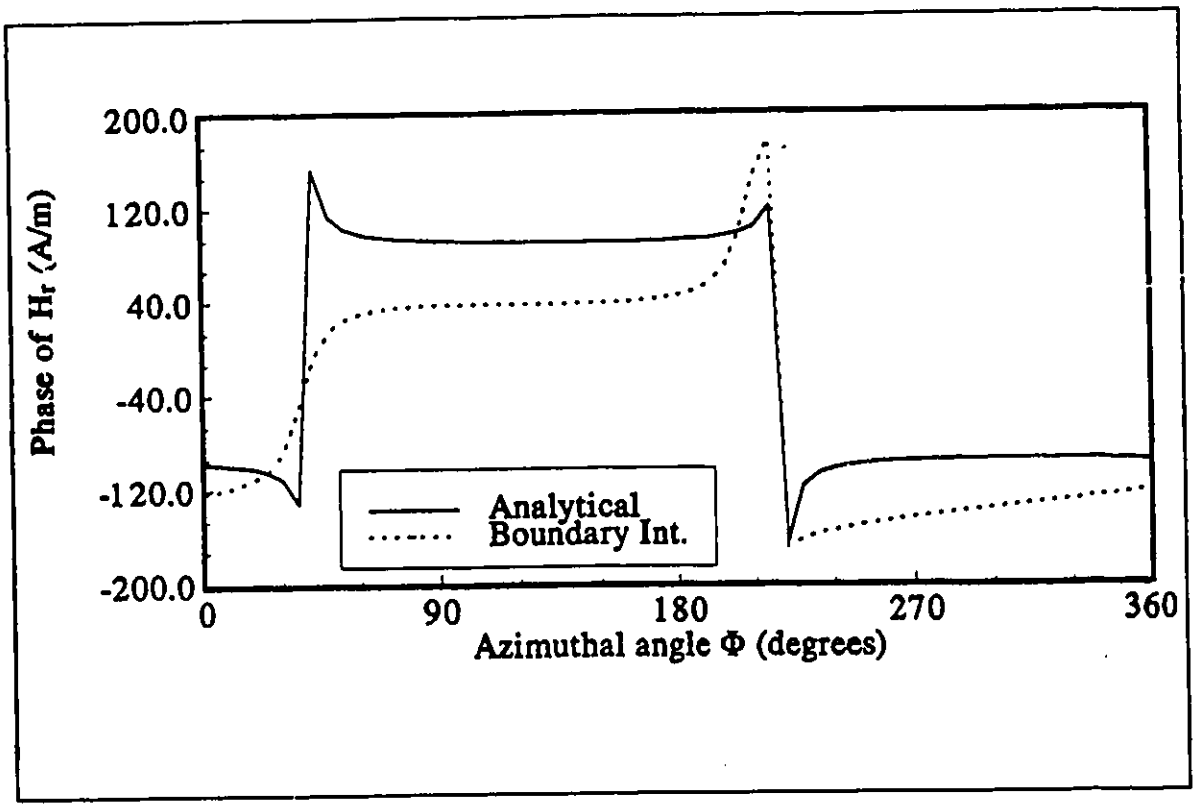

Figure 4.34: Phase of the r-component of the magnetic field at r=30m and $\theta = 37^0$

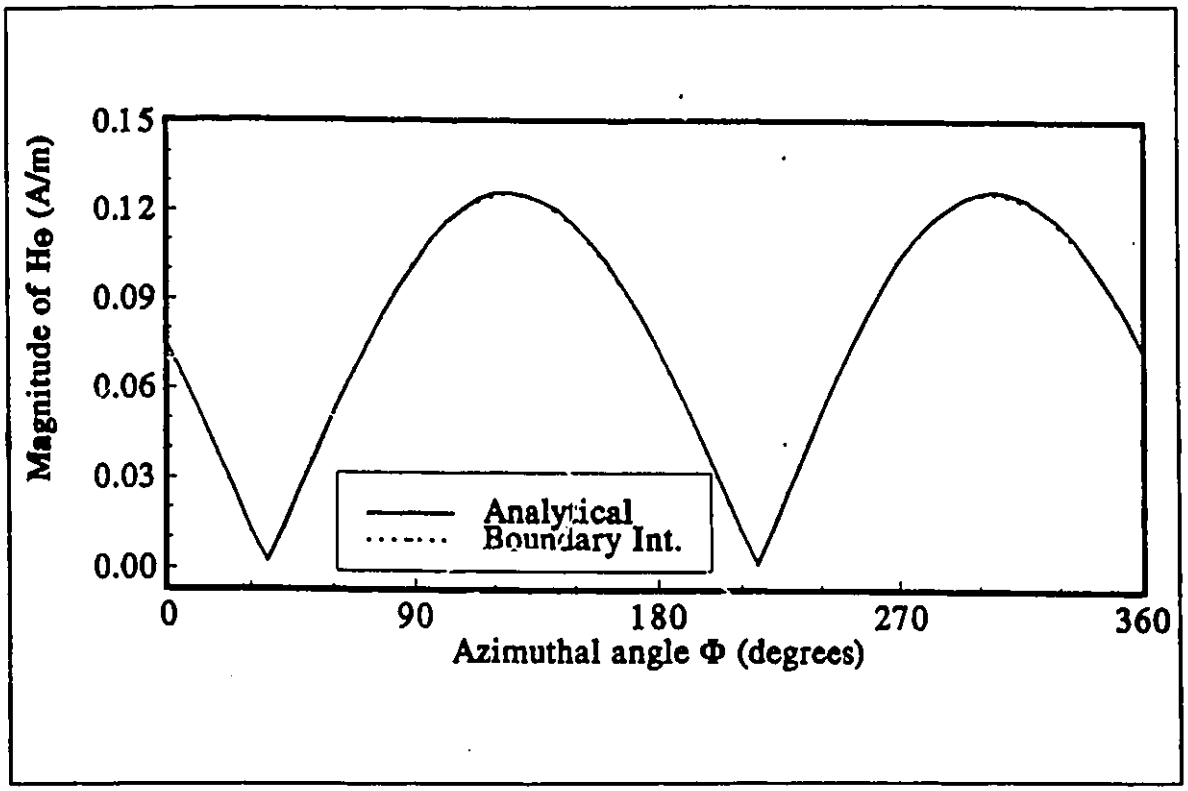

Figure 4.35: Magnitude of the θ-component of the magnetic field at r=30m and $\theta = 37^0$

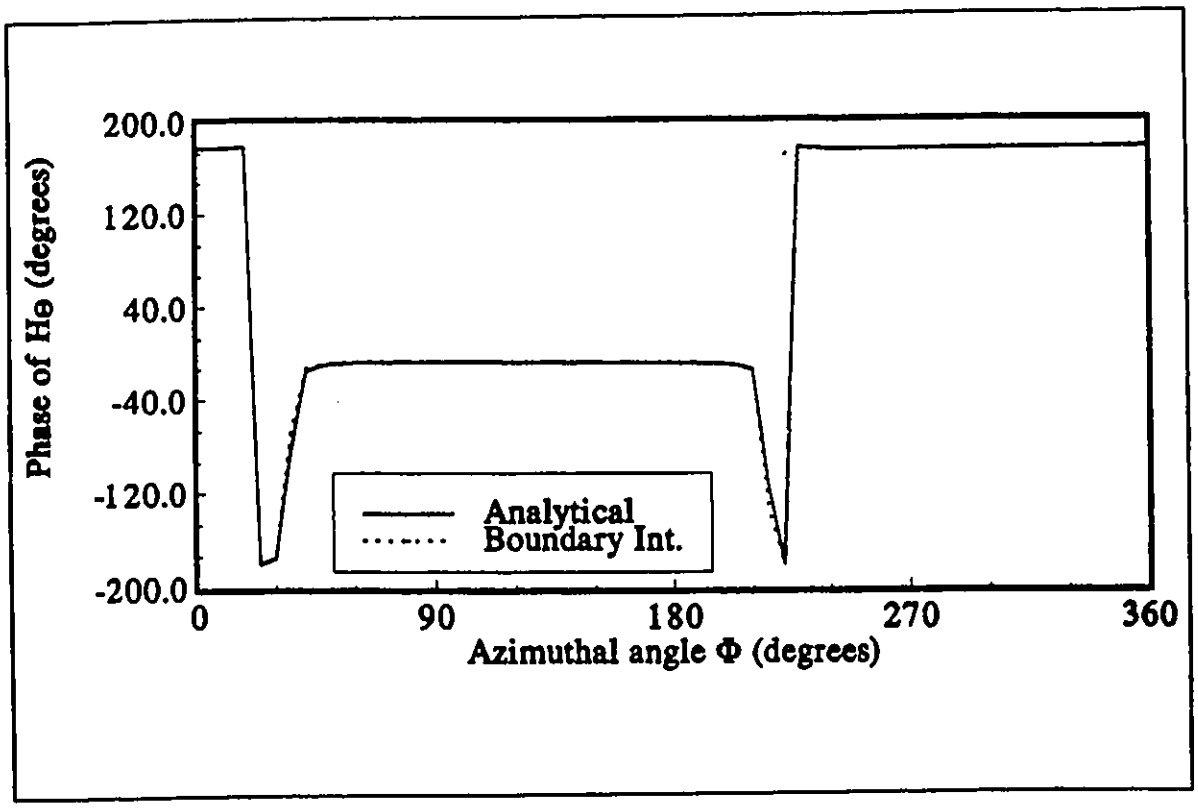

Figure 4.36: Phase of the θ-component of the magnetic field at r=30m and $\theta = 37^0$

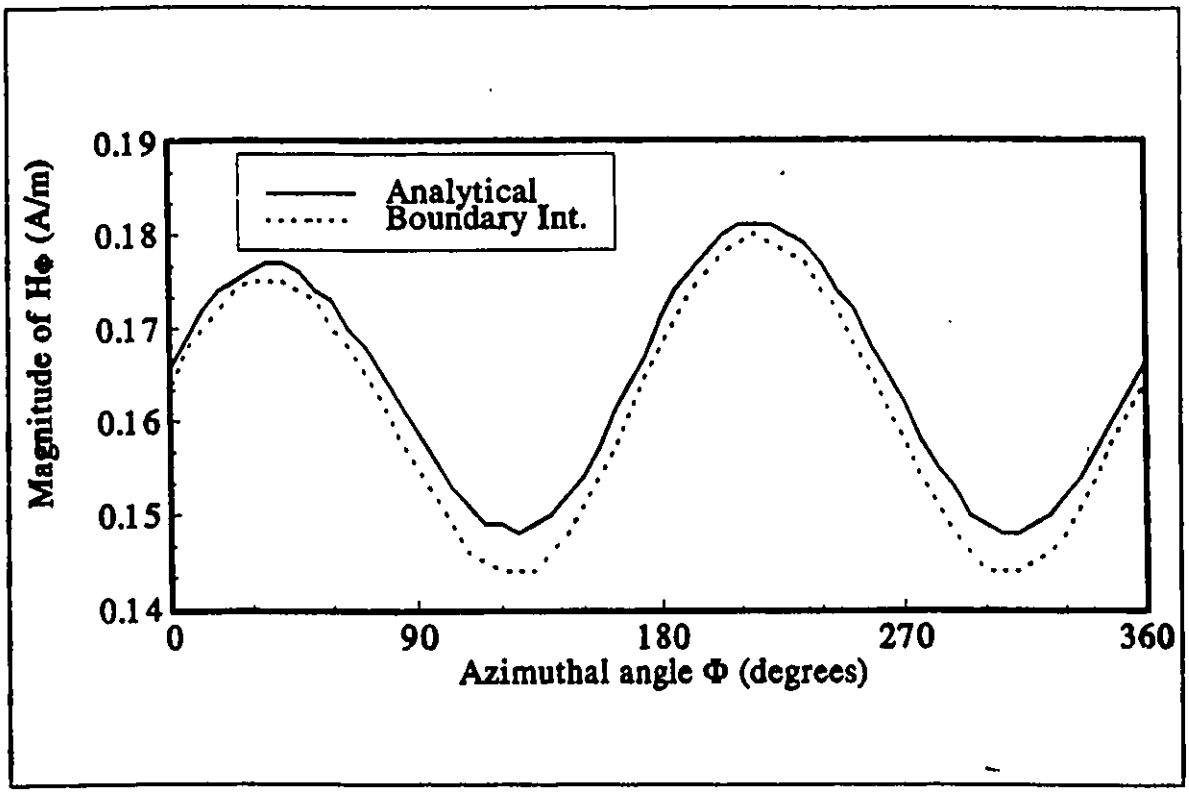

Figure 4.37: Magnitude of the ϕ-component of the magnetic field at r=30m and $\theta = 37^0$

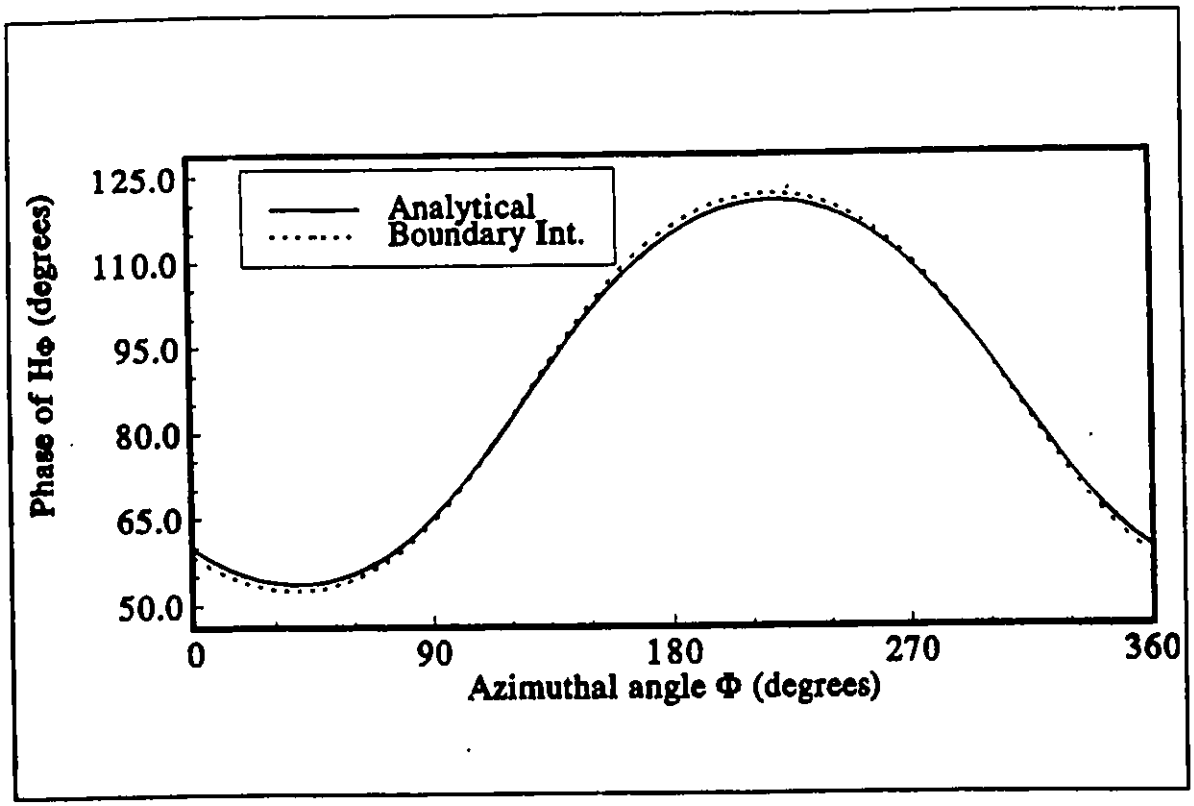

Figure 4.38: Phase of the ϕ-component of the magnetic field at r=30m and $\theta = 37^0$

Chapter 5

Conclusions

The intent of this thesis has been to derive a method to transform near-field measurements to far-field values. The problem is a three dimensional wave propagation problem. The finite element method was used first. It required the generation of a finite element mesh to discretize the 3-D solution domain and the application of absorbing boundary conditions to truncate the domain. The approach was implemented via a series of FORTRAN programs (Appendix C).
The far-field distances were not achievable with the FEM because of technical limitations. Nevertheless, it was proven that the FEM method, in spurious mode free formulation, can be used to solve 3-D wave propagation problems and eventually, hardware allowing, be used for near-field-far-field transformations with the advantage of being able to add inhomogeneities to the solution domain.

To achieve far-field distances, a second method based on a boundary integral equation solution to the scattering problem has been used for near-field to far-field transformation. This method revealed itself to be more appropriate for the problem and provided quick and accurate results. One of its drawbacks was that it required a homogeneous domain. The same boundary integral solution was then modified to derive a global ABC for the FEM solution that proved to be better than existing ABCs.

The 3-D automatic mesh generator, one of the major problems with 3-D FEM, was written to incorporate as much versatility as possible and is in fact able to discretize a wide range of shapes and forms of solution domains. It is also able to introduce inhomogeneous media in the domain. To take advantage of this and solve the wave propagation in the inhomogeneous domain, the FEM formulation in Cartesian coordinates which uncouples the components would have to be modified.

Appendix A
Vector and Integral Identities

$$\nabla \times \nabla \vec{G} = \nabla(\nabla \cdot \vec{G}) - \nabla^2 \vec{G} \tag{A1}$$

$$\nabla \cdot (f\vec{G}) = \vec{G}\nabla f + f(\nabla \cdot \vec{G}) \tag{A2}$$

$$\nabla \times (f\vec{G}) = \nabla f \times \vec{G} + f(\nabla \times \vec{G}) \tag{A3}$$

$$\nabla \times (\vec{F} \times \vec{G}) = \vec{F}(\nabla \cdot \vec{G}) - \vec{G}(\nabla \cdot \vec{F}) + (\vec{G} \cdot \nabla)\vec{F} - (\vec{F} \cdot \nabla)\vec{G} \tag{A4}$$

$$\nabla(\vec{F} \cdot \vec{G}) = (\vec{F} \cdot \nabla)\vec{G} + (\vec{G} \cdot \nabla)\vec{F} + \vec{F} \times (\nabla \times \vec{G}) + \vec{G} \times (\nabla \times \vec{F}) \tag{A5}$$

$$\vec{F} \cdot (\vec{G} \times \vec{H}) = \vec{G} \cdot (\vec{H} \times \vec{F}) = \vec{H} \cdot (\vec{H} \times \vec{G}) \tag{A6}$$

$$\int_V (\nabla \times \vec{G})\, dV = \int_S (\hat{n} \times \vec{G})\, dS \tag{A7}$$

$$\vec{A} \times (\vec{B} \times \vec{C}) = (\vec{A} \cdot \vec{C})\vec{B} - (\vec{A} \cdot \vec{B})\vec{C} \tag{A8}$$

$$\oint_S [(\hat{n} \cdot \nabla)(\phi\vec{H}) + \hat{n} \times \nabla(\phi\vec{H}) - \hat{n}\nabla \cdot (\phi\vec{H})]\, dS = 0 \tag{A9}$$

Appendix B
Analytical Integration in Simplex Coordinates

For the calculation of the contribution to the general system of equations from a single element, surface and volume integrals have to be performed over the FE in question. With the simplex coordinates, the integrals can be done analytical without introducing any numerical errors.

The 3-D domain of all the finite elements is defined by a variation of the simplex variables $\delta_1, \delta_2, \delta_3$ and δ_4 from zero to one. The integrals arising from the evaluation of the FEM system have all one of the following two forms:

$$\int_{V_e} \delta_1^i \delta_2^j \delta_3^k \delta_4^l \, dV_e \tag{.1}$$

or

$$\int_{S_e} \delta_1^i \delta_2^j \delta_3^k \delta_4^l \, dS_e \tag{.2}$$

where V_e is the volume of the e^{th} finite element and S_e is the surface of one of the four faces defining the tetrahedral FE.
Because the simplex coordinates satisfy the identity

$$\delta_1 + \delta_2 + \delta_3 + \delta_4 = 1, \tag{.3}$$

only three variables are needed to describe a volume, and only two for a surface, one remaining constant.
For V_e,

$$\delta_4 = 1 - \delta_1 + \delta_2 + \delta_3, \tag{.4}$$

and for one face of V_e (i.e. S_e), one of the simplexes will be the constant zero.

$$\delta_3 = 1 - \delta_1 + \delta_2, \quad \delta_4 = 0. \tag{.5}$$

The integrals are integrals of polynomials which have a closed form analytical expression and hence for a volume integral

$$\int_{V_e} \delta_1^i \delta_2^j \delta_3^k \delta_4^l \, dV_e = 3! V_e \left(\frac{i!j!k!l!3!}{(i+j+k+l+3)!}\right), \tag{.6}$$

and for a surface integral

$$\int_{V_e} \delta_1^i \delta_2^j \delta_3^k \, dS_e = 2! S_e \left(\frac{i!j!k!2!}{(i+j+k+2)!}\right). \tag{.7}$$

where the Jacobian has been used to change variables in the integral;

$$dV_e = 3! V_e d\delta_1 d\delta_2 d\delta_3 \tag{.8}$$

and

$$dS_e = 2! S_e d\delta_1 d\delta_2 \tag{.9}$$

The volume V_e and the surface S_e is readily calculated from the coordinates of the tetrahedron vertices. In the surface integrals the simplex variable which is constant is always dropped before evaluation.

Appendix C
The Computer Programs

All the code with the exception of the iterative solving routines has been written by the author in DEC FORTRAN77 for RISC Version 1.0 and executed on the Digital 3100 RISC DECstations(16Meg RAM, MIPS2000 CPU chip).
The first program POINTSFECREAT.F generates the list of node indexes, their coordinates and their boundary IDs to be used in the meshing program. If a mesh already exits, POINTSFECREAT can be used to generate the same list of points as was used for the mesh but modify the boundary IDs. This permits the user to have similar meshed structures but with different boundary conditions. The new boundary conditions are implemented in the mesh by running PROPERTIES.F. The mesh itself is generated by running MESHMAIN.F. Before a FEM problems is solved, all the Dirichlet and Neumann boundary values have to be defined according to the data in the FE mesh. This is done by running the program PREPROS.F. The system is solved then by running MAINCOMPSG.F which will ask all kind of questions about the type of ABC to be used, the maximum number of iterations for the iterative method, the maximum error tolerance and so on. The non-zero entries in the system solved is stored in one large column vector. If the number of non-zero entries in each individual column in the system is known it will speed up the building up of the system. For the local ABC, this number is around 70. If it is not known, it is good to overshoot if the computer memory allows it, otherwise it's a good idea to store this filling pattern for speedier subsequent runs. The program RESPROS.F will process the results and provide field values at the desired coordinates.

The program BEMRAD.F is based on the boundary integral method. It uses a smaller mesh and provides field results at any range. The mesh can be the same as the one used for FEM. There are bem versions of PROPERTIES.F, PREPROS.F and POINTSFECREAT.F that have to be rerun on the FEM mesh. They are not listed since they are very similar, with minor variations in the boundary conditions handling.

ANALYTICRAD.F and DERANALRAD.F are the two programs that give the analytical results for comparison. ANALYTICRAD.F gives the values of the field at any desired position and DERANALRAD.F gives the derivative of the field in the radial direction. The programs listed after DERANALRAD.F are listings of subroutines called by the main programs mentioned above. The code has been written with as many subroutines as possible with descriptive subroutine names. Further documentation is included in the code listings

POINTSFECREAT.F

```
CPROGRAM THAT GENERATES TTHE POINTS TO BE USED BY THE MESH GENERATOR
CAND THE VARIOUS MESH MODIFIERS
CTHE ID CODE OF EVERY NODE IS DEFINED AS FOLLOWS
CCODED BY POSITION  NXABCD
CX=1 OUTER BOUNDARY
CX=0 INNER BOUNDARY(NO SPECIAL PROPERTY)
CA OR B OR C=1 DIRICHLET
CA OR B OR C=2 NEUMANN
CA OR B OR C=3 UNKNOWN
CA OR B OR C=4 ON THE BOUNDARY OF A DIRICHLET AND A NEUMANN
CA FOR X COMPONENT
CB FOR Y COMPONENT
CC FOR Z COMPONENT
CD DIGIT CODE FOR THE MEDIUM USUALLY ONE FOR ALL HOMOGENEOUS
CN=5 NODE LYING ON A SURFACES NEEDED FOR THE INTEGRAL ABC
CNPS IS THE NUMBER OF NODES ON THE DIRICHLET SURFACE
CNPV IS THE TOTAL NUMBER OF NODES
C
IMPLICIT NONE
REAL P(41000,3)
INTEGER NPS,NPV,PID(41000),NP,I
C
NP=0
CALL SURFACEPOINTGEN(NP,P,PID,1,1,2,0.5,0.5,0.5)
NPS=NP
CALL EXTRAPOINTS(NP,P,PID,1,1,2,3,3,4,0.5,0.5,0.5)
NPV=NP
CALL WRITER(NPS,NPV,P,PID)
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT GENERATES THE POINTS OFF THE DIDIRCHLET SURFACE.
CDEPENDING ON THE DIFFERENT IF STATEMENTS THE LAST OR ALL LAYERS ARE
CMAPPED ON A SHERICAL DOME
SUBROUTINE EXTRAPOINTS(NP,P,PID,LX1,LY1,LZ1,LX,LY,LZ,DX,DY,DZ)
INTEGER NP,PID(41000),LX,LY,LZ,LX1,LY1,LZ1
REAL P(41000,3),DX,DY,DZ
C
```

```
IMPLICIT NONE
REAL X,Y,Z,R,SX,SY,SZ
INTEGER I,J,K,RX,RY,RZ
LOGICAL LAST1,LAST,RLAST,LAY1
DO 10 I=-LX,LX
  DO 20 J=-LY,LY
    DO 30 K=0,LZ
RX=ABS(I)
RY=ABS(J)
RZ=ABS(K)
IF ((RX.GE.RY).AND.(RX.GE.RZ))  THEN
  RY=RX
  RZ=RX+1
ELSE IF ((RY.GE.RX).AND.(RY.GE.RZ)) THEN
  RX=RY
  RZ=RY+1
ELSE IF ((RZ.GE.RY).AND.(RZ.GE.RX)) THEN
  RX=RZ-1
  RY=RZ-1
      END IF
     R=SQRT((RX*DX)**2+(RY*DY)**2+(RZ*DZ)**2)
IF ((ABS(I).LE.LX1).AND.(ABS(J).LE.LY1).AND.(ABS(K).LE.LZ1)) GOTO 30
   RLAST=(ABS(J).EQ.LY).OR.(ABS(I).EQ.LX).OR.(K.EQ.LZ)
   LAST=(ABS(J).GT.LY1).OR.(ABS(I).GT.LX1).OR.(K.GT.LZ1)
c    last like this will convert all layers into domes
   LAY1=(ABS(J).GE.(LY-1)).OR.(ABS(I).GE.(LX-1)).OR.(K.GE.(LZ-1))
   LAST1=(LAY1.AND.(.NOT.(RLAST)))
    NP=NP+1
    X=I*DX
    Y=J*DY
    Z=DZ*K
    P(NP,1)=X
    P(NP,2)=Y
      P(NP,3)=Z
    PID(NP)=1
c     IF (LAST) THEN
    IF (RLAST) THEN
       CALL CONVCARTOSFE(X,Y,Z,SX,SY,SZ,R)
       P(NP,1)=SX
       P(NP,2)=SY
       P(NP,3)=SZ
IF (RLAST) PID(NP)=10001
c IF (RLAST) PID(NP)=11111
    END IF
    IF (K.EQ.0) PID(NP)=12211
c     IF ((K.EQ.0).AND.RLAST) PID(NP)=14411
    IF (LAST1) PID(NP)=PID(NP)+500000
30     CONTINUE
20   CONTINUE
10  CONTINUE
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CONVCARTOSFE(CX,CY,CZ,SX,SY,SZ,R)
REAL CX,CY,CZ,SX,SY,SZ,R
C
IMPLICIT NONE
REAL TET,FI,R1
C
 R1=SQRT(CX**2+CY**2+CZ**2)
 TET=ACOS(CZ/R1)
 IF (R1.EQ.0) TET=0.0
 FI=ATAN(CY/CX)
 IF (CX.EQ.0) FI=0.0
 IF ((CX.EQ.0).AND.(CY.GT.0)) FI=1.570796327
 IF ((CX.EQ.0).AND.(CY.L.'.0)) FI=-1.570796327
 IF ((CX.GT.0).AND.(CY.EQ.0)) FI=0.0
 IF ((CX.LT.0).AND.(CY.EQ.0)) FI=3.1415926
 IF ((CX.LT.0).AND.(CY.LT.0)) FI=FI+3.1415926
 IF ((CX.LT.0).AND.(CY.GT.0)) FI=FI+3.1415926
 SX=R*SIN(TET)*COS(FI)
 SY=R*SIN(TET)*SIN(FI)
 SZ=R*COS(TET)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT STORES THE POINTS GENERATED. IF THE POINTS ARE INTENDED FOR
CA NEW MESH unpoint.dat  HAS TO BE COPIED INTO unmesh.dat
SUBROUTINE WRITER(NPS,NPV,P,PID)
INTEGER NPS,NPV,PID(41000)
REAL P(41000,3)
C
IMPLICIT NONE
INTEGER REK,OFFREC,I,SCP,SCF,SCE
C
OPEN(UNIT=2,FILE='point.dat',STATUS='UNKNOWN',ACCESS='SEQUENTIAL',
  *      FORM='FORMATTED')
OPEN(UNIT=3,FILE='unpoint.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
  *      FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
C
SCP=0
SCF=0
SCE=0
WRITE(2,FMT=100) NPS,'    THE NUMBER OF POINTS ON SURFACES'
WRITE(2,FMT=100) NPV,'    THE NUMBER OF ALL POINTS'
WRITE(2,FMT=100) SCP,'     THE NUMBER OF PYRAMIDS'
WRITE(2,FMT=100) SCF,'     THE NUMBER OF FACES'
WRITE(2,FMT=100) SCE,'     THE NUMBER OF EDGES'
WRITE(2,FMT=120) '---POINTS ON SURFACE----'
WRITE(3,REC=1) NPS
WRITE(3,REC=2) NPV
WRITE(3,REC=3) SCP
WRITE(3,REC=4) SCF
WRITE(3,REC=5) SCE
c goto 1000
OFFREC=5
DO 10 I=1,NPS
  REK=OFFREC+I
WRITE(2,FMT=110) I,P(I,1),P(I,2),P(I,3),PID(I)
10 WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
OFFREC=REK
```

```
WRITE(3,FMT=120) '----INSIDE SURFACE----'
OFFREC=OFFREC-NPS
DO 20 I=NPS+1,NPV
  REK=OFFREC+I
WRITE(3,FMT=110) I,P(I,1),P(I,2),P(I,3),PID(I)
20 WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
C
C
100  FORMAT(I8,(A))
110 FORMAT(X,I5,3X,F9.5,3X,F9.5,3X,F9.5,3X,I7)
120 FORMAT((A))
130 FORMAT(X,F9.5,3X,F9.5,3X,F9.5,3X,I5,2X,I5,2X,I5)
1000 CLOSE(3)
CLOSE(3)
END
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT GENERATES THE POINTS ON THE DIRICHLET SURFACE
SUBROUTINE SURFACEPOINTGEN(NP,P,PID,LX,LY,LZ,DX,DY,DZ)
INTEGER NP,PID(41000),LX,LY,LZ
REAL P(41000,3),DX,DY,DZ
C
C ON THE CUBE
IMPLICIT NONE
INTEGER I,J
DO 10 I=-LX,LX
  DO 20 J=-LY,LY
    NP=NP+1
    P(NP,1)=I*DX
    P(NP,2)=J*DY
    P(NP,3)=LZ*DZ
20      PID(NP)=11111
10 CONTINUE
C
DO 30 I=-LY,LY
  DO 40 J=0,(LZ-1)
    NP=NP+1
    P(NP,1)=LX*DX
    P(NP,2)=I*DY
    P(NP,3)=J*DZ
    PID(NP)=11111
    IF (J.EQ.0) PID(NP)=14411
    NP=NP+1
    P(NP,1)=-LX*DX
    P(NP,2)=I*DY
    P(NP,3)=J*DZ
    PID(NP)=11111
    IF (J.EQ.0) PID(NP)=14411
40   CONTINUE
30 CONTINUE
C
DO 50 I=-(LX-1),(LX-1)
  DO 60 J=0,(LZ-1)
    NP=NP+1
    P(NP,1)=I*DX
    P(NP,2)=-LY*DY
    P(NP,3)=J*DZ
    PID(NP)=11111
    IF (J.EQ.0) PID(NP)=14411
    NP=NP+1
    P(NP,1)=I*DX
    P(NP,2)=LY*DY
    P(NP,3)=J*DZ
    PID(NP)=11111
    IF (J.EQ.0) PID(NP)=14411
60   CONTINUE
50 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

MESHMAIN.F

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C PROGRAM   CONNECTS POINTS IN A MESH CREATE TETRAHEDRAL FINITE      C
C ELEMENTS    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ANY TIME YOU CHANGE THE GEOMETRY OF THE PROBLEM YOU HAVE TO
C CHANGE THE PARAMTERS IN CENTERVALIDS AND CENTERVALIDV OR THEIR
C CALLS
C
IMPLICIT NONE
INTEGER FACE(60000,4),EDGE(60000,3),PYR(40000,5),NP,SCF,SCE,SCP
INTEGER EXTRAPYR(40000,5),PID(30000),NPS,NPV,EPV
REAL P(30000,3)
CHARACTER*24 fdate
EXTERNAL fdate
C
C
SCF=0
SCE=0
SCP=0
NP=0
WRITE(*,*) fdate()
CALL POINTREADER(NPS,NPV,PID,P)
WRITE(*,*) 'I JUST READ THE POINTS'
CALL FACECREATIONSURFACE(NP,NPS,SCF,SCE,P,PID,EDGE,FACE)
WRITE(*,*) 'I JUST CREATED THE SURFACES'
NPV=NPS+NPV
CALL PYRAMIDGENERATOR(NP,NPV,SCF,SCE,SCP,P,PID,EDGE,FACE,PYR)
C
WRITE(*,*) 'I JUST CREATED PYRAMIDS'
NP=NPV
CALL SCNDORDER(SCE,SCP,EDGE,FACE,PYR,EXTRAPYR)
WRITE(*,*) 'I JUST CREATED THE 6 EXTRA POINTS'
CALL MESHWRITER(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,EXTRAPYR)
WRITE(*,*) 'I M DONE.'
WRITE(*,*) fdate()
C
END
```

PROPERTIES.F

```
C PROGRAM THAT MODIFIES THE PROPERTIES OF FACES IN AN ALREADY
C MESHED VOLUME. CREATES NEW POINTS ON ALL EDGES AND STORES THE
C  ONE ON BOUNDARY CONDITIONS IN dirichletz.dat
C PROGRAM WORKS SUCCESSFULLY ONLY IF THE NUMBER OF NEW POINTS IS
C THE SAME AS THE NUMBER OF OLD POINTS AND THE COORDINATE TRANSFORMATION
C DOES NOT CREATE OVERLAPPING FINITE ELEMENTS
C
IMPLICIT NONE
C
INTEGER NP,SCP,SCF,SCE,PID(30000),EDGE(80000,3),FACE(68000,4),NPS
INTEGER PYR(40000,6),EXTRAPYR(40000,6),FIL,NPS1,NP1,I,PIDN(21000,4)
INTEGER EDGEID,FACEID,PYRID,IND(4),OCCUR,PYRCOM,PIDD(21000,4),NN,ND
INTEGER I1,I2,I3,I4
REAL P(30000,3),DUM1(3),DUM2(3),DUM(3)
C
OPEN(UNIT=4,FILE='dirichletz.dat',STATUS='UNKNOWN',
   *      ACCESS='SEQUENTIAL',FORM='FORMATTED')
CALL GETBASICINFO(NPS,NP,SCE,SCF,SCP,FIL)
CALL MESHREADER(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,EXTRAPYR,FIL)
CALL NEWPOINTREADER(NPS1,NP1,PID,P)
IF ((NPS.NE.NPS1).OR.(NP.NE.NP1)) THEN
  WRITE(*,*) 'YOU RE ABOUT TO SCREWUP THE OLD MESH !'
  WRITE(*,*) 'I M GONNA SAVE THE DAY!'
  GOTO 100
END IF
DO 20 I=1,SCF
  CALL BASE3NODE(EDGE,FACE,I,IND(1),IND(2),IND(3))
  CALL OCCURENCEP(I,SCP,OCCUR,PYR,PYRCOM)
  FACE(I,4)=FACEID(PID(IND(1)),PID(IND(2)),PID(IND(3)))
20 CONTINUE
DO 10 I=1,SCE
  IND(1)=EDGE(I,1)
  IND(2)=EDGE(I,2)
  EDGE(I,3)=EDGEID(PID(IND(1)),PID(IND(2)))
  NP1=NP1+1
  CALL ONEDIM(P(IND(1),1),P(IND(1),2),P(IND(1),3),DUM1)
  CALL ONEDIM(P(IND(2),1),P(IND(2),2),P(IND(2),3),DUM2)
  CALL MIDPOI(DUM1,DUM2,DUM)
  P(NP1,1)=DUM(1)
  P(NP1,2)=DUM(2)
  P(NP1,3)=DUM(3)
  PID(NP1)=EDGE(I,3)
10 CONTINUE
CALL GETPIDW(PID,PIDD,PIDN,NP1,NN,ND)
WRITE(4,*) ND
DO 50 I=1,ND
  I1=PIDD(I,1)
  I2=PIDD(I,2)
  I3=PIDD(I,3)
  I4=PIDD(I,4)
50   WRITE(4,130) P(I1,1),P(I1,2),P(I1,3),I2,I3,I4
WRITE(4,*) NN
DO 40 I=1,NN
  I1=PIDN(I,1)
  I2=PIDN(I,2)
  I3=PIDN(I,3)
  I4=PIDN(I,4)
40   WRITE(4,130) P(I1,1),P(I1,2),P(I1,3),I2,I3,I4
C
DO 30 I=1,SCP
  CALL PYR4NODE(EDGE,FACE,PYR,I,IND)
30 PYR(I,5)=PYRID(PID(IND(1)),PID(IND(2)),PID(IND(3)),PID(IND(4)))
CALL MESHWRITWO(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,EXTRAPYR)
100 CONTINUE
130 FORMAT(X,F9.5,3X,F9.5,3X,F9.5,3X,I5,2X,I5,2X,I5)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT READS THE SIZE OF THE ALREADY EXISTING MESH
SUBROUTINE GETBASICINFO(NPS,NP,SCE,SCF,SCP,FIL)
INTEGER NP,SCE,SCF,SCP,FIL,NPS
C
IMPLICIT NONE
OPEN(UNIT=2,FILE='unmesh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
   *         FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
READ(2,REC=1) NPS
READ(2,REC=2) NP
READ(2,REC=3) SCP
READ(2,REC=4) SCF
READ(2,REC=5) SCE
FIL=2
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT READS THE MESH POINTS' NEW COORDINATE AND PROPERTY IDS
SUBROUTINE NEWPOINTREADER(NPS,NP,PID,P)
INTEGER NPS,NP,PID(30000)
REAL P(30000,3)
C
IMPLICIT NONE
INTEGER REK,OFFREC,I,SCP,SCF,SCE
C
OPEN(UNIT=3,FILE='unpoint.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
   *      FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
C
READ(3,REC=1) NPS
READ(3,REC=2) NP
READ(3,REC=3) SCP
READ(3,REC=4) SCF
READ(3,REC=5) SCE
OFFREC=5
DO 10 I=1,NPS
  REK=OFFREC+I
10 READ(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
OFFREC=REK
OFFREC=OFFREC-NPS
DO 20 I=NPS+1,NP
  REK=OFFREC+I
20 READ(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
C
```

```
      CLOSE(3)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT UPDATES THE MESH
      SUBROUTINE MESHWRITWO(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,EXTRAPYR)
      INTEGER NP,SCP,SCF,SCE,PID(30000),EDGE(60000,3),FACE(65000,4),NPS
      INTEGER PYR(40000,5),EXTRAPYR(40000,5)
      REAL P(30000,3)
C
      IMPLICIT NONE
        INTEGER OFFREC,REK,I,IN1,IN2,IN3
      OPEN(UNIT=1,FILE='edge.dat',STATUS='UNKNOWN')
      OPEN(UNIT=2,FILE='mesh.dat',STATUS='UNKNOWN',ACCESS='SEQUENTIAL',
     *       FORM='FORMATTED')
      OPEN(UNIT=3,FILE='unmesh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
     *       FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
      WRITE(2,FMT=100) NPS,'   THE NUMBER OF POINTS ON THE SURFACE'
      WRITE(2,FMT=100) NP,'    THE NUMBER OF ALL POINTS'
      WRITE(2,FMT=100) SCP,'    THE NUMBER OF PYRAMIDS'
      WRITE(2,FMT=100) SCF,'    THE NUMBER OF FACES'
      WRITE(2,FMT=100) SCE,'    THE NUMBER OF EDGES'
      WRITE(2,FMT=120) '---ON THE SURFACE---'
      WRITE(3,REC=1) NPS
      WRITE(3,REC=2) NP
      WRITE(3,REC=3) SCP
      WRITE(3,REC=4) SCF
      WRITE(3,REC=5) SCE
      OFFREC=5
      DO 60 I=1,NPS
        REK=OFFREC+I
        WRITE(2,110) P(I,1),P(I,2),P(I,3),PID(I)
   60   WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
      OFFREC=REK
      WRITE(2,FMT=120) '---INSIDE SURFACE---'
      DO 70 I=NPS+1,NP
        REK=OFFREC+I-NPS
        WRITE(2,110) P(I,1),P(I,2),P(I,3),PID(I)
   70   WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
      WRITE(2,FMT=120) '---EDGES---'
      OFFREC=REK
      DO 20 I=1,SCE
        REK=OFFREC+I
      WRITE(2,FMT=105) I,EDGE(I,1),EDGE(I,2),EDGE(I,3)
   20 WRITE(3,REC=REK) EDGE(I,1),EDGE(I,2),EDGE(I,3)
      OFFREC=REK
      WRITE(2,FMT=120) '--FACES---'
      DO 30 I=1,SCF
        REK=OFFREC+I
      WRITE(2,FMT=115) I,FACE(I,1),FACE(I,2),FACE(I,3),FACE(I,4)
   30 WRITE(3,REC=REK) FACE(I,1),FACE(I,2),FACE(I,3),FACE(I,4)
      OFFREC=REK
      WRITE(2,FMT=120) '--PYRAMIDS---'
      DO 40 I=1,SCP
        REK=OFFREC+I
      WRITE(2,FMT=130) PYR(I,1),PYR(I,2),PYR(I,3),PYR(I,4),
     *       PYR(I,5),EXTRAPYR(I,1),EXTRAPYR(I,2),EXTRAPYR(I,3),
     *       EXTRAPYR(I,4),EXTRAPYR(I,5),EXTRAPYR(I,6)
   40 WRITE(3,REC=REK) PYR(I,1),PYR(I,2),PYR(I,3),PYR(I,4),
     *       PYR(I,5),EXTRAPYR(I,1),EXTRAPYR(I,2),EXTRAPYR(I,3),
     *       EXTRAPYR(I,4),EXTRAPYR(I,5),EXTRAPYR(I,6)
C
      WRITE(1,*) SCF
c     DO 50 I=1,SCF
c       CALL BASE3NODE(EDGE,FACE,I,IN1,IN2,IN3)
c       WRITE(1,*) P(IN1,1)
c       WRITE(1,*) P(IN1,2)
c       WRITE(1,*) P(IN1,3)
c       WRITE(1,*) P(IN2,1)
c            WRITE(1,*) P(IN2,2)
c       WRITE(1,*) P(IN2,3)
c       WRITE(1,*) P(IN3,1)
c       WRITE(1,*) P(IN3,2)
c50     WRITE(1,*) P(IN3,3)
C
   90 FORMAT(<NP>(/))
  100  FORMAT(I8,(A))
  110 FORMAT(3X,F9.5,3X,F9.5,3X,F9.5,3X,I7)
  105 FORMAT(X,I6,3X,I6,3X,I6,3X,I6)
  115 FORMAT(X,I6,3X,I6,3X,I6,3X,I6,3X,I6)
  120 FORMAT((A))
  130  FORMAT(1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,
     *          I6,1X,I6)
C
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT DECIDES WHICH OF THE NEW POINTS ARE DIRICHLET OR NEUMANN
CAND GIVES THEM AN APROPRIATE ID
      SUBROUTINE GETPIDW(PID,PIDD,PIDN,NPV,NN,ND)
      INTEGER PIDD(21000,4),PIDN(21000,4),NPV,NN,ND,PID(41000)
C
      IMPLICIT NONE
      INTEGER I,ID,J,XYZ(3)
      LOGICAL WHAT1,WHAT2
C
      DO 10 I=1,NPV
        ID=PID(I)/10
        IF (ID.GT.999) THEN
          ID=ID-1000
          CALL GETINDICATOR(ID,XYZ)
          WHAT1=(XYZ(1).EQ.1).OR.(XYZ(2).EQ.1).OR.(XYZ(3).EQ.1)
          WHAT2=(XYZ(1).EQ.4).OR.(XYZ(2).EQ.4).OR.(XYZ(3).EQ.4)
          IF (WHAT1.OR.WHAT2) THEN
      ND=ND+1
      PIDD(ND,1)=I
          DO 20 J=1,3
            IF ((XYZ(J).EQ.1).OR.(XYZ(J).EQ.4)) THEN
          PIDD(ND,J+1)=1
              ELSE
                PIDD(ND,J+1)=999
            END IF
   20     CONTINUE
```

```
    END IF
   END IF
10 CONTINUE
DO 40 I=1,NPV
 ID=PID(I)/10
 IF (ID.GT.999) THEN
  ID=ID-1000
  CALL GETINDICATOR(ID,XYZ)
  WHAT1=(XYZ(1).EQ.2).OR.(XYZ(2).EQ.2).OR.(XYZ(3).EQ.2)
  WHAT2=(XYZ(1).EQ.4).OR.(XYZ(2).EQ.4).OR.(XYZ(3).EQ.4)
  IF (WHAT1.OR.WHAT2) THEN
NN=NN+1
PIDN(NN,1)=I
    DO 30 J=1,3
      IF ((XYZ(J).EQ.2).OR.(XYZ(J).EQ.4)) THEN
   PIDN(NN,J+1)=1
      ELSE
        PIDN(NN,J+1)=999
      END IF
30    CONTINUE
   END IF
  END IF
40 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT EXTRACTS THE POINT ID CODE FOR ERY COMPONENT
SUBROUTINE GETINDICATOR(ID,XYZ)
INTEGER ID,XYZ(3)
C
IMPLICIT NONE
INTEGER I,VAL
DO 10 I=1,3
  VAL=INT(ID/10)*10
  XYZ(4-I)=ID-VAL
10 ID=VAL/10
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

PREPROS.F

```
C PREPROCESSOR FOR MEASUREMENTS. PROGRAM THAT CREATES THE SECOND ORDER
C NODES ON THE MESH(USUALLY HALF WAY ON EVERY EDGE). THE FILE C
C dirichletx.dat IS READ TO SEE WHICH ARE ON DIRICHLET OR NEUMANN.
C IF THE NODE IS DIRICHLET OR NEUMANN A VALUE IS CALCULATED FOR IT
C FROM THE SOURCE EXPRESSION
C
C
IMPLICIT NONE
INTEGER NP,OFFSET,I,ND,J,D1,E(6),OFFSET1,IND,IND1,IND2
INTEGER SCP,SCF,SCE,FLGX,FLGY,FLGZ,FD,NDR
REAL P(41000,3),D(10000,3),PREC,DX,DY,DZ,X,Y,Z
REAL DUM1(3),DUM2(3),DUM3(3),FREQ
COMPLEX V(10000,3),DIRICHVALXFL,DIRICHVALYFL,DIRICHVALZFL
C
PREC=0.0001
NDR=0
OPEN(UNIT=2,FILE='unmesh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
READ(2,REC=2) NP
READ(2,REC=3) SCP
READ(2,REC=4) SCF
READ(2,REC=5) SCE
DO 10 I=1,NP
  OFFSET=5+I
10 READ(2,REC=OFFSET) P(I,1),P(I,2),P(I,3)
OFFSET=5+NP+SCE+SCF
DO 15 I=1,SCP
  OFFSET=OFFSET+1
  READ(2,REC=OFFSET) D1,D1,D1,D1,D1,E(1),E(2),E(3),E(4),E(5),E(6)
  DO 16 J=1,6
    OFFSET1=5+NP+E(J)
    READ(2,REC=OFFSET1) IND1,IND2
    CALL ONEDIM(P(IND1,1),P(IND1,2),P(IND1,3),DUM1)
    CALL ONEDIM(P(IND2,1),P(IND2,2),P(IND2,3),DUM2)
    CALL MIDPOI(DUM1,DUM2,DUM3)
    IND=E(J)+NP
    P(IND,1)=DUM3(1)
    P(IND,2)=DUM3(2)
    P(IND,3)=DUM3(3)
16  CONTINUE
15 CONTINUE
NP=NP+SCE
CLOSE(2)
C
OPEN(UNIT=2,FILE='functions.dat',STATUS='UNKNOWN',ACCESS=
     *     'SEQUENTIAL',FORM='FORMATTED')
READ(2,*) FREQ
CLOSE(2)
OPEN(UNIT=2,FILE='dirichletx.dat',STATUS='UNKNOWN',ACCESS=
     *     'SEQUENTIAL',FORM='FORMATTED')
READ(2,*) ND
DO 20 I=1,ND
  READ(2,*) D(I,1),D(I,2),D(I,3),FLGX,FLGY,FLGZ
  V(I,1)=DIRICHVALXFL(D(I,1),D(I,2),D(I,3),FLGX,FREQ)
  V(I,2)=DIRICHVALYFL(D(I,1),D(I,2),D(I,3),FLGY,FREQ)
  V(I,3)=DIRICHVALZFL(D(I,1),D(I,2),D(I,3),FLGZ,FREQ)
20 CONTINUE
CLOSE(2)
OPEN(UNIT=2,FILE='ordirichlet.dat',STATUS='UNKNOWN',ACCESS=
     *     'SEQUENTIAL',FORM='FORMATTED')
C
WRITE(2,*) ND
DO 35 I=1,ND
   X=D(I,1)
  Y=D(I,2)
  Z=D(I,3)
  FD=0
  DO 45 J=1,NP
    DX=ABS(X-P(J,1))
```

```
   DY=ABS(Y-P(J,2))
   DZ=ABS(Z-P(J,3))
   IF ( (DX.LT.PREC).AND.(DY.LT.PREC).AND.(DZ.LT.PREC) ) FD=1
   IF (FD.EQ.1) GOTO 55
45  CONTINUE
55  IF (FD.EQ.1) WRITE(2,100) J,V(I,1),V(I,2),V(I,3)
  IF (FD.NE.1) WRITE(*,*) 'ENTRY',I,'IN dirichletx.dat NOT FOUND'
35 CONTINUE
100 FORMAT(I8,3F14.7)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ONEDIM(P1,P2,P3,DUM)
REAL P1,P2,P3,DUM(3)
C
DUM(1)=P1
DUM(2)=P2
DUM(3)=P3
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MIDPOI(DUM1,DUM2,DUM3)
REAL DUM1(3),DUM2(3),DUM3(3)
C
INTEGER I
DO 10 I=1,3
10 DUM3(I)=(DUM1(I)+DUM2(I))/2
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

MAINCOMPSG.F

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C THIS PROGRAM SOLVES THE PROPAGATION OF EM WAVES USING THE FEM METHOD C
C THREE ABCs ARE IMPLEMENTED TO COMPENSATE FOR DOMAIN TRUNCATION.     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
IMPLICIT NONE
COMMON/JACOB/ ABCD,ABCDI
COMMON/PLANES/ AA,B,C,D
REAL ABCD(2000,4,4),ABCDI(2000,4,4),SOB(16,4,4),H2OS(16,4,10)
REAL AA(2000,4),B(2000,4),C(2000,4),D(2000,4),SS(16,4,4)
INTEGER SIZ,AI(1900000),AJ(20000),FIL2,NZERO,N,ITMAX,ITER,IERR,ABC
INTEGER MDIR,NZ,IWORK(30),MAXFAC,NP,NPS,CO,NODES(2000,10),IDCOD
INTEGER PYRIND(2000),NPYR,SIMP(2000),BSPYR(2000),ALP(10,5,2),NPOS
INTEGER GUESS,I,DIRN(5000),NOIN(5000),HNODE(2000,10),SPC,UPD
REAL NOI(5000),PER,NOR(2000,3),BET,H2(16,4,10),LW1,LH1,LW2,LH2
CHARACTER*24 fdate
COMPLEX DIR(5000),EPS(10)
REAL*8 A(1900000),RHS(20000),SOL(20000),TOL,ERR
REAL H(2000,16,10),DH(2000,16,10),TIMEARRAY(2),ELAPSE,ETIME
CCCCCCCCCCCCCCCCCCCCCCCCCCC
INTEGER ITOL,REPET(3)
INTEGER IUNIT,ISYM,LENW,LENIW
REAL*8 RWORK(190000)
C
WRITE(*,*) fdate()
TYPE *,'ENTER ITERATION NUMBER:'
ACCEPT *,ITMAX
TYPE *,'DO YOU HAVE AN INITIAL GUESS(1/0)?'
ACCEPT *,GUESS
TYPE *,'ENTER TOLERANCE(MIN=5.55E-14)?'
ACCEPT *,TOL
OPEN(UNIT=10,FILE='ordirichlet.dat',STATUS='UNKNOWN',
     *       ACCESS='SEQUENTIAL',FORM='FORMATTED')
FIL2=10
IDCOD=500000 !ID CODE BY WHICH PYRLAYBAS RECOGNIZES THE
!PYRAMID NEEDED FOR INTEGRATION
SIZ=0 !SIZ MUST BE < N
NZERO=0 !NZERO MUST BE < NZ
N=20000 !MAXIMUM NUMBER OF UNKNOWNS*2
NZ=1900000 !MAXIMUM NUMBER OF NONZERO ENTRIES IN A
MDIR=N*0.1 !MAXIMUM NUMBER OF DIRICHLET AND NEUMANN NODES
MAXFAC=2000 !MAXIMUM NUMBER OF FACES ON SURFACE S FOR ABC
NP=16 !MAXIMUM NUMBER OF POINTS USED/INTEGRAL ON S
NPOS=4 !ACTUAL NUMBER OF POINTS USED/INTEGRAL ON OUTER BOUND
NPS=4 !ACTUAL NUMBER OF POINTS USED ON S(NPUS<=NP)
C THE NEXT 4 VALUES HAVE TO BE CHANGED IF THE MESH SIZE IS CHANGED
LW1=0.5  !WIDTH/LENGTH OF THE FIRST BOX1 ON WICH MEASUREMENTS
!ARE MADE(FEM ONLY 1.0)
LH1=1.0  !HEIGTH OF BOX1(FEM ONLY 1.5)
LW2=0.75  !WIDTH/LENGTH OF THE SECOND BOX2 CONCENTRIC TO BOX1
!DEFINING THE VOLUME TO BE DISCRITIZED BY FEs (FEM1.5)
LH2=1.25 !HEIGTH OF BOX2(FEM ONLY 3.0)
C
CALL FANCTIONS(ALP,EPS,BET)
TYPE *,'DO YOU WANT TO READ S MATRIX FILLING?'
TYPE *,'999999 WILL READ.'
TYPE *,'ANY OTHER NUMBER WILL BE THE ASSUMED NONZERO ENTRIES/COLUMN'
TYPE *,'THIS PARAMETER IS CRUCIAL IN THE SPEED OF FORMING S'
TYPE *,'IF THIS IS NOT THE FIRST RUN USE 999999.'
ACCEPT *,SPC
TYPE *,'DO YOU WANT TO UPDATE THE FILLING (1/0)'
ACCEPT *,UPD
TYPE *,'DO YOU WANT TO USE ANY ABC(1:NEW ABC,2:BT2,3:BT1,4:SOMMER,
     * ANYTHING ELSE:NONE)?'
ACCEPT *,ABC
IF (ABC.EQ.1) THEN
WRITE(*,*) 'PREPARING IN CASE YOU USE NEW ABC, BE PATIENT!'
CALL PYRLAYRBAS(LW1,LH1,LW2,LH2,NPYR,PYRIND,NODES,MAXFAC,CO,
     * NOR,SIMP,BSPYR,IDCOD)
CALL GETSMPVALNP(SS,NP,NPS)
CALL GETSMPVALNP(SOB,NP,NPOS)
CALL GETH2(ALP,H2,SS,NP,NPS)
CALL GETH2(ALP,H2OS,SOB,NP,NPOS)
CALL GETFULLVALONS(BSPYR,SIMP,NOR,NODES,MAXFAC,CO,ALP,H2,SS,NP,
     * NPYR,PYRIND,NPS,H,DH,HNODE)
```

```
END IF
WRITE (*,*) 'OK, HERE WE GO!'
WRITE(*,*)
DO 6 I=1,3
 CALL BUILDS(A,AI,AJ,RHS,N,NZ,SIZ,NZERO,FIL2,I,DIR,NOI,
     *            DIRN,NOIN,MDIR,
     *   BSPYR,SIMP,NOR,NODES,MAXFAC,CO,ALF,H3OS,SS,SOB,NP,EPS,BET,
     *   NPYR,PYRIND,NPS,NPOS,H,DH,HNODE,ABC,SPC,UPD)
 WRITE(*,*) 'I VE JUST BUILT S AND SIZE IS:',SIZ
 PER=((NZERO*1.0)/SIZ**2)*100.0
 WRITE(*,*) 'PERCENT OCCUPANCY:',PER
 IF (GUESS.EQ.1) THEN
CALL VFILLGUESS( SIZ, SOL, N, I)
 ELSE
  CALL VFILLDIR( SIZ, SOL, RHS, N)
 END IF
 WRITE(*,*) 'SOLVING FOR THE ',I,'th COMPONENT.'
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       CALL SOLVER(SIZ, RHS, SOL, NZERO, AI, AJ, A,
C     *TOL, ITMAX, ITER, ERR, IERR,IWORK,N,NZ)
ISYM=0
ITOL=1
c IUNIT=0
cFOR MONITORING CONVERGENCE
IUNIT=6
LENW=N*9
LENIW=20
      CALL DSDCGS(SIZ, RHS, SOL, NZERO, AI, AJ, A, ISYM,
     *ITOL, TOL, ITMAX, ITER, ERR, IERR, IUNIT,RWORK,LENW,IWORK,LENIW)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
REPET(I)=ITER
      WRITE(*,*) 'I VE JUST SOLVED S FOR I EQUAL',I
WRITE(*,*) 'IT TOOK:',ITER,'ITERATIONS!'
WRITE(*,*) 'THE ERROR RESIDUE IS:',ERR
WRITE(*,*) 'STATUS IS:',IERR
WRITE(*,*) 'IT USED:',IWORK(10),' STORAGE LOCATIONS FOR REAL.'
WRITE(*,*) 'IT USED:',IWORK(9),' STORAGE LOCATIONS FOR INTEGER.'
CALL SOLWRITER(SOL,N,SIZ,I)
WRITE(*,*) 'DONE WITH ONE COMPONENT!'
6 CONTINUE
WRITE(*,*) 'I M DONE.'
CLOSE(FIL2)
DO 8 I=1,3
8 WRITE(*,*) 'COMPONENT ',I,'TOOK ',REPET(I),' ITERATIONS.'
WRITE(*,*) fdate()
ELAPSE=ETIME(TIMEARRAY)
WRITE(*,*) ELAPSE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C SUBROUTINE SOLVER(SIZ,RHS,SOL,NZERO,AI,AJ,A,TOL,ITMAX,ITER,ERR,IERR
C     *   ,IWORK,N,NZ)
C INTEGER SIZ,AI(NZ),AJ(N),NZERO,NZ
C INTEGER IWORK(20),ITER,ITMAX,IERR,N
C REAL*8 A(NZ),RHS(N),SOL(N),TOL,ERR
CC
C IMPLICIT NONE
C INTEGER ITOL
C INTEGER IUNIT,ISYM,LENW,LENIW
C REAL*8 RWORK(500000)
CC
C ISYM=0
C ITOL=1
C IUNIT=6
CcFOR MONITORING CONVERGENCE
Cc IUNIT=0
C LENW=N*10
C LENIW=20
C       CALL DSDCGN(SIZ, RHS, SOL, NZERO, AI, AJ, A, ISYM,
C     *ITOL, TOL, ITMAX, ITER, ERR, IERR, IUNIT,RWORK,LENW,IWORK,LENIW)
C END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

RESPROS.F

```
CPROGRAM THAT PROCESSES THE RESULTS IN THE FILE unsolution.dat (IT ASKS FOR
CTHE COORDINATES WHRE THE FILED IS DESIRED
C
C
IMPLICIT NONE
REAL AP(4),BP(4),CP(4),DP(4),POINT(1,3),COOR(4,3),I,J,Z,Y1,Z1
REAL SIMP(1,4),A(4),B(4),C(4),D(4),BET,IX,FX,SX,IY,IZ,FY,FZ,SY,SZ
COMPLEX EPSILON(10),EX,EY,EZ,EX0(10),EY0(10),EZ0(10),COMPUTE
INTEGER PYR,K,L,INDP(6),INDEX(4),NODE(10),NP,SCP,REK
INTEGER ALF(10,3,5),FIL,FIL2,SCE,SCF,EDGE(4,3),POINTS(4,3,2),MEDP
INTEGER MEDF(4),MEDE(4,3),SCREEN,VAR,SC
LOGICAL VAL(4),ABOVE
CHARACTER*3 AA
CHARACTER*20 FILENAME,FNAME,FNAME1
COMPLEX HR,HTET,HFI
C
C
TYPE *,'ENTER DATA FILENAME!'
ACCEPT 130,FILENAME
TYPE *,'ENTER MESH FILENAME!'
ACCEPT 130,FNAME
TYPE *,'WAIT A SEC!'
OPEN(UNIT=4,FILE=FILENAME,STATUS='UNKNOWN',ACCESS='DIRECT'
     *       ,FORM='UNFORMATTED',RECL=2,RECORDTYPE='FIXED')
CALL FANCTIONS(ALF,EPSILON,BET)
CALL DEFINEPYR(FIL,SCP,NP,FIL2,SCE,SCF,FNAME)
5 TYPE *,'ENTER OUTPUT VARIABLE(1 FOR SCREEN/2 FOR FILES) :'
ACCEPT *,SCREEN
IF (SCREEN.EQ.2) THEN
TYPE *,'ENTER OUTPUT FILENAME!'
ACCEPT 130,FNAME1
  OPEN(UNIT=20,FILE=FNAME1,STATUS='UNKNOWN',ACCESS=
     *    'SEQUENTIAL',FORM='FORMATTED')
  TYPE *,'ENTER COORDINATE THAT VARIES(R/1,THETA/2,PHI/3) :'
  ACCEPT *,VAR
```

```
END IF
TYPE *,'ENTER 99 FOR SPHERICAL/ANYTHING ELSE FOR CARTESIAN'
ACCEPT *,SC
TYPE *,'ENTER RADIUS:'
ACCEPT *,IX,FX,SX
TYPE *,'ENTER INITIAL THETA, FINAL THETA AND THE STEP :'
ACCEPT *,IY,FY,SY
TYPE *,'ENTER INITIAL PHI, FINAL PHI AND THE STEP :'
ACCEPT *,IZ,FZ,SZ
DO 10 I=IX,FX,SX
 DO 20 J=IY,FY,SY
  DO 25 Z=IZ,FZ,SZ
   Y1=J*3.14159/180.0
   Z1=Z*3.14159/180.0
   POINT(1,1)=I*SIN(Y1)*COS(Z1)
   POINT(1,2)=I*SIN(Y1)*SIN(Z1)
   POINT(1,3)=I*COS(Y1)
   PYR=0
   DO 30 K=1,SCP
 READ(FIL,REC=K) AP(1),BP(1),CP(1),DP(1),AP(2),BP(2),CP(2),DP(2),
    *    AP(3),BP(3),CP(3),DP(3),AP(4),BP(4),CP(4),DP(4)
DO 40 L=1,4
40  VAL(L)=ABOVE(POINT,AP(L),BP(L),CP(L),DP(L))
IF (VAL(1).AND.VAL(2).AND.VAL(3).AND.VAL(4)) PYR=K
IF (PYR.NE.0) GOTO 35
30 CONTINUE
35 IF (PYR.EQ.0) WRITE(*,*) 'PROBLEM I could not find a pyramid
    *enclosing these coordinates.This point is outside the mesh:'
 IF (PYR.EQ.0) WRITE(*,*)  'POINT:',POINT
IF (PYR.EQ.0) GOTO 70
CALL READINFO(PYR,NP,SCE,SCF,EDGE,POINTS,MEDP,
    *                  MEDF,MEDE,INDP,COOR,INDEX,FIL2)
CALL ALLCOEFFICIENT(COOR,A,B,C,D)
CALL GETSIMPLEX(POINT,1,3,SIMP,1,4,A,B,C,D)
CALL GETNODES(NODE,NP,INDEX,INDP)
DO 50 K=1,10
 REK=NODE(K)
 READ(4,REC=REK) EX0(K)
 REK=NODE(K)+NP+SCE
 READ(4,REC=REK) EY0(K)
 REK=NODE(K)+2*(NP+SCE)
50  READ(4,REC=REK) EZ0(K)
EX=COMPUTE(EX0,ALF,SIMP)
EY=COMPUTE(EY0,ALF,SIMP)
EZ=COMPUTE(EZ0,ALF,SIMP)
IF (SC.EQ.99) THEN
HR=EX*SIN(Y1)*COS(Z1)+EY*SIN(Y1)*SIN(Z1)+EZ*COS(Y1)
HTET=EX*COS(Y1)*COS(Z1)+EY*COS(Y1)*SIN(Z1)-EZ*SIN(Y1)
HFI=-EX*SIN(Z1)+EY*COS(Z1)+EZ*0.0
ELSE
HR=EX
HTET=EY
HFI=EZ
END IF
CALL CONVTOMAGFAZ(HR)
CALL CONVTOMAGFAZ(HTET)
CALL CONVTOMAGFAZ(HFI)
IF (SCREEN.EQ.2) THEN
 IF (VAR.EQ.1) WRITE(20,110) I,HR,HTET,HFI
 IF (VAR.EQ.2) WRITE(20,110) J,HR,HTET,HFI
 IF (VAR.EQ.3) WRITE(20,110) Z,HR,HTET,HFI
END IF
IF (SCREEN.EQ.1) WRITE(*,100)
    * POINT(1,1),POINT(1,2),POINT(1,3),HR,HTET,HFI
25 CONTINUE
20 CONTINUE
10 CONTINUE
IF (SCREEN.EQ.2) CLOSE(20)
70 TYPE *,'OTHER VALUES (y/n)?'
ACCEPT 120,AA
IF (AA.EQ.'y') GOTO 8
100 FORMAT('FIELD AT: (',F7.3,',',F7.3,',',F7.3,') IS Hr=(',E10.3,
    *  ',',F7.2,');Htet=(',E10.3,',',F7.2,');Hphi=(',E10.3,',',
    *   F7.2,').')
110  FORMAT(1X,F7.3,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2)
120 FORMAT(A2)
130 FORMAT(A20)
CLOSE(4)
CLOSE(FIL)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES FE INFORMATION AND STORES IT IN pyramid.dat FOR
CFASTER EXECUTION
SUBROUTINE DEFINEPYR(FIL,SCP,NP,FIL1,SCE,SCF,FNAME)
INTEGER FIL,SCP,NP,FIL1
CHARACTER*20 FNAME
C
IMPLICIT NONE
INTEGER I,INDP(6),INDEX(4),SCF,EDGE(4,3),POINTS(4,3,2)
INTEGER MEDP,MEDF(4),MEDE(4,3),SCE,J
REAL COOR(4,3),AP(4),BP(4),CP(4),DP(4),DUM1(3),DUM2(3),DUM3(3)
REAL DUM4(3),DUM5(3),DUM6(3),DUM7(3),DUM8(3),ANG(4),COSINUS
C
OPEN(UNIT=2,FILE=FNAME,STATUS='UNKNOWN',ACCESS='DIRECT',
    *     FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
OPEN(UNIT=3,FILE='pyramid.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
    *     FORM='UNFORMATTED',RECL=16,RECORDTYPE='FIXED')
C
READ(2,REC=2) NP
READ(2,REC=3) SCP
READ(2,REC=4) SCF
READ(2,REC=5) SCE
FIL1=2
FIL=3
DO 10 I=1,SCP
 CALL READINFO(I,NP,SCE,SCF,EDGE,POINTS,MEDP,
    *                  MEDF,MEDE,INDP,COOR,INDEX,FIL1)
 CALL ONEDIM(COOR(1,1),COOR(1,2),COOR(1,3),DUM1)
 CALL ONEDIM(COOR(2,1),COOR(2,2),COOR(2,3),DUM2)
 CALL ONEDIM(COOR(3,1),COOR(3,2),COOR(3,3),DUM3)
 CALL ONEDIM(COOR(4,1),COOR(4,2),COOR(4,3),DUM4)
 CALL PLANEPAR3(DUM1,DUM2,DUM3,AP(1),BP(1),CP(1),DP(1))
```

```
CALL PLANEPAR3(DUM1,DUM2,DUM4,AP(2),BP(2),CP(2),DP(2))
CALL PLANEPAR3(DUM1,DUM3,DUM4,AP(3),BP(3),CP(3),DP(3))
CALL PLANEPAR3(DUM2,DUM3,DUM4,AP(4),BP(4),CP(4),DP(4))
CALL DUMP4PROJ(DUM4,AP(1),BP(1),CP(1),DP(1),DUM5)
CALL DUMP4PROJ(DUM3,AP(2),BP(2),CP(2),DP(2),DUM6)
CALL DUMP4PROJ(DUM2,AP(3),BP(3),CP(3),DP(3),DUM7)
CALL DUMP4PROJ(DUM1,AP(4),BP(4),CP(4),DP(4),DUM8)
CALL DIRECTION(DUM4,DUM6,DUM5)
CALL DIRECTION(DUM3,DUM6,DUM6)
CALL DIRECTION(DUM2,DUM7,DUM7)
CALL DIRECTION(DUM1,DUM8,DUM8)
ANG(1)=COSINUS(DUM5,AP(1),BP(1),CP(1))
ANG(2)=COSINUS(DUM6,AP(2),BP(2),CP(2))
ANG(3)=COSINUS(DUM7,AP(3),BP(3),CP(3))
ANG(4)=COSINUS(DUM8,AP(4),BP(4),CP(4))
DO  20 J=1,4
  IF (ANG(J).LT.0.0) THEN
     AP(J)=-AP(J)
BP(J)=-BP(J)
CP(J)=-CP(J)
DP(J)=-DP(J)
    END IF
  IF (ABS(AP(J)).LT.1.0E-08)  AP(J)=0.0
  IF (ABS(BP(J)).LT.1.0E-08)  BP(J)=0.0
  IF (ABS(CP(J)).LT.1.0E-08)  CP(J)=0.0
  IF (ABS(DP(J)).LT.1.0E-08)  DP(J)=0.0
20 CONTINUE
WRITE(3,REC=I) AP(1),BP(1),CP(1),DP(1),AP(2),BP(2),CP(2),
     *     DP(2),AP(3),BP(3),CP(3),DP(3),AP(4),BP(4),CP(4),DP(4)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
COMPLEX FUNCTION COMPUTE(VAL,ALF,SIMP)
COMPLEX VAL(10)
INTEGER ALF(10,2,5)
REAL SIMP(1,4)
C
IMPLICIT NONE
COMPLEX SUM
INTEGER I,J,POW1,POW2,POW3,POW4,COEF
SUM=CMPLX(0.0,0.0)
DO 10 I=1,10
  DO 20 J=1,2
    POW1=ALF(I,J,1)
    POW2=ALF(I,J,2)
    POW3=ALF(I,J,3)
    POW4=ALF(I,J,4)
    COEF=ALF(I,J,5)
   SUM=( SIMP(1,1)**POW1*SIMP(1,2)**POW2*SIMP(1,3)**POW3*SIMP(1,4)
     *          **POW4)*COEF*VAL(I)+SUM
20 CONTINUE
10 CONTINUE
COMPUTE=SUM
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CONVERTS AN IMAGINARY FIELD TO POLAR FORM
SUBROUTINE CONVTOMAGFAZ(EX)
COMPLEX EX
C
IMPLICIT NONE
REAL RVAL,IVAL,PHASE
C
RVAL=CABS(EX)
IVAL=PHASE(EX)
EX=CMPLX(RVAL,IVAL)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION PHASE(VAL)
COMPLEX VAL
C
IMPLICIT NONE
REAL DX,DY,FI,PI,ANG
C
PI=ACOS(-1.0)
DX=REAL(VAL)
DY=AIMAG(VAL)
FI=ATAN(DY/DX)
IF (DX.EQ.0) FI=0.0
IF ((DX.EQ.0).AND.(DY.GT.0)) FI=PI/2
IF ((DX.EQ.0).AND.(DY.LT.0)) FI=-PI/2
IF ((DX.GT.0).AND.(DY.EQ.0)) FI=0.0
IF ((DX.LT.0).AND.(DY.EQ.0)) FI=PI
IF ((DX.LT.0).AND.(DY.LT.0)) FI=FI+PI
IF ((DX.LT.0).AND.(DY.GT.0)) FI=FI+PI
ANG=FI*180.0/PI
IF (ANG.GT.180.0) ANG=ANG-360.0
IF (ANG.LT.-180.0) ANG=ANG+360.0
PHASE=ANG
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

BEMRAD.F

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  THIS PROGRAM COMPUTES THE FIELD VALUES AT ANY RANGE FROM A THE INTEGRAL  C
C  OF NEAR-FIELD VALUES ON A CLOSED SURFACE. THE ALGORITHM IS BASED ON GREEN C
C  FIRST THEOREM APPLIED TO RADIATION FIELDS                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
IMPLICIT NONE
COMMON/DIR/ ALLDIR,ALLDIRN,MAXDIR
COMMON/JACOB/ ABCD,ABCDI
COMMON/PLANES/ A,B,C,D
INTEGER NODES(2000,10),I,NOD1(10),ALLDIRN(6000),NPUS,IDCOD
INTEGER SIMP(3000),CI,SIZE,ALF(10,2,5),SMP1,MAXN,VAR,PASS
INTEGER MAXDIR,NPI,PYRIND(2000),NPYR,BSPYR(2000),SCREEN,SC
```

```
      REAL ABCD(2000,4,4),ABCDI(2000,4,4),NOR(2000,3),NVAL
      REAL ABCD1(4,4),ABCDI1(4,4),NOR1(3),H2(30,4,10)
      REAL BET,X,Y,Z,IX,IY,FY,SY,IZ,FZ,SZ,S(30,4,4),LW1,LH1,LW2,LH2
      REAL A(2000,4),B(2000,4),C(2000,4),D(2000,4),Y1,Z1,Y2,Z2,X1
      COMPLEX EPS(10),VAL(3),HF(3),ALLDIR(6000,3),DIR(10,3)
      COMPLEX H(2000,30,3),DH(2000,30,3),HI(2000,30,3),DHI(2000,30,3)
      COMPLEX HR,HTET,HFI
      CHARACTER*2 AA
      CHARACTER*20 FNAME
C
C
      IDCOD=811111 !ID CODE BY WHICH PYRLAYBAS RECOGNIZES THE PYRAMIDS
      !NEEDED FOR INTEGRATION
      NPUS=4       !NUMBER OF COLLOCATION POINT FOR THE RIEMMAN
      !SUM(INTEGRAL)
      SIZE=2000 !MAXIMUM NUMBER OF PYRAMIDS IN THE LAYER OF FE
      NPI=30 !MAXIMUM NUMBER OF NPUS
      MAXN=6000 !MAXIMUM NUMBER OF NODES IN THE FIRST LAYER OF FE
      SCREEN=1 !PARAMETER FOR SCREEN/FILE OUTPUT
      NVAL=0 !NUMBER OF POINTS COMPUTED IN THE FAR-FIELD
      PASS=0 !PARAMETER TO BYPASS ROUTINES NECESSARY FOR FIRST RUN
      !ONLY
C THE NEXT 4 VALUES HAVE TO BE CHANGED IF THE MESH SIZE IS CHANGED
      LW1=0.5 !WIDTH/LENGTH OF THE FIRST BOX1 ON WICH MEASUREMENTS
      !ARE MADE
      LH1=1.0 !HEIGTH OF BOX1
      LW2=0.75 !WIDTH/LENGTH OF THE SECOND BOX2 CONCENTRIC TO BOX1
      !DEFINING THE VOLUME TO BE DISCRITIZED BY FE
      LH2=1.25 !HEIGTH OF BOX2
      CALL PYRLAYRBAS(LW1,LH1,LW2,LH2,NPYR,PYRIND,NODES,SIZE,CI,
     *  NOR,SIMP,BSPYR,IDCOD)
      WRITE(*,*) '# OF INNER FACES=',CI
      WRITE(*,*) '# OF BOUNDARY PYRAMIDS=',NPYR
      CALL FANCTIONS(ALF,EPS,BET)
      CALL READDIRVB(ALLDIR,ALLDIRN,MAXN,MAXDIR)
      CALL GETSMPVALNP(S,NPI,NPUS)
      CALL GETH2(ALF,H2,S,NPI,NPUS)
5     TYPE *,'ENTER OUTPUT VARIABLE(1 FOR SCREEN/2 FOR FILES) :'
      ACCEPT *,SCREEN
      IF (SCREEN.EQ.2) THEN
      TYPE *,'ENTER OUTPUT FILENAME?'
      ACCEPT 140,FNAME
       OPEN(UNIT=20,FILE=FNAME,STATUS='UNKNOWN',ACCESS=
     *      'SEQUENTIAL',FORM='FORMATTED')
       TYPE *,'ENTER COORDINATE THAT VARIES(X/1,Y/2,Z/3) :'
       ACCEPT *,VAR
      END IF
      TYPE *,'ENTER RADIUS :'
      ACCEPT *,IX
      TYPE *,'ENTER INITIAL THETA, FINAL THETA AND THE STEP :'
      ACCEPT *,IY,FY,SY
      TYPE *,'ENTER INITIAL PHI, FINAL PHI AND THE STEP :'
      ACCEPT *,IZ,FZ,SZ
      TYPE *,'ENTER 99 FOR SPHERICAL/ANYTHING ELSE FOR CARTESIAN'
      ACCEPT *, SC
      DO 10 X1=IX,IX,IX
       DO 20 Y2=IY,FY,SY
        DO 30 Z2=IZ,FZ,SZ
         HF(1)=CMPLX(0.0,0.0)
         HF(2)=CMPLX(0.0,0.0)
         HF(3)=CMPLX(0.0,0.0)
         NVAL=NVAL+1.0
         Y1=Y2*3.14159/180.0
         Z1=Z2*3.14159/180.0
         X=X1*SIN(Y1)*COS(Z1)
         Y=X1*SIN(Y1)*SIN(Z1)
         Z=X1*COS(Y1)
         DO 40 I=1,CI
           CALL SELECTDATA(BSPYR(I),I,SIMP,NOR,NODES,SIZE,NOD1,SMP1,
     *          ABCD1,ABCDI1,NOR1)
      IF (PASS.EQ.0) THEN
           CALL GETDIR10(DIR,NOD1,PYRIND(BSPYR(I)))
                CALL INTEGRAL(X,Y,Z,BET,VAL,SMP1,ABCD1,ABCDI1,NOR1,
     *          ALF,H2,DIR,S,NPI,NPYR,PYRIND,NODES,SIZE,NPUS,BSPYR(I),
     *          I,H,DH,HI,DHI)
      ELSE
                CALL INTEGRAL1(X,Y,Z,BET,VAL,SMP1,ABCD1,ABCDI1,NOR1,
     *          S,NPI,SIZE,NPUS,I,H,DH,HI,DHI)
       END IF
           HF(1)=HF(1)+VAL(1)
           HF(2)=HF(2)+VAL(2)
40         HF(3)=HF(3)+VAL(3)
         PASS=1
         IF (SC.EQ.99) THEN
         HR=HF(1)*SIN(Y1)*COS(Z1)+HF(2)*SIN(Y1)*SIN(Z1)+HF(3)*COS(Y1)
         HTET=HF(1)*COS(Y1)*COS(Z1)+HF(2)*COS(Y1)*SIN(Z1)-HF(3)*SIN(Y1)
         HFI=-HF(1)*SIN(Z1)+HF(2)*COS(Z1)+HF(3)*0.0
         ELSE
         HR=HF(1)
         HTET=HF(2)
         HFI=HF(3)
           END IF
         CALL CONVTOMAGFAZ(HR)
         CALL CONVTOMAGFAZ(HTET)
         CALL CONVTOMAGFAZ(HFI)
         IF (SCREEN.EQ.2) THEN
      IF (VAR.EQ.1) WRITE(20,110) X1,HR,HTET,HFI
      IF (VAR.EQ.2) WRITE(20,110) Y2,HR,HTET,HFI
      IF (VAR.EQ.3) WRITE(20,110) Z2,HR,HTET,HFI
         END IF
         IF (SCREEN.EQ.1) WRITE(*,100) X,Y,Z,HR,HTET,HFI
50      CONTINUE
20     CONTINUE
10    CONTINUE
      IF (SCREEN.EQ.2) CLOSE(20)
      TYPE *,'OTHER VALUES (y/n)?'
      ACCEPT 120,AA
      IF (AA.EQ.'y') GOTO 5
100   FORMAT('AT:(',F8.5,',',F8.5,',',F8.5,');Hr=(',E12.5,
     *  ',',F10.5,');Htet=(',E12.5,',',F10.5,');Hphi=(',E12.5,',',
     *  F10.5,').')
110    FORMAT(1X,F7.3,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2)
120   FORMAT(A2)
```

```
140 FORMAT(A20)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTE THE INTEGRAL JUST LIKE THE SUBROUTINE 'INTEGRAL'
CHOWEVER ONLY THE GREEN'S FUNCTIONS ARE RECOMPUTED. AFTER THE FIRST PASS
CTHE H,DH/DN AND THEIR IMAGES ARE STORED FROM THE SUBROUTINE 'INTEGRAL'
CTHIS SPEEDS UP THING TREMENDESLY
SUBROUTINE INTEGRAL1(X,Y,Z,K,VAL,SIMP,ABCD1,ABCDI1,NOR,S,
     * NP,SIZE,NPUS,BS,H,DH,HI,DHI)
COMPLEX VAL(3),H(SIZE,NP,3),DH(SIZE,NP,3),DHI(SIZE,NP,3)
COMPLEX HI(SIZE,NP,3)
INTEGER SIMP,NP,SIZE
INTEGER NPUS,BS
REAL ABCD1(4,4),ABCDI1(4,4)
REAL NOR(3),S(NP,4,4),X,Y,Z,K
C
IMPLICIT NONE
COMPLEX GG(30,3),G(30),GGI(30,3),GI(30)
COMPLEX GN(30),GNI(30),DUM
REAL P(30,3),PI(30,3),NORI(3)
INTEGER AX,I
C
NORI(1)=NOR(1)
NORI(2)=NOR(2)
NORI(3)=-NOR(3)
CALL GET3PGLOB(ABCDI1,S,P,PI,SIMP,NP,NPUS)
CALL GETGRADG(PI,GGI,K,X,Y,Z,NP,NPUS)
CALL GETG(PI,GI,K,X,Y,Z,NP,NPUS)
CALL GETGRADG(P,GG,K,X,Y,Z,NP,NPUS)
CALL GETG(P,G,K,X,Y,Z,NP,NPUS)
CALL DOTGN(GG,NOR,GN,NP,NPUS)
CALL DOTGN(GGI,NORI,GNI,NP,NPUS)
DO 10 AX=1,3
  DUM=CMPLX(0.0,0.0)
  DO 20 I=1,NPUS
20   DUM=DUM+(H(BS,I,AX)*GN(I)-G(I)*DH(BS,I,AX))
  DO 30 I=1,NPUS
30   DUM=DUM+(HI(BS,I,AX)*GNI(I)-GI(I)*DHI(BS,I,AX))
c minus sign because Nor was directed inward instead of outward
  VAL(AX)=-(DUM/12.56637)*0.0078125
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT DOES THE INTEGRAL OVER A SURFACE
SUBROUTINE INTEGRAL(X,Y,Z,K,VAL,SIMP,ABCD1,ABCDI1,NOR,ALF,HS,DIR,S,
     * NP,NPYR,PYRIND,NODES,SIZE,NPUS,PYR,P'S,H,DH,HI,DHI)
COMPLEX VAL(3),DIR(10,3),H(SIZE,NP,3),DH(SIZE,NP,3),DHI(SIZE,NP,3)
COMPLEX HI(SIZE,NP,3)
INTEGER SIMP,NP,NPYR,PYRIND(SIZE),SIZE,ALF(10,3,8),NODES(SIZE,10)
INTEGER NPUS,PYR,BS
REAL ABCD1(4,4),ABCDI1(4,4),HS(NP,4,10)
REAL NOR(3),S(NP,4,4),X,Y,Z,K
C
IMPLICIT NONE
COMPLEX GG(30,3),G(30),GGI(30,3),GI(30),GFN(30,3),GFNI(30,3)
COMPLEX GN(30),GNI(30),DUM,F(30,3),FI(30,3)
REAL P(30,3),PI(30,3),NORI(3)
INTEGER AX,I
C
NORI(1)=NOR(1)
NORI(2)=NOR(2)
NORI(3)=-NOR(3)
CALL GET3PGLOB(ABCDI1,S,P,PI,SIMP,NP,NPUS)
CALL GETGRADG(PI,GGI,K,X,Y,Z,NP,NPUS)
CALL GETG(PI,GI,K,X,Y,Z,NP,NPUS)
CALL GETGRADG(P,GG,K,X,Y,Z,NP,NPUS)
CALL GETG(P,G,K,X,Y,Z,NP,NPUS)
CALL DOTGN(GG,NOR,GN,NP,NPUS)
CALL DOTGN(GGI,NORI,GNI,NP,NPUS)
CALL GETGRADH(P,NP,NPUS,GFN,NOR,NPYR,PYRIND,NODES,SIZE,ALF,
     * PYR,DIR,ABCD1)
CALL GETGRADHI(GFN,GFNI,NOR,NORI,NP,NPUS)
CALL GETH(HS,DIR,SIMP,F,NP,NPUS)
CALL GETHI(F,FI,NP,NPUS)
DO 10 AX=1,3
  DUM=CMPLX(0.0,0.0)
  DO 20 I=1,NPUS
    H(BS,I,AX)=F(I,AX)
    DH(BS,I,AX)=GFN(I,AX)
20   DUM=DUM+(F(I,AX)*GN(I)-G(I)*GFN(I,AX))
  DO 30 I=1,NPUS
    HI(BS,I,AX)=FI(I,AX)
    DHI(BS,I,AX)=GFNI(I,AX)
30   DUM=DUM+(FI(I,AX)*GNI(I)-GI(I)*GFNI(I,AX))
c minus sign because Nor was directed inward instead of outward
CO.0078125 DEPENS ON THE SIZE OF THE FACES AND HAS TO BE CHANGED EVERY TIME
CTHE MESH SIZE IS MODIFIED(GIMME A BREAK)
  VAL(AX)=-(DUM/12.56637)*0.0078125
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

CSUBROUTINE THAT RETURNS THE IMAGE OF DH/DN ABOUT THE XY PLANE FOR ALL 3
CCARTESIANCOMPONENETS.
SUBROUTINE GETGRADHI(F,FI,NOR,NORI,N,NS)
COMPLEX F(N,3),FI(N,3)
INTEGER AX,N,NS
REAL NOR(3),NORI(3)
C
IMPLICIT NONE
INTEGER I,DIR
C
DIR=NOR(1)*NORI(1)+NOR(2)*NORI(2)+NOR(3)*NORI(3)
DO 10 AX=1,3
  DO 8 I=1,NS
        FI(I,AX)=F(I,AX)
    IF (AX.EQ.3) FI(I,AX)=-FI(I,AX)
8   CONTINUE
```

```
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CONTROLS THE COMPUTATION OF THE GRADIENT OF H AT ALL CCOLLOCATION POINTS AND CORECTES FOR THE NOR
MAL DIRECTION
SUBROUTINE GETGRADH(P,N,NS,FN,NOR,NPYR,PYRIND,NODES,SIZ,ALF,
   * PYR,DIR,ABCD)
INTEGER NPYR,SIZ,PYRIND(SIZ),N,NODES(SIZ,10),ALF(10,2,5),NS,PYR
COMPLEX FN(N,3),DIR(10,3)
REAL P(N,3),NOR(3),ABCD(4,4)
C
IMPLICIT NONE
COMPLEX DHN(3)
INTEGER I,J,NODE(10),AX
REAL PD(3),S(4)
C
DO 10 I=1,NS
 DO 20 J=1,3
20   PD(J)=P(I,J)
 CALL EVALUATEGH(PD,NOR,NPYR,PYRIND,NODES,DHN,SIZ,ALF,PYR,
   *   ABCD,S,NODE,DIR)
C MINUS SIGN DEPENDING ON DEFINITION OF NORMAL
 DO 30 AX=1,3
30   FN(I,AX)=-DHN(AX)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT EVALUATES DH/DN USING 3 POINT FORWARD DERIVATIVE FROM THE
CPOINT PD
SUBROUTINE EVALUATEGH(PD,NOR,NPYR,PYRIND,NODES,DHN,SIZE,ALF,PYR,
   * ABCD,S,NODE,DIR)
INTEGER NPYR,SIZE,PYRIND(SIZE),NODES(SIZE,10),ALF(10,2,5),PYR
INTEGER NODE(10)
REAL PD(3),NOR(3),ABCD(4,4),S(4)
COMPLEX DHN(3),DIR(10,3)
C
IMPLICIT NONE
COMMON/DIR/ ALLDIR,ALLDIRN,MAXDIR
INTEGER I,J,ALLDIRN(6000),MAXDIR,OLPYR,AX
REAL P(3),P1(3),P2(3),MN
COMPLEX ALLDIR(6000,3),VAL(3)
C
DO 5 AX=1,3
5 DHN(AX)=CMPLX(0.0,0.0)
DO 10 J=0,2
 DO 20 I=1,3
  P(I)=PD(I)-0.001*J*NOR(I)
  IF (J.EQ.0) P1(I)=P(I)
20  IF (J.EQ.1) P2(I)=P(I)
 OLPYR=PYR
 CALL GETPYR(P,NPYR,PYRIND,SIZE,PYR)
 IF (OLPYR.NE.PYR) THEN
  CALL GETNODEJACOB(ABCD,NODE,PYR,NODES,SIZE)
  CALL GETDIR10(DIR,NODE,PYRIND(PYR))
 END IF
 CALL GETSPLOC(ABCD,P,S)
 CALL GETVALUE(ALF,S,DIR,J,VAL)
 DO 30 AX=1,3
30   DHN(AX)=DHN(AX)+VAL(AX)
10 CONTINUE
MN=SQRT( (P2(1)-P1(1))**2+(P2(2)-P1(2))**2+(P2(3)-P1(3))**2)
DO 40 AX=1,3
40 DHN(AX)=DHN(AX)/MN/2.0
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE FIELD AT S, USINGTHE DIRICHLET VALUES AND
CADAPTED TO THE NUMERICAL DERIVATIVE. SIMILAR TO GETH2 AND GETH COMBINED
SUBROUTINE GETVALUE(ALF,S,DIR,OP,VAL2)
COMPLEX VAL2(3),DIR(10,3)
INTEGER ALF(10,2,5),OP
REAL S(4)
C
IMPLICIT NONE
INTEGER I,SP,P1,P2,P3,P4,AX
COMPLEX VAL1,VAL
C
DO 5 AX=1,3
 VAL2(AX)=CMPLX(0.0,0.0)
 DO 10 I=1,10
  VAL=CMPLX(0.0,0.0)
  DO 20 SP=1,2
   P1=ALF(I,SP,1)
   P2=ALF(I,SP,2)
   P3=ALF(I,SP,3)
   P4=ALF(I,SP,4)
   VAL1=( (S(1)**P1)*(S(2)**P2)*(S(3)**P3)*(S(4)**P4) )*ALF(I,SP,5)
20     VAL=VAL+VAL1
10  VAL2(AX)=VAL2(AX)+VAL*DIR(I,AX)
 IF (OP.EQ.0) VAL2(AX)=-3.0*VAL2(AX)
 IF (OP.EQ.1) VAL2(AX)=4.0*VAL2(AX)
 IF (OP.EQ.2) VAL2(AX)=-VAL2(AX)
c  IF (OP.EQ.0) VAL2(AX)=-VAL2(AX)
c  IF (OP.EQ.1) VAL2(AX)=VAL2(AX)
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMUTE STHE IMAGE OF THE H FIELD ABOUT THE XY PLANE FOR
CTHE 3 CARTESIAN COMPONENETS
SUBROUTINE GETHI(F,FI,N,NS)
COMPLEX F(N,3),FI(N,3)
INTEGER N,NS
C
IMPLICIT NONE
INTEGER I,FAC,AX
C
DO 5 AX=1,3
 FAC=1
 IF (AX.EQ.3) FAC=-1
 DO 10 I=1,NS
```

```
10   FI(I,AX)=F(I,AX)*FAC
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE GREEN'S FUNCTION FOR ALL COLLOCATION POINTS
SUBROUTINE GETG(P,G,K,X,Y,Z,N,NS)
REAL X,Y,Z,K,P(N,3)
COMPLEX G(N)
INTEGER N,NS
C
IMPLICIT NONE
C
REAL R
INTEGER I
COMPLEX EX
C
DO 10 I=1,NS
  R=SQRT( (X-P(I,1))**2+(Y-P(I,2))**2+(Z-P(I,3))**2 )
  EX=CMPLX(0.0,-1.0)*K*R
10   G(I)=CEXP(EX)/R
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE GRADIENT OF THE GREEN'S FUNCTION FOR ALL
CCOLLOCATION POINTS
SUBROUTINE GETGRADG(P,GG,K,X,Y,Z,N,NS)
REAL X,Y,Z,K,P(N,3)
CCMPLEX GG(N,3)
INTEGER N,NS
C
IMPLICIT NONE
C
REAL R
INTEGER I
COMPLEX FI,EX,FIP
C
DO 10 I=1,NS
  R=SQRT( (X-P(I,1))**2+(Y-P(I,2))**2+(Z-P(I,3))**2 )
  EX=CMPLX(0.0,-1.0)*K*R
  FI=CEXP(EX)/R
  FIP=(FI/(R**2)+CMPLX(0.0,1.0)*K*FI/R)
  GG(I,1)=(X-P(I,1))*FIP
  GG(I,2)=(Y-P(I,2))*FIP
10   GG(I,3)=(Z-P(I,3))*FIP
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE FIELD ON THE SURFACE PATCH(ONE FACE OF THE
CPYRAMID FOR ONE PARTICULAR NORMAL SP AND USING THE FIELD NODE VALUES
CTHE OUTPUT IS THE FIELD FOR ALL COLLOCATION POINT AND IN THE 3 CARTESIAN
CDIRECTION OF THE FIELD.
SUBROUTINE GETH(H,DIR,SP,F,N,NS)
REAL H(N,4,10)
COMPLEX F(N,3),DIR(10,3)
INTEGER SP,N,NS
C
IMPLICIT NONE
C
INTEGER I,P,AX
COMPLEX VAL(3)
C
DO 5 AX=1,3
  DO 10 P=1,NS
    VAL(AX)=CMPLX(0.0,0.0)
    DO 20 I=1,10
20       VAL(AX)=VAL(AX)+H(P,SP,I)*DIR(I,AX)
    F(P,AX)=VAL(AX)
10     CONTINUE
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT READS ALL THE DIRICHLET VALUES AVAILABLE AND STORES THEM IN
CVECTORS. THIS IS FASTER THAT ACCESSING THE FILE EVERY TIME.
CALLDIR IS THE VALUE OF THE NODE X,Y,Z COMPONENT
CALLDIRN IS THE INDEX OF THE NODE
CMAXDIR IS THE NUMBER OF DIRICHLET NODES
C
SUBROUTINE READDIRVS(ALLDIR,ALLDIRN,N,MAXDIR)
INTEGER ALLDIRN(N),MAXDIR,N
COMPLEX ALLDIR(N,3)
C
IMPLICIT NONE
INTEGER J,IND
REAL X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
C
OPEN(UNIT=11,FILE='ordirichlet.bem',STATUS='UNKNOWN',ACCESS=
     * 'SEQUENTIAL',FORM='FORMATTED')
READ(11,*) MAXDIR
DO 10 J=1,MAXDIR
  READ(11,*) IND,X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
       ALLDIRN(IND)=J
ALLDIR(J,1)=CMPLX(X1X,X1Y)
ALLDIR(J,2)=CMPLX(Y1X,Y1Y)
  ALLDIR(J,3)=CMPLX(Z1X,Z1Y)
10 CONTINUE
CLOSE(11)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

ANALYTICRAD.F

```
C PROGRAM THAT GIVES THE ANALYTICAL SOLUTION FOR THE SOURCES USED IN
C AND IN MAINCOMP(SOURCES ARE DEFINED IN 'SOURCESUBS.F'
C
C
IMPLICIT NONE
REAL I,J,K,IX,FX,SX,IY,FY,SY,IZ,FZ,SZ,Y1,Z1
REAL X,Y,Z,FREQ,NVAL
```

```
COMPLEX EX,EY,EZ,DIRICHLETVALX,DIRICHLETVALY,DIRICHLETVALZ
COMPLEX HR,HTET,HFI
INTEGER SCREEN,VAR,SC
CHARACTER*2 AA
CHARACTER*20 FNAME
C
C
OPEN(UNIT=3,FILE='functions.dat',STATUS='UNKNOWN',ACCESS=
     *       'SEQUENTIAL',FORM='FORMATTED')
READ(3,*) FREQ
CLOSE(3)
TYPE *,'ENTER OUTPUT VARIABLE(1 FOR SCREEN/2 FOR FILES) :'
ACCEPT *,SCREEN
IF (SCREEN.EQ.2) THEN
TYPE *,'ENTER OUTPUT FILENAME?'
ACCEPT 140,FNAME
 OPEN(UNIT=20,FILE=FNAME,STATUS='UNKNOWN',ACCESS=
     *       'SEQUENTIAL',FORM='FORMATTED')
 TYPE *,'ENTER COORDINATE THAT VARIES(X/1,Y/2,Z/3,TWO/4) :'
 ACCEPT *,VAR
END IF
5 TYPE *,'ENTER RADIUS :'
ACCEPT *,IX,FX,SX
TYPE *,'ENTER INITIAL THETA, FINAL THETA AND THE STEP :'
ACCEPT *,IY,FY,SY
TYPE *,'ENTER INITIAL PHI, FINAL PHI AND THE STEP :'
ACCEPT *,IZ,FZ,SZ
TYPE *,'ENTER 99 FOR SPHERICAL/ANYTHING ELSE FOR CARTESIAN'
ACCEPT *, SC
DO 10 I=IX,FX,SX
  DO 20 J=IY,FY,SY
    DO 25 K=IZ,FZ,SZ
      Y1=J*3.14159/180.0
      Z1=K*3.14159/180.0
      X=I*SIN(Y1)*COS(Z1)
      Y=I*SIN(Y1)*SIN(Z1)
      Z=I*COS(Y1)
      NVAL=NVAL+1.0
      EX=DIRICHLETVALX(X,Y,Z,FREQ)
      EY=DIRICHLETVALY(X,Y,Z,FREQ)
      EZ=DIRICHLETVALZ(X,Y,Z,FREQ)
      IF (SC.EQ.99) THEN
      HR=EX*SIN(Y1)*COS(Z1)+EY*SIN(Y1)*SIN(Z1)+EZ*COS(Y1)
      HTET=EX*COS(Y1)*COS(Z1)+EY*COS(Y1)*SIN(Z1)-EZ*SIN(Y1)
      HFI=-EX*SIN(Z1)+EY*COS(Z1)+EZ*0.0
      ELSE
      HR=EX
      HTET=EY
      HFI=EZ
      END IF
      CALL CONVTOMAGFAZ(HR)
      CALL CONVTOMAGFAZ(HTET)
      CALL CONVTOMAGFAZ(HFI)
      IF (SCREEN.EQ.2) THEN
 IF (VAR.EQ.1) WRITE(20,110) I,HR,HTET,HFI
 IF (VAR.EQ.2) WRITE(20,110) J,HR,HTET,HFI
 IF (VAR.EQ.3) WRITE(20,110) K,HR,HTET,HFI
 IF (VAR.EQ.4) WRITE(20,130) J,K,HR,HTET,HFI
      END IF
      IF (SCREEN.EQ.1) WRITE(*,100) X,Y,Z,HR,HTET,HFI
 25    CONTINUE
 20   CONTINUE
 10 CONTINUE
 IF (SCREEN.EQ.2) CLOSE(20)
 TYPE *,'OTHER VALUES (y/n)?'
 ACCEPT 120,AA
 IF (AA.EQ.'y') GOTO 5
 100 FORMAT('AT:(',F8.5,',',F8.5,',',F8.5,');Hr=(',E13.5,
     *   ',',F10.5,');Htet=(',E12.5,',',F10.5,');Hphi=(',E12.5,',',
     *   F10.5,').')
 110  FORMAT(1X,F7.3,2X,E10.3,2X,F7.3,2X,E10.3,2X,F7.3,2X,E10.3,2X,F7.2)
 130  FORMAT(1X,F7.3,2X,F7.3,2X,F7.3,2X,E10.3,2X,F7.3,2X,E10.3,2X,
     *  F7.2,2X,E10.3,2X,F7.2)
 120 FORMAT(A2)
 140 FORMAT(A20)
 END
 C
 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 C
```

DERANALRAD.F

```
 C PROGRAM THAT PRINTS ANALYTICAL DERIVATIVE OF FIELD IN THE RADIAL
 C DIRECTION GIVEN FOR THE SOURCE DEFINED IN sourcesube.f.
 C
 IMPLICIT NONE
 C
 COMPLEX H(1,3)
 REAL IX,FX,SX,IY,FY,SY,IZ,FZ,SZ,P(1,3),NOR(3),PI,FREQ
 REAL Y1,Z1,BET,I,J,K
 CHARACTER*2 AA
 INTEGER SCREEN,VAR
 CHARACTER*20 FNAME
 C
 C
 OPEN(UNIT=3,FILE='functions.dat',STATUS='UNKNOWN',ACCESS=
     *       'SEQUENTIAL',FORM='FORMATTED')
 READ(3,*) FREQ
 CLOSE(3)
 PI=ACOS(-1.0)
 BET= 2.0*PI*FREQ*SQRT(8.85E-12*1.25664E-6)
 5 TYPE *,'ENTER OUTPUT VARIABLE(1 FOR SCREEN/2 FOR FILES) :'
 ACCEPT *,SCREEN
 IF (SCREEN.EQ.2) THEN
 TYPE *,'ENTER OUTPUT FILENAME?'
 ACCEPT 140,FNAME
  OPEN(UNIT=20,FILE=FNAME,STATUS='UNKNOWN',ACCESS=
     *       'SEQUENTIAL',FORM='FORMATTED')
  TYPE *,'ENTER COORDINATE THAT VARIES(R/1,THETA/2,PHI/3) :'
  ACCEPT *,VAR
 END IF
 TYPE *,'ENTER RADIUS :'
```

```
ACCEPT *,IX,FX,SX
TYPE *,'ENTER INITIAL THETA, FINAL THETA AND THE STEP :'
ACCEPT *,IY,FY,SY
TYPE *,'ENTER INITIAL PHI, FINAL PHI AND THE STEP :'
ACCEPT *,IZ,FZ,SZ
TYPE *,'DERIVATIVE IN THE R DIRECTION'
DO 10 I=IX,FX,SX
  DO 20 J=IY,FY,SY
    DO 25 K=IZ,FZ,SZ
      Y1=J*3.14159/180.0
      Z1=K*3.14159/180.0
      P(1,1)=I*SIN(Y1)*COS(Z1)
      P(1,2)=I*SIN(Y1)*SIN(Z1)
      P(1,3)=I*COS(Y1)
      NOR(1)=P(1,1)
      NOR(2)=P(1,2)
      NOR(3)=P(1,3)
      CALL GETREALDHDN1(P,H,NOR,1,1,BET)
      CALL CONVTOMAGFAZ(H(1,1))
      CALL CONVTOMAGFAZ(H(1,2))
      CALL CONVTOMAGFAZ(H(1,3))
      IF (SCREEN.EQ.2) THEN
IF (VAR.EQ.1) WRITE(20,110) I,H(1,1),H(1,2),H(1,3)
IF (VAR.EQ.2) WRITE(20,110) J,H(1,1),H(1,2),H(1,3)
IF (VAR.EQ.3) WRITE(20,110) K,H(1,1),H(1,2),H(1,3)
      END IF
      IF (SCREEN.EQ.1) WRITE(*,100)
     *        P(1,1),P(1,2),P(1,3),H(1,1),H(1,2),H(1,3)
25    CONTINUE
20   CONTINUE
10 CONTINUE
IF (SCREEN.EQ.2) CLOSE(20)
TYPE *,'OTHER VALUES (y/n)?'
ACCEPT 120,AA
IF (AA.EQ.'y') GOTO 5
100 FORMAT('AT:(',F8.5,',',F8.5,',',F8.5,');Hr=(',E12.5,
     *  ',',F10.5,');Htet=(',E12.5,',',F10.5,');Hphi=(',E12.5,',',
     *  F10.5,').')
110  FORMAT(1X,F7.3,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2,2X,E10.3,2X,F7.2)
120 FORMAT(A2)
140 FORMAT(A20)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

DFDNCOMP.F

```
C THIS LIST OF SUBROUTINES ARE USED BY MAINCOMP FOR THE IMPLEMENTATION
C OF THE ABC BASED ON BEM. MOST OF THE ROUTINES ARE SIMILAR TO THE ONE
C IN BEMRAD BUT NOT QUITE THE SAME.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETFULLVALONS(BSPYR,SIMP,NOR,NODES,SIZE,CO,ALF,H2,S,NP,
     *  NPYR,PYRIND,NPS,H,DH,HNODE)
INTEGER BSPYR(SIZE),SIMP(SIZE),SIZE,NODES(SIZE,10),CO,ALF(10,8,2)
INTEGER NP,NPYR,PYRIND(SIZE),NPS,HNODE(SIZE,10)
REAL NOR(SIZE,3),H2(NP,4,10),S(NP,4,4)
REAL H(SIZE,NP,10),DH(SIZE,NP,10)
C
IMPLICIT NONE
INTEGER NOD1(10),SMP1,I
REAL ABCD1(4,4),ABCDI1(4,4),NOR1(3)
C
DO 40 I=1,CO
  CALL SELECTDATA(BSPYR(I),I,SIMP,NOR,NODES,SIZE,NOD1,SMP1,
     *  ABCD1,ABCDI1,NOR1)
        CALL GETHDHHIDHI(SMP1,ABCD1,ABCDI1,NOR1,
     *  ALF,H2,S,NOD1,NP,NPYR,PYRIND,NODES,SIZE,NPS,
     *  BSPYR(I),I,H,DH,HNODE)
40 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETHDHHIDHI(SIMP,ABCD1,ABCDI1,NOR,ALF,HS,S,NODE,
     *  NP,NPYR,PYRIND,NODES,SIZE,NPUS,PYR,BS,H,DH,HNODE)
REAL H(SIZE,NP,10),DH(SIZE,NP,10)
INTEGER SIMP,NP,NPYR,PYRIND(SIZE),SIZE,ALF(10,2,8),NODES(SIZE,10)
INTEGER NPUS,PYR,BS,NODE(10),HNODE(SIZE,10)
REAL ABCD1(4,4),ABCDI1(4,4),HS(NP,4,10)
REAL NOR(3),S(NP,4,4)
C
IMPLICIT NONE
REAL P(16,10),GFN(16,10)
REAL P(16,3)
INTEGER ND,I
C
CALL GET3PGLOBSING(ABCDI1,S,P,SIMP,NP,NPUS)
CALL GETGRADH(P,NP,NPUS,GFN,NOR,NPYR,PYRIND,NODES,SIZE,ALF,
     *  PYR,ABCD1)
CALL GETH(HS,SIMP,F,NP,NPUS)
DO 10 ND=1,10
   HNODE(BS,ND)=NODE(ND)
  DO 20 I=1,NPUS
    H(BS,I,ND)=(F(I,ND)/12.56637)*0.03125
20     DH(BS,I,ND)=(GFN(I,ND)/12.56637)*0.03125
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INTEGRAL1(POS,NPOS,K,VAL,VALNOD,NVAL,MXVND,SIMP,ELAREA,
     *  ABCD1,ABCDI1,NOR,H2OS,S,NP,MXFAC,NPUS,NVEC,BS,H,DH,HNODE,
     *  AX,CRF,IN,SIMPLEX)
REAL H(MXFAC,NP,10),DH(MXFAC,NP,10)
INTEGER SIMP,NP,VALNOD(MXVND),MXVND,NVAL,NPOS,IN(6),SIMPLEX
INTEGER NPUS,BS,HNODE(MXFAC,10),AX,CRF(MXVND),MXFAC,IND
REAL ABCD1(4,4),ABCDI1(4,4),NVEC(3),ELAREA
COMPLEX VAL(6,MXVND)
```

```
REAL NOR(3),S(NP,4,4),K,POB(NP,3),H3OS(NP,4,10)
C
IMPLICIT NONE
COMPLEX GG(16,3),G2N(16),GGI(16,3),G2NI(16),DIR(10,3)
COMPLEX GN(16),GNI(16),DUM(10),G2(16,3),G2I(16,3),DUM10ALF(6,10)
REAL P(16,3),PI(16,3),FAC,NORI(3)
INTEGER I,L,ND,ALF,IND1,IND2,IND3
C
FAC=1.0
NORI(1)=NOR(1)
NORI(2)=NOR(2)
NORI(3)=-NOR(3)
IF (AX.EQ.3) FAC=-1.0
CALL INITIALIZE(DUM10ALF,6,10)
DO 10 L=1,NPOS
  CALL GET3PGLOB(ABCDI1,S,P,PI,SIMP,NP,NPUS)
  CALL GETGRADGN(PI,GGI,K,POB(L,1),POB(L,2),POB(L,3),NP,NPUS)
  CALL GETGRADGNNP(PI,G2I,K,POB(L,1),POB(L,2),POB(L,3),NORI,NP,NPUS)
  CALL GETGRADGN(P,GG,K,POB(L,1),POB(L,2),POB(L,3),NP,NPUS)
  CALL GETGRADGNNP(P,G2,K,POB(L,1),POB(L,2),POB(L,3),NOR,NP,NPUS)
  CALL DOTGN(GG,NVEC,GN,NP,NPUS)
  CALL DOTGN(GGI,NVEC,GNI,NP,NPUS)
  CALL DOTGN(G2,NVEC,G2N,NP,NPUS)
  CALL DOTGN(G2I,NVEC,G2NI,NP,NPUS)
  DO 50 ND=1,10
    DUM(ND)=CMPLX(0.0,0.0)
    DO 20 I=1,NPUS
20    DUM(ND)=DUM(ND)+(H(BS,I,ND)*G2N(I)-GN(I)*DH(BS,I,ND))
     *                +FAC*(H(BS,I,ND)*G2NI(I)-GNI(I)*DH(BS,I,ND))
    DO 30 ALF=1,6
30    DUM10ALF(ALF,ND)=DUM10ALF(ALF,ND)+DUM(ND)*H3OS(L,SIMPLEX,IN(ALF))
ccc30     DUM10ALF(ALF,ND)=DUM(ND)
50  CONTINUE
10  CONTINUE
DO 60 ND=1,10
  IND1=HNODE(BS,ND)
  IND2=CRF(IND1)
  DO 70 ALF=1,6
  DUM10ALF(ALF,ND)=DUM10ALF(ALF,ND)*ELAREA
ccccc   DUM10ALF(ALF,ND)=DUM10ALF(ALF,ND)
  VAL(ALF,IND2)=VAL(ALF,IND2)-DUM10ALF(ALF,ND)
c minus sign because Nor was directed inward instead of outward
70 CONTINUE
60 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETGRADH(P,N,NS,FN,NOR,NPYR,PYRIND,NODES,SIZ,ALF,
     * PYR,ABCD)
INTEGER NPYR,SIZ,PYRIND(SIZ),N,NODES(SIZ,10)
INTEGER ALF(10,2,6),NS,PYR
REAL P(N,3),NOR(3),ABCD(4,4),FN(N,10)
C
IMPLICIT NONE
INTEGER I,J,ND
REAL PD(3),S(4),DHN(10)
C
DO 10 I=1,NS
  DO 20 J=1,3
20  PD(J)=P(I,J)
  CALL EVALUATEGH(PD,NOR,NPYR,PYRIND,NODES,DHN,SIZ,ALF,PYR,
     *    ABCD,S)
C MINUS SIGN DEPENDING ON DEFINITION OF NORMAL
  DO 30 ND=1,10
30  FN(I,ND)=-DHN(ND)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EVALUATEGH(PD,NOR,NPYR,PYRIND,NODES,DHN,SIZE,ALF,PYR,
     * ABCD,S)
INTEGER NPYR,SIZE,PYRIND(SIZE),NODES(SIZE,10),ALF(10,2,6),PYR
REAL PD(3),NOR(3),ABCD(4,4),S(4),DHN(10)
C
IMPLICIT NONE
INTEGER I,J,OLPYR,ND,NODE(10)
REAL P(3),P1(3),P2(3),MN,VAL(10)
C
DO 5 ND=1,10
5 DHN(ND)=0.0
DO 10 J=0,2
  DO 20 I=1,3
    P(I)=PD(I)-0.001*J*NOR(I)
    IF (J.EQ.0) P1(I)=P(I)
20  IF (J.EQ.1) P2(I)=P(I)
  OLPYR=PYR
  CALL GETPYR(P,NPYR,PYRIND,SIZE,PYR)
  IF (OLPYR.NE.PYR) THEN
    CALL GETNODEJACOB(ABCD,NODE,PYR,NODES,SIZE)
    WRITE(*,*) 'YOU HAVE NEW PYRAMID THIS IS NOT GONNA WORK
     *    BECAUSE YOU STEP IN DERIVATIVE IS TOO LARGE THAT YOU USE
     *      MORE THAN ONE PYAMID TO EVALUATE DH/DN'
    WRITE(*,*) 'I TRIED TO GET NEW NODES BUT IT WONT WORK!'
  END IF
  CALL GETSPLOC(ABCD,P,S)
  CALL GETVALUE(ALF,S,J,VAL)
  DO 30 ND=1,10
30  DHN(ND)=DHN(ND)+VAL(ND)
10 CONTINUE
MN=SQRT( (P2(1)-P1(1))**2+(P2(2)-P1(2))**2+(P2(3)-P1(3))**2)
DO 40 ND=1,10
40 DHN(ND)=DHN(ND)/MN/2.0
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETVALUE(ALF,S,OP,VAL2)
INTEGER ALF(10,2,6),OP
REAL S(4),VAL2(10)
C
```

```
      IMPLICIT NONE
      INTEGER SP,P1,P2,P3,P4,ND
      REAL VAL1,VAL
C
      DO 10 ND=1,10
        VAL2(ND)=0.0
        VAL=0.0
        DO 20 SP=1,2
          P1=ALF(ND,SP,1)
          P2=ALF(ND,SP,2)
          P3=ALF(ND,SP,3)
          P4=ALF(ND,SP,4)
          VAL1=( (S(1)**P1)*(S(2)**P2)*(S(3)**P3)*(S(4)**P4) )*ALF(ND,SP,5)
20        VAL=VAL+VAL1
        VAL2(ND)=VAL
        IF (OP.EQ.0) VAL2(ND)=-3.0*VAL2(ND)
        IF (OP.EQ.1) VAL2(ND)=4.0*VAL2(ND)
        IF (OP.EQ.2) VAL2(ND)=-VAL2(ND)
10    CONTINUE
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE GETGRADGN(P,GG,K,X,Y,Z,N,NS)
      REAL X,Y,Z,K,P(N,3)
      COMPLEX GG(N,3)
      INTEGER N,NS
C
      IMPLICIT NONE
C
      REAL R
      INTEGER I
      COMPLEX FI,EX,FIP
C
      DO 10 I=1,NS
        R=SQRT( (X-P(I,1))**2+(Y-P(I,2))**2+(Z-P(I,3))**2 )
        EX=CMPLX(0.0,-1.0)*K*R
        FI=CEXP(EX)/R
        FIP=-(FI/(R**2)+CMPLX(0.0,1.0)*K*FI/R)
        GG(I,1)=(X-P(I,1))*FIP
        GG(I,2)=(Y-P(I,2))*FIP
10      GG(I,3)=(Z-P(I,3))*FIP
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE GETGRADGNNP(P,GG,K,X,Y,Z,NOR,N,NS)
      REAL X,Y,Z,K,P(N,3),NOR(3)
      COMPLEX GG(N,3)
      INTEGER N,NS
C
      IMPLICIT NONE
C
      REAL R
      INTEGER I
      COMPLEX FI,EX,FIA,FIB,A,B,IMG,XX,YY,ZZ,XY,XZ,YZ
C
      IMG=CMPLX(0.0,1.0)
      DO 10 I=1,NS
        R=SQRT( (X-P(I,1))**2+(Y-P(I,2))**2+(Z-P(I,3))**2 )
        EX=-IMG*K*R
        FI=CEXP(EX)/R
        A=IMG*K/R+1.0/R**2
        B=(K**2-3.0*IMG*K/R-3.0/R**2)/(R**2)
        FIA=FI*A
        FIB=FI*B
        XX=FIA+(X-P(I,1))**2*FIB
        YY=FIA+(Y-P(I,2))**2*FIB
        ZZ=FIA+(Z-P(I,3))**2*FIB
        XY=(X-P(I,1))*(Y-P(I,2))*FIB
        XZ=(X-P(I,1))*(Z-P(I,3))*FIB
        YZ=(Y-P(I,2))*(Z-P(I,3))*FIB
        GG(I,1)=NOR(1)*XX+NOR(2)*XY+NOR(3)*XZ
        GG(I,2)=NOR(1)*XY+NOR(2)*YY+NOR(3)*YZ
10      GG(I,3)=NOR(1)*XZ+NOR(2)*YZ+NOR(3)*ZZ
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE GETH(H,SP,F,N,NS)
      REAL H(N,4,10),F(N,10)
      INTEGER SP,N,NS
C
      IMPLICIT NONE
C
      INTEGER P,ND
      REAL VAL(10)
C
        DO 10 P=1,NS
          DO 20 ND=1,10
20          F(P,ND)=H(P,SP,ND)
10        CONTINUE
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

DFDNSUBS.F

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE IDENTIFY(MEDF,IDCOD,INNER,BON,J)
      INTEGER MEDF(4),IDCOD,J
      LOGICAL INNER,BON
C
      IMPLICIT NONE
      BON=.FALSE.
      IF (IDCOD.EQ.500000) THEN
        BON=(INNER.AND.(J.EQ.1)).OR.(MEDF(J).GE.IDCOD)
      ELSE IF (IDCOD.EQ.11111) THEN
        BON=(INNER.AND.(J.EQ.1)).OR.(MEDF(J).EQ.IDCOD)
```

```
ELSE IF (IDCOD.EQ.511111) THEN
  BON=(INNER.AND.(J.EQ.1)).OR.(MEDF(J).EQ.IDCOD)
ELSE
  WRITE(*,*) 'IDCOD IS NOT DEFINED'
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE RETURNINNER(MEDF,IDCOD,INNER)
INTEGER MEDF(4),IDCOD
LOGICAL INNER
C
IMPLICIT NONE
INNER=.FALSE.
IF (IDCOD.EQ.500000) THEN
     INNER=((MEDF(1).LT.IDCOD).AND.(MEDF(2).LT.IDCOD).AND.
     *          (MEDF(3).LT.IDCOD).AND.(MEDF(4).LT.IDCOD) )
ELSE IF (IDCOD.EQ.11111) THEN
     INNER=((MEDF(1).NE.IDCOD).AND.(MEDF(2).NE.IDCOD).AND.
     *          (MEDF(3).NE.IDCOD).AND.(MEDF(4).NE.IDCOD) )
ELSE IF (IDCOD.EQ.511111) THEN
     INNER=((MEDF(1).NE.IDCOD).AND.(MEDF(2).NE.IDCOD).AND.
     *          (MEDF(3).NE.IDCOD).AND.(MEDF(4).NE.IDCOD) )
ELSE
  WRITE(*,*) 'IDCOD IS NOT DEFINED'
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE AJOUTEOUNON(X,AJOUT,IDCOD)
INTEGER X,IDCOD
LOGICAL AJOUT
C
IMPLICIT NONE
AJOUT=.FALSE.
IF (IDCOD.EQ.500000) THEN
  AJOUT=(X.GE.IDCOD)
ELSE IF (IDCOD.EQ.11111) THEN
  AJOUT=(X.EQ.IDCOD)
ELSE IF (IDCOD.EQ.511111) THEN
  AJOUT=(X.EQ.IDCOD)
ELSE
  WRITE(*,*) 'IDCOD IS NOT DEFINED'
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPILES MESH INFORMATION IN THE FILE unmesh.dat
CCREATED BY THE MESHER . THE INFORMATION IS COLLECTED ONLY ABOUT THE
CVOLUME DEFINED BY THE TWO BOXES OF DIMENSIONS LXY1,LZ1,LXY2,LZ2.
CPYRN IS THE NUMBER OF FE(TETRAHEDRAL) FOUND IN THE ABOVE VOLUME
CPYRIND IS THE FE NUMBER OF EACH OF THESE TETRAHEDRAL
CNODES IS THE LIST OF THE INDEXES OF THE 10 NODES ON EACH OF THE PYRN FE
CCI IS THE NUMBER OF FACES(SIDES OF TETRAHEDRALS) THAT DEFINE THE CLOSED
CSURFACE OF INTEGRATION.
CNOR IS THE NORMAL VECTOR(X,Y,Z) OF THE CI FACES
CSIMP IS THE NORMAL SIMPLEX OF THE CI FACES
CBSPYR IS THE LIST OF THE FE THAT HAVE A FACE ON THE SURFACE OF
CINTEGRATION(ONLY)
CTHE SUBROUTINE WILL RETURN
CA,B,C,D WHICH ARE THE PLANE PARAMETERS WHOSE INTERSECTION DEFINES THE FE
CABCD,ABCDI ARE THE COEFFICIENTS OF THE SIMPLEX-GLOBAL COORDINATES TRANSFORM
CEQUATIONS
CIDCOD DECIDES WHICH PYRAMIDS WILL BE USED FOR THE INTEGRAL
SUBROUTINE PYRLAYRBAS(LXY1,LZ1,LXY2,LZ2,PYRN,PYRIND,NODES,SIZ,CO,
     *  NOR,SIMP,BSPYR,IDCOD)
REAL LXY1,LZ1,LXY2,LZ2,NOR(SIZ,3)
INTEGER PYRN,PYRIND(SIZ),SIZ,SIMP(SIZ),BSPYR(SIZ)
INTEGER CO,NODES(SIZ,10),IDCOD
C
IMPLICIT NONE
COMMON/JACOB/ ABCD,ABCDI
COMMON/PLANES/ A,B,C,D
INTEGER I,J,K,L,INDP(8),INDEX(4),SCF,EDGE(4,3),POINTS(4,3,2),IN(8)
INTEGER IN10(8),ND(3),SIMPLEX
INTEGER MEDP,MEDF(4),MEDE(4,3),SCE,FIL1,NP,SCP,NODE(10)
REAL ABCD(2000,4,4),ABCDI(2000,4,4),A(2000,4),B(2000,4),C(2000,4)
REAL ABCD1(4,4),ABCDI1(4,4)
REAL D(2000,4),A1(4),B1(4),C1(4),D1(4),DCOR(4,3),AREA,NV(3)
REAL COR(4,3),DUM1(3),DUM2(3),DUM3(3),AP(4),BP(4),CP(4),DP(4)
REAL DUM4(3),DUM5(3),DUM6(3),DUM7(3),DUM8(3),ANG(4),COSINUS
LOGICAL GOOD,PYRVALID,INNER,BON,AJOUT
C
PYRN=0
CO=0
CALL READBASICINFO(NP,SCE,SCF,SCP,FIL1)
DO 10 I=1,SCP
  CALL READINFO(I,NP,SCE,SCF,EDGE,POINTS,MEDP,
     *              MEDF,MEDE,INDP,COR,INDEX,FIL1)
  CALL ONEDIM(COR(1,1),COR(1,2),COR(1,3),DUM1)
  CALL ONEDIM(COR(2,1),COR(2,2),COR(2,3),DUM2)
  CALL ONEDIM(COR(3,1),COR(3,2),COR(3,3),DUM3)
  CALL ONEDIM(COR(4,1),COR(4,2),COR(4,3),DUM4)
  CALL MIDPOI(DUM1,DUM2,DUM5)
  CALL MIDPOI(DUM3,DUM4,DUM6)
  CALL MIDPOI(DUM5,DUM6,DUM7)
  GOOD=PYRVALID(DUM7,LXY1,LXY2,LZ1,LZ2)
  IF (GOOD) THEN
    CALL GETNODES(NODE,NP,INDEX,INDP)
    CALL ALLCOEFFICIENT(COR,A1,B1,C1,D1)
    CALL RETURNINNER(MEDF,IDCOD,INNER)
     DO 25 J=1,4
 CALL IDENTIFY(MEDF,IDCOD,INNER,BON,J)
       IF ( BON ) THEN
         PYRN=PYRN+1
  IF (PYRN.GE.SIZ) WRITE(*,*) 'MAXIMUM NUMBER OF PYRAMIDS TO
    *CONTRIBUTE TO GLOBAL BOUNDARY HAS BEEN EXCEED. PYRN=',PYRN
         PYRIND(PYRN)=I
           CALL INITIALIZE2(IN,8,1)
            CALL INITIALIZE2(IN10,8,1)
         CALL BASE3NODEP(POINTS,J,ND)
```

```
      CALL SPECIALNODE(J,IN,IN10,ND,EDGE,NP,NODE)
   CALL GETNORMALVERTEX(ND,COR,A1,B1,C1,D1,INDEX,SIMPLEX,AREA,DCOR)
      CALL GETUNITNORMAL(DCOR,NV)
      CALL SETUPID(ABCDI1)
      CALL GETABCDINVABCD(A1,B1,C1,D1,ABCD1,ABCDI1)
      DO 45 K=1,4
  DO 55 L=1,4
   ABCD(PYRN,K,L)=ABCD1(K,L)
   ABCDI(PYRN,K,L)=ABCDI1(K,L)
55    CONTINUE
45       CONTINUE
      DO 65 K=1,10
65  NODES(PYRN,K)=NODE(K)
      CALL PLANEPAR3(DUM1,DUM2,DUM3,AP(1),BP(1),CP(1),DP(1))
      CALL PLANEPAR3(DUM1,DUM2,DUM4,AP(2),BP(2),CP(2),DP(2))
     CALL PLANEPAR3(DUM1,DUM3,DUM4,AP(3),BP(3),CP(3),DP(3))
     CALL PLANEPAR3(DUM2,DUM3,DUM4,AP(4),BP(4),CP(4),DP(4))
      CALL DUMP4PROJ(DUM4,AP(1),BP(1),CP(1),DP(1),DUM5)
     CALL DUMP4PROJ(DUM3,AP(2),BP(2),CP(2),DP(2),DUM6)
     CALL DUMP4PROJ(DUM2,AP(3),BP(3),CP(3),DP(3),DUM7)
     CALL DUMP4PROJ(DUM1,AP(4),BP(4),CP(4),DP(4),DUM8)
     CALL DIRECTION(DUM4,DUM5,DUM8)
      CALL DIRECTION(DUM3,DUM6,DUM6)
     CALL DIRECTION(DUM2,DUM7,DUM7)
     CALL DIRECTION(DUM1,DUM8,DUM9)
     ANG(1)=COSINUS(DUM5,AP(1),BP(1),CP(1))
     ANG(2)=COSINUS(DUM6,AP(2),BP(2),CP(2))
     ANG(3)=COSINUS(DUM7,AP(3),BP(3),CP(3))
     ANG(4)=COSINUS(DUM8,AP(4),BP(4),CP(4))
     DO  60 K=1,4
       IF (ANG(K).LT.0.0) THEN
   AP(K)=-AP(K)
BP(K)=-BP(K)
CP(K)=-CP(K)
DP(K)=-DP(K)
   END IF
        IF (ABS(AP(K)).LT.1.0E-08)  AP(K)=0.0
        IF (ABS(BP(K)).LT.1.0E-08)  BP(K)=0.0
        IF (ABS(CP(K)).LT.1.0E-08)  CP(K)=0.0
        IF (ABS(DP(K)).LT.1.0E-08)  DP(K)=0.0
60       CONTINUE
       DO 30 K=1,4
         A(PYRN,K)=AP(K)
         B(PYRN,K)=BP(K)
         C(PYRN,K)=CP(K)
30         D(PYRN,K)=DP(K)
C
  CALL AJOUTEOUNON(MEDF(J),AJOUT,IDCOD)
       IF (AJOUT) THEN
   CO=CO+1
  IF (CO.GE.SIZ) WRITE(*,*) 'MAXIMUM NUMBER OF FACES TO
    *CONTRIBUTE TO GLOBAL BOUNDARY HAS BEEN EXCEED. CO=',CO
   SIMP(CO)=SIMPLEX
   BSPYR(CO)=PYRN
   NOR(CO,1)=NV(1)
   NOR(CO,2)=NV(2)
   NOR(CO,3)=NV(3)
       END IF
      END IF
35      CONTINUE
   END IF
10  CONTINUE
CLOSE(2)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

CSUBROUTINE THAT CHECKS THAT THE PYRAMID (OUT OF ALL POSSIBLE PYRAMIDS) IS IN
CTHE VOLUME SPECIFIED BY LXY1,LXY2,....
LOGICAL FUNCTION PYRVALID(DUM,LXY1,LXY2,LZ1,LZ2)
REAL LXY1,LXY2,LZ1,LZ2,DUM(3)
C
IMPLICIT NONE
REAL LX1,LX2,LY1,LY2
LOGICAL XOUT,YOUT,XIN,YIN,ZOUT,ZIN,XYOUT,XYIN
LOGICAL XIN1,YIN1,ZIN1,ZOUT1,TOP,SIDE
C
PYRVALID=.FALSE.
c DUM(1)=DUM(1)-20.0
LX1=LXY1
LX2=LXY2
LY1=LXY1
LY2=LXY2
XOUT=ABS(DUM(1)).GT.LX1
YOUT=ABS(DUM(2)).GT.LY1
ZOUT=DUM(3).GT.0.0
XYOUT=XOUT.OR.YOUT
XIN=ABS(DUM(1)).LT.LX2
YIN=ABS(DUM(2)).LT.LY2
ZIN=DUM(3).LT.LZ2
XYIN=XIN.OR.YIN
XIN1=ABS(DUM(1)).LT.LX1
YIN1=ABS(DUM(2)).LT.LY1
ZIN1=DUM(3).LT.LZ2
ZOUT1=DUM(3).GT.LZ1
TOP=XIN1.AND.YIN1.AND.ZIN1.AND.ZOUT1
SIDE=XYIN.AND.XYOUT.AND.ZIN.AND.ZOUT
PYRVALID=TOP.OR.SIDE
c DUM(1)=DUM(1)+20.0
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT RETURNS THE 10 NODES OF THE PYRAMID PYR AND THE COEFFICIENTS
COF THE TRANSFORM RELATIONS SIMPLEX TO GLOBAL FOR THE PYRAMID PYR
SUBROUTINE GETNODEJACOB(ABCD,NODE,PYR,NODES,SIZE)
INTEGER SIZE,NODES(SIZE,10),NODE(10),PYR
REAL ABCD(4,4)
C
IMPLICIT NONE
COMMON/JACOB/ ABCD1,ABCDI1
REAL ABCD1(2000,4,4),ABCDI1(2000,4,4)
INTEGER I,J
C
```

```
      DO 10 I=1,10
   10 NODE(I)=NODES(PYR,I)
      DO 20 I=1,4
        DO 30 J=1,4
   30 ABCD(I,J)=ABCD1(PYR,I,J)
   20 CONTINUE
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT RETURNS THE SIMPLEX COORDINATES OF A POINT FROM IT GLOBAL
CCOORDINATES USING ABCD
      SUBROUTINE GETSPLOC(ABCD,P,S)
      REAL P(3),S(4),ABCD(4,4)
C
      IMPLICIT NONE
      INTEGER I
C
      DO 10 I=1,4
   10 S(I)=ABCD(I,1)+ABCD(I,2)*P(1)+ABCD(I,3)*P(2)+ABCD(I,4)*P(3)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT RETURNS PYR THE PYRAMID CONTAINING THE POINT PD
CIT USES THE PARAMETERS OF THE PLANE EQUATIONS WHOSE INTERSECTION FORM THE
CTETRAHEDRAL. IT CHECKS ON WHICH SIDE OF EACH PLANE THE POINT PD IS USING
CTHE FUNCTION 'INSIDE'
      SUBROUTINE GETPYR(PD,NPYR,PYRIND,SIZE,PYR)
      REAL PD(3)
      INTEGER NPYR,PYRIND(SIZE),SIZE
C
      IMPLICIT NONE
      COMMON/PLANES/ A,B,C,D
      REAL A(2000,4),B(2000,4),C(2000,4),D(2000,4)
      LOGICAL INSIDE,VAL(4),SAME
      INTEGER K,L,PYR
C
      SAME=.FALSE.
      IF (PYR.NE.0) THEN
        DO 40 L=1,4
   40   VAL(L)=INSIDE(PD,A(PYR,L),B(PYR,L),C(PYR,L),D(PYR,L))
        IF (VAL(1).AND.VAL(2).AND.VAL(3).AND.VAL(4)) SAME=.TRUE.
      END IF
      IF (.NOT.SAME) THEN
        PYR=0
        DO 30 K=1,NPYR
          DO 50 L=1,4
   50       VAL(L)=INSIDE(PD,A(K,L),B(K,L),C(K,L),D(K,L))
          IF (VAL(1).AND.VAL(2).AND.VAL(3).AND.VAL(4)) PYR=K
          IF (PYR.NE.0) GOTO 35
   30   CONTINUE
   35   CONTINUE
      END IF
        IF (PYR.EQ.0) WRITE(*,*) 'PROBLEM'
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      LOGICAL FUNCTION INSIDE(COOR,AP,BP,CP,DP)
      REAL COOR(3),AP,BP,CP,DP
C
      IMPLICIT NONE
      REAL VAL
      INSIDE=.FALSE.
      VAL=AP*COOR(1)+BP*COOR(2)+CP*COOR(3)+DP
      IF (VAL.GE.-1.0e-6) INSIDE=.TRUE.
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE DOT PRODUCT BETWEEN TWO VETORS
      SUBROUTINE DOTGN(V,N,SCAL,NP,NS)
      COMPLEX V(NP,3),SCAL(NP)
      REAL N(3)
      INTEGER NP,NS
C
      IMPLICIT NONE
C
      INTEGER I
C
      DO 10 I=1,NS
   10 SCAL(I)=V(I,1)*N(1)+V(I,2)*N(2)+V(I,3)*N(3)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTTINE THAT COLLECTS THE SIMPLEX TO GLOBAL COORDINATES TRANSFORM
CEQUATION PARAMETERS FOR A PYRAMID, IT ALSO GETS THE 10 NODES ON THE PYRAMID
CAND THE NORMAL FOR THE FACE OF INTEREST ON THE PYRAMID
CIND IS THE PYRAMYD(TETRAHEDRE) IN QUESTION
      SUBROUTINE SELECTDATA(IND,CI,SIMP,NOR,NODES,SIZ,NOD1,SMP1,
     *  ABCD1,ABCDI1,NOR1)
      INTEGER SIMP(SIZ),SIZ,NODES(SIZ,10),IND,CI
      REAL NOR(SIZ,3)
      INTEGER SMP1,NOD1(10)
      REAL ABCD1(4,4),ABCDI1(4,4),NOR1(3)
C
      IMPLICIT NONE
      COMMON/JACOB/ ABCD,ABCDI
      REAL ABCD(2000,4,4),ABCDI(2000,4,4)
      INTEGER I,J
C
      SMP1=SIMP(CI)
      DO 10 I=1,10
   10   NOD1(I)=NODES(IND,I)
      DO 20 I=1,4
        DO 30 J=1,4
          ABCDI1(I,J)=ABCDI(IND,I,J)
   30   ABCD1(I,J)=ABCD(IND,I,J)
   20 CONTINUE
      DO 40 I=1,3
   40 NOR1(I)=NOR(CI,I)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE COMPUTES THE INVERSE OF MAT AND MAT
```

```
      SUBROUTINE GJTABCDINVABCD(A,B,C,D,MAT,ID)
      REAL A(4),B(4),C(4),D(4),ID(4,4)
C
      IMPLICIT NONE
      INTEGER I
      REAL MAT(4,4),MATI(4,4),EO,INDX(4)
C
      DO 10 I=1,4
        MAT(I,1)=A(I)
        MATI(I,1)=A(I)
        MAT(I,2)=B(I)
        MATI(I,2)=B(I)
        MAT(I,3)=C(I)
        MATI(I,3)=C(I)
        MAT(I,4)=D(I)
10      MATI(I,4)=D(I)
      CALL LUDCMP(MATI,4,4,INDX,EO)
      DO 20 I=1,4
20    CALL LUBKSB(MATI,4,4,INDX,ID(1,I))
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE FOR INVERSION TAKEN FROM NUMERICAL RECIPIES FOR FORTRAN
C
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      DIMENSION A(NP,NP),INDX(N),B(N)
      II=0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11        CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13        CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14    CONTINUE
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

CSUBROUTINE FOR INVERSION TAKEN FROM NUMERICAL RECIPIES FOR FORTRAN
C
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      PARAMETER (NMAX=100,TINY=1.0E-20)
      DIMENSION A(NP,NP),INDX(N),VV(NMAX)
      D=1.0
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11      CONTINUE
        IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
        VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
        IF (J.GT.1) THEN
          DO 14 I=1,J-1
            SUM=A(I,J)
            IF (I.GT.1)THEN
              DO 13 K=1,I-1
                SUM=SUM-A(I,K)*A(K,J)
13            CONTINUE
              A(I,J)=SUM
            ENDIF
14        CONTINUE
        ENDIF
        AAMAX=0.
        DO 16 I=J,N
          SUM=A(I,J)
          IF (J.GT.1)THEN
            DO 15 K=1,J-1
              SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
            A(I,J)=SUM
          ENDIF
          DUM=VV(I)*ABS(SUM)
          IF (DUM.GE.AAMAX) THEN
            IMAX=I
            AAMAX=DUM
          ENDIF
16      CONTINUE
        IF (J.NE.IMAX)THEN
          DO 17 K=1,N
            DUM=A(IMAX,K)
            A(IMAX,K)=A(J,K)
            A(J,K)=DUM
17        CONTINUE
          D=-D
          VV(IMAX)=VV(J)
        ENDIF
        INDX(J)=IMAX
        IF(J.NE.N)THEN
          IF(A(J,J).EQ.0.)A(J,J)=TINY
          DUM=1./A(J,J)
          DO 18 I=J+1,N
            A(I,J)=A(I,J)*DUM
18        CONTINUE
        ENDIF
19    CONTINUE
```

```
      IF(A(N,N).EQ.0.)A(N,N)=TINY
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES A PSEUDO CENTER OF A TRIANGULAR SURFACE DEFINED BY
CDUM1,DUM2,DUM3.
C
SUBROUTINE MIDCENTER(COR,DUM,DUM4)
REAL COR(4,3),DUM(3),DUM4(4)
C
IMPLICIT NONE
INTEGER I
REAL DUM1(3),DUM2(3),DUM3(3),MID(3)
C
DO 10 I=1,3
  DUM1(I)=COR(1,I)
  DUM2(I)=COR(2,I)
  DUM4(I)=COR(4,I)
10   DUM3(I)=COR(3,I)
  CALL MIDPOI(DUM1,DUM2,MID)
  CALL MIDPOI(MID,DUM3,DUM)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION INNERBOUND(DM,D4,V,LXY,LZ)
REAL DM(3),V(3),LXY,LZ,D4(3)
C
IMPLICIT NONE
REAL DLXY,DLZ,MD(3)
C
DLXY=LXY-0.001
DLZ=LZ-0.001
CALL MIDPOI(DM,D4,MD)
INNERBOUND=.FALSE.
IF ((ABS(V(1)).NE.1).XOR.(ABS(V(2)).NE.1).XOR.(ABS(V(3)).NE.1))
     * GOTO 10
IF ( ((ABS(DM(1)).LT.DLXY).AND.(ABS(DM(2)).LT.DLXY)).AND.
     *       (DM(3).LT.DLZ)     ) GOTO 10
IF ( (ABS(MD(1)).GT.DLXY).OR.(ABS(MD(2)).GT.DLXY) ) GOTO 10
IF ( V(3).LT.0.0) GOTO 10
INNERBOUND=.TRUE.
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE WEIGTH DUE TO THE ALPHA FUNCTIONS(INTERPOLATION)
CAT ALL THE COLLOCATION POINT, FOR THE FOUR POSSIBLE FE ORIENTATION AT ALL THE C10 NODES.
C
SUBROUTINE GETH3(ALF,H3,S,N,NS)
INTEGER ALF(10,2,5),N,NS
REAL S(N,4,4),H3(N,4,10)
C
IMPLICIT NONE
INTEGER P,SP,ROW,P1,P2,SPN,P3,P4
REAL VAL,VAL1
C
DO 10 P=1,NS
  DO 20 ROW=1,10
    DO 35 SPN=1,4
      VAL1=0.0
      DO 30 SP=1,2
        P1=ALF(ROW,SP,1)
        P2=ALF(ROW,SP,2)
        P3=ALF(ROW,SP,3)
        P4=ALF(ROW,SP,4)
        VAL=(S(P,1,SPN)**P1)*(S(P,2,SPN)**P2)*(S(P,3,SPN)**P3)
     *          *(S(P,4,SPN)**P4)
        VAL=VAL*ALF(ROW,SP,5)
30        VAL1=VAL1+VAL
      H3(P,SPN,ROW)=VAL1
35    CONTINUE
20    CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT COMPUTES THE SIMPLEX COORDINATES OF THE COLLOCATION POINTS
CFOR ALL FOUR POSSIBLE FACES.(VALID ONLY FOR NS=4 OR NS=1)
CS IS THE MATRIX OF THE COORDINATES
CN IS MAXIMUM COLLOCATION POINT NUMBER
CNS IS NUMBER OF COLLOCATION POINTS
C
SUBROUTINE GETSMPVALNP(S,N,NS)
REAL S(N,4,4)
INTEGER N,NS
C
IMPLICIT NONE
INTEGER SPN,SP,P
REAL S1(100,4),TEMP
C
  DO 20 P=1,NS
     S1(P,4)=0.0
20 CONTINUE
IF (NS.EQ.1) THEN
S1(1,1)=1.0/3.0 !
S1(1,2)=1.0/3.0 !3 LINES FOR NS=1
S1(1,3)=1.0/3.0 !
ELSE IF (NS.EQ.4) THEN
S1(1,1)=1.0/8.0 !!
S1(1,2)=1.0/4.0 !!
S1(1,3)=5.0/8.0 !!
S1(2,1)=1.0/8.0 !!
S1(2,2)=1.0/2.0 !!12 LINES FOR NS=4
S1(2,3)=3.0/8.0 !!
S1(3,1)=3.0/8.0 !!
S1(3,2)=1.0/2.0 !!
S1(3,3)=1.0/8.0 !!
S1(4,1)=5.0/8.0 !!
S1(4,2)=1.0/4.0 !!
S1(4,3)=1.0/8.0 !!
ELSE
```

```
WRITE(*,*) 'GETSMPVALNP IS NOT SET FOR THIS RESOLUTION'
END IF
DO 30 SPN=1,4
  DO 40 P=1,NS
    DO 50 SP=1,4
    S(P,SP,SPN)=S1(P,SP)
50     CONTINUE
40   CONTINUE
30 CONTINUE
DO 60 SPN=1,4
  DO 70 P=1,N
  TEMP=S(P,SPN,SPN)
  S(P,SPN,SPN)=S(P,4,SPN)
70   S(P,4,SPN)=TEMP
60 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE SETS UP THE IDENTITY MATRIX USED BY INVERSION ROUTINES
C
SUBROUTINE SETUPID(ID)
REAL ID(4,4)
C
IMPLICIT NONE
INTEGER I,J
DO 10 I=1,4
  DO 20 J=1,4
20   ID(I,J)=0.0
ID(I,I)=1.0
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT READS ALL THE DIRICHLET VALUES AVAILABLE AND STORES THEM IN
CVECTORS. THIS IS FASTER THAT ACCESSING THE FILE EVERY TIME.
CALLDIR IS THE VALUE OF THE NODE X,Y,Z COMPONENT
CALLDIRN IS THE INDEX OF THE NODE
CMAXDIR IS THE NUMBER OF DIRICHLET NODES
C
SUBROUTINE READDIRV(ALLDIR,ALLDIRN,N,MAXDIR)
INTEGER ALLDIRN(N),MAXDIR,N
COMPLEX ALLDIR(N,3)
C
IMPLICIT NONE
INTEGER J,IND
REAL X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
C
OPEN(UNIT=11,FILE='ordirichlet.dat',STATUS='UNKNOWN',ACCESS=
     * 'SEQUENTIAL',FORM='FORMATTED')
READ(11,*) MAXDIR
DO 10 J=1,MAXDIR
  READ(11,*) IND,X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
      ALLDIRN(IND)=J
ALLDIR(J,1)=CMPLX(X1X,X1Y)
ALLDIR(J,2)=CMPLX(Y1X,Y1Y)
  ALLDIR(J,3)=CMPLX(Z1X,Z1Y)
10 CONTINUE
CLOSE(11)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT GETS THE DIRICHLET VALUES FOR THE 10 NODES ON TETRAHEDRAL PYR
CWHOSE FACE IS BEING INTEGRATED.
C
SUBROUTINE GETDIR10(DIR,NODE,PYR)
INTEGER NODE(10),PYR
COMPLEX DIR(10,3)
C
IMPLICIT NONE
COMMON/DIR/ ALLDIR,ALLDIRN,MAXDIR
COMPLEX ALLDIR(6000,3),DUM
INTEGER I,ALLDIRN(6000),MAXDIR,IND
C
DUM=CMPLX(999.0,999.0)
  DO 10 I=1,10
    DIR(I,1)=DUM
    DIR(I,2)=DUM
    DIR(I,3)=DUM
    IND=ALLDIRN(NODE(I))
    IF (IND.EQ.0) WRITE(*,*) 'ALLDIRN FOR',NODE(I),'IS NOT SET
     *FOR THE PYRAMID',PYR
    DIR(I,1)=ALLDIR(IND,1)
    DIR(I,2)=ALLDIR(IND,2)
    DIR(I,3)=ALLDIR(IND,3)
  IF (DIR(I,1).EQ.DUM) WRITE(*,*) 'OH,OH! IN GETDIR10 I CANT FIND
     *   THE X COMPONENT VALUE OF NODE',NODE(I)
  IF (DIR(I,2).EQ.DUM) WRITE(*,*) 'OH,OH! IN GETDIR10 I CANT FIND
     *   THE Y COMPONENT VALUE OF NODE',NODE(I)
  IF (DIR(I,3).EQ.DUM) WRITE(*,*) 'OH,OH! IN GETDIR10 I CANT FIND
     *   THE Z COMPONENT VALUE OF NODE',NODE(I)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE TO CONVERT THE COLLOCATION POINT COORDINATES IN SIMPLEX TO
CTHEIR GLOBAL COORDINATES USING ABCDI
SUBROUTINE GETJPGLOB(MI,S,P,PI,SPN,N,NS)
REAL MI(4,4),P(N,3),S(N,4,4),PI(N,3)
INTEGER SPN,N,NS
C
IMPLICIT NONE
INTEGER I
C
C CONVERTS THE COORDINATES
CALL GETJPGLOBSING(MI,S,P,SPN,N,NS)
C COMPUTES THE IMAGE WITH RESPECT TO XY PLANE
DO 10 I=1,NS
PI(I,1)=P(I,1)
PI(I,2)=P(I,2)
10 PI(I,3)=-P(I,3)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C
SUBROUTINE GET3PGLOBSING(MI,S,P,SPN,N,NS)
REAL MI(4,4),P(N,3),S(N,4,4)
INTEGER SPN,N,NS
C
IMPLICIT NONE
INTEGER I
C
DO 10 I=1,NS
P(I,1)=MI(2,1)*S(I,1,SPN)+MI(2,2)*S(I,2,SPN)
     *          +MI(2,3)*S(I,3,SPN)+MI(2,4)*S(I,4,SPN)
P(I,2)=MI(3,1)*S(I,1,SPN)+MI(3,2)*S(I,2,SPN)
     *          +MI(3,3)*S(I,3,SPN)+MI(3,4)*S(I,4,SPN)
P(I,3)=MI(4,1)*S(I,1,SPN)+MI(4,2)*S(I,2,SPN)
     *          +MI(4,3)*S(I,3,SPN)+MI(4,4)*S(I,4,SPN)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

MESHCOMPSG.F

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ALFFIELDCONT(CONTS,ALFF,VOL)
REAL ALFF(10,10)
COMPLEX CONTS(10,10),VOL
C
IMPLICIT NONE
INTEGER I,J
DO 10 I=1,10
  DO 20 J=1,10
    CONTS(I,J)=CONTS(I,J)+ALFF(I,J)*VOL
20   CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETCONTS
     * (CONTS,BCONTS,II,NODE,NP,SCE,SCF,EPSILON,FIL1,ALLDIR,ALLNOI,
     * ALLDIRN,ALLNOIN,MAXN,MAXDIR,MAXNOI,DIR,DIRN,AX,BET,
     * BSPYR,SIMP,NOR,NODES,MXFAC,CO,ALF,H2OS,S,SOB,NPI,NPYR,PYRIND,
     * NPS,NPOS,H,DH,HNODE,VAL,VALNOD,NVAL,CRF,MXVND,ABC,
     * NZERO,AS,AI,AJ,N,NZ,TOTNONZERO,VALNODSG)
COMPLEX CONTS(10,10),BCONTS(10),EPSILON(10),DIR(10)
COMPLEX ALLDIR(N),ALLNOI(N),VAL(6,MXVND)
INTEGER II,NODE(10),NP,SCE,SCF,FIL1,DIRN(10),AX,ALLDIRN(N),ABC
INTEGER ALLNOIN(N),MAXDIR,MAXNOI,MAXN,BSPYR(MXFAC),SIMP(MXFAC),NPOS
INTEGER NODES(MXFAC,10),MXFAC,CO,ALF(10,6,2),NPI,NPYR,PYRIND(MXFAC)
INTEGER NPS,MXVND,VALNOD(MXVND),HNODE(MXFAC,10),NVAL,CRF(MXVND),IN(6)
REAL H(MXFAC,NPI,10),DH(MXFAC,NPI,10),H2OS(NPI,4,10)
REAL BET,NOR(MXFAC,3),S(NPI,4,4),SOB(NPI,4,4)
REAL*8 AS(NZ)
INTEGER NZERO,AI(NZ),AJ(N),N,NZ,TOTNONZERO,VALNODSG(MXVND)

C
IMPLICIT NONE
COMMON/OPERATEUR/ LAPLACIAN,TRANSLAPLACIAN,GRADGRAD,ALFF
     *         ,NORM,FIELDN,TRANSFIELDN,TANLAPLAC,SURFALFF
REAL LAPLACIAN(10,10,1,4),TRANSLAPLACIAN(10,10,1,4),NORM(10,10,1,4)
c REAL LAPLACIAN(10,10,1,4),TRANSLAPLACIAN(10,10,1,4),NORM(10,4,1,1)
REAL ALFF(10,10,1,1),GRADGRAD(10,10,4,4),FIELDN(10,10,1,4)
REAL TRANSFIELDN(10,10,1,4),TANLAPLAC(10,10,4,4),SURFALFF(10,10,1,4)
INTEGER EDGE(4,3),MEDP,MEDF(4),MEDE(4,3),INDP(6),SIMPLEX
INTEGER POINTS(4,3,2),J,INDEX(4),MEDIA,TYP(3),ND(3),IN10(6)
INTEGER NOD1(10),SMP1,I,B12ND(12)
REAL ABCD1(4,4),ABCDI1(4,4),ELAREA,VALSG(12,2000)
REAL COOR(4,3),A(4),B(4),C(4),D(4),DCOOR(4,3),NOR1(3)
REAL VOL,AREA,VOLUME,NVEC(3),R,FI,TET,POB(16,3)
COMPLEX VOL1,KSQUARE,NSID(6)
LOGICAL OUTER
C
CALL READINFO(II,NP,SCE,SCF,EDGE,POINTS,
     *                MEDP,MEDF,MEDE,INDP,COOR,INDEX,FIL1)
VOL=VOLUME(COOR)
CALL GETNODES(NODE,NP,INDEX,INDP)
CALL ALLCOEFFICIENT(COOR,A,B,C,D)
VOL1=-VOL
CALL GRADGRADCONT(CONTS,B,C,D,GRADGRAD,VOL1)
MEDIA=MEDP-MEDP/10
KSQUARE=EPSILON(MEDIA)
VOL1=VOL*EPSILON(MEDIA)
CALL ALFFIELDCONT(CONTS,ALFF,VOL1)
CALL GETDIR(DIR,DIRN,ALLDIR,ALLDIRN,MAXN,MAXDIR,NODE)
DO 10 J=1,4
  CALL BOUNDARYTYPE(MEDF(J),3,TYP(1))
  CALL BOUNDARYTYPE(MEDF(J),2,TYP(2))
  CALL BOUNDARYTYPE(MEDF(J),1,TYP(3))
      CALL INITIALIZE3(IN,6,1)
      CALL INITIALIZE3(IN10,6,1)
  OUTER=(MEDF(J).GT.9999).AND.(MEDF(J).LT.500000)
  IF ((TYP(AX).EQ.2).OR.(TYP(AX).EQ.4)) THEN
    CALL BASE3NODEP(POINTS,J,ND)
    CALL SPECIALNODE(J,IN,IN10,ND,EDGE,NP,NODE)
    CALL GETNORMALVERTEX(ND,COOR,A,B,C,D,INDEX,SIMPLEX,AREA,DCOOR)
    CALL GETNSID(NSID,IN,ALLNOI,ALLNOIN,N,MAXNOI)
    CALL GETCONTALFNORM(BCONTS,NORM,NSID,AREA,SIMPLEX,IN10)
  ELSE IF ((TYP(AX).EQ.0).AND.OUTER) THEN
CCC THIS THE BEM BASED ABC
   IF (ABC.EQ.1) THEN
     CALL BASE3NODEP(POINTS,J,ND)
     CALL SPECIALNODE(J,IN,IN10,ND,EDGE,NP,NODE)
     CALL GETNORMALVERTEX(ND,COOR,A,B,C,D,INDEX,SIMPLEX,AREA,DCOOR)
     CALL GETUNITNORMAL(DCOOR,NVEC)
     CALL SETUPID(ABCDI1)
     CALL GETABCDINVABCD(A,B,C,D,ABCD1,ABCDI1)
           CALL GETPOINTOFINT(ABCDI1,SOB,NPOS,NPI,POB,SIMPLEX)
     ELAREA=AREA/NPOS
     CALL INITIALIZVAL(VAL,NVAL,MXVND)
```

```
      DO 40 I=1,CO
        CALL SELECTDATA(BSPYR(I),I,SIMP,NOR,NODES,MXFAC,NOD1,SMP1,
     *       ABCD1,ABCDI1,NOR1)
             CALL INTEGRAL1(POS,NPOS,BET,VAL,VALNOD,NVAL,MXVND,SMP1,
     *       ELAREA,ABCD1,ABCDI1,NOR1,H2OS,S,NPI,MXFAC,NPS,NVEC,I,H,
     *       DH,HNODE,AX,CRF,IN10,SIMPLEX)
40      CONTINUE
        CALL CONVVALVALSG(VAL,VALSG,VALNODSG,MXVND,NVAL,NP,
     *       SCE,AX,IN,B12ND)
      CALL ADVALSGTOS(VALSG,VALNODSG,MXVND,NVAL,AS,AI,AJ,N,NZ,
     *       NZERO,B12ND,TOTNONZERO,SCE,NP)
     ELSE IF ((ABC.GT.1).AND.(ABC.LT.3)) THEN
CCC THIS IS SOMMERFIELD,BT1 AND BT2
      CALL BASE3NODEF(POINTS,J,ND)
      CALL SPECIALNODE(J,IN,IN10,ND,EDGE,NP,NODE)
      CALL GETNORMALVERTEX(ND,COOR,A,B,C,D,INDEX,SIMPLEX,AREA,DCOOR)
      CALL GETUNITNORMAL(DCOOR,NVEC)
      CALL GETAVERAGER(DCOOR,R,FI,TET)
      CALL GETCONTSOMMER(CONTS,SURFALFF,AREA,SIMPLEX,R,FI,TET,
     *       BET,AX,ABC)
    END IF
  END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE GETPOINTOFINT(ABCDI1,SOB,NPOS,NP,POB,SIMP)
REAL ABCDI1(4,4),POB(NP,3),SOB(NP,4,4)
INTEGER NPOS,NP,SIMP
C
IMPLICIT NONE
CALL GET3PGLOBSING(ABCDI1,SOB,POB,SIMP,NP,NPOS)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INITIALIZVAL(VAL,NVAL,MXVND)
COMPLEX VAL(0,MXVND)
INTEGER MXVND,NVAL
C
IMPLICIT NONE
INTEGER I,J
C
DO 20 I=1,NVAL
  DO 25 J=1,6
25   VAL(J,I)=CMPLX(0.0,0.0)
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETCONTSOMMER(CONTS,SURFALFF,AREA,SIMP,R,FI,TET,BET,AX,ABC)
COMPLEX CONTS(10,10)
REAL SURFALFF(10,10,1,4),AREA,BET,R,FI,TET
INTEGER SIMP,AX,ABC
C
IMPLICIT NONE
INTEGER I,J
COMPLEX TEMP,IM,FAC
C
IM=CMPLX(0.0,1.0)
  IF (ABC.EQ.3) FAC=-( IM*BET-(BET**2)*R ) !BT2
IF (ABC.EQ.3) FAC=-( IM*BET+1/R ) !BT1
IF (ABC.EQ.4) FAC=-( IM*BET ) !SOMMER
DO 10 I=1,10
  DO 20 J=1,10
    TEMP=SURFALFF(I,J,1,SIMP)*AREA*FAC
20    CONTS(I,J)=CONTS(I,J)+TEMP
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETCONTBAYLISS(CONTS,B,C,D,TANLAP,ALFF,N,R,SX,AREA,K2,IN)
COMPLEX CONTS(10,10),K2
REAL AREA,R,ALFF(10,10,1,4),B(4),C(4),D(4),TANLAP(10,10,4,4),N(3)
INTEGER  SX,IN(8)
C
IMPLICIT NONE
COMPLEX K,FR,GR,IMJ,GRADF(3),NIX
REAL COEF
INTEGER J,M,L,Q
C
IMJ=CMPLX(0.0,1.0)
NIX=CMPLX(0.0,0.0)
K=CSQRT(K2)
FR=( -2.0+2*K2*R**2-3.0*IMJ*K*R ) / ( 3*R+2*IMJ*K*R**2 )
GR=R**2 / ( 2*IMJ*K*R**2+3*R )
DO 10 J=1,10
  DO 20 Q=1,10
    DO 30 M=1,3
      GRADF(M)=NIX
      DO 40 L=1,4
IF (M.EQ.1) COEF=C(L)**2+D(L)**2
IF (M.EQ.2) COEF=B(L)**2+D(L)**2
IF (M.EQ.3) COEF=B(L)**2+C(L)**2
40 GRADF(M)=GRADF(M)+COEF*TANLAP(J,Q,L,SX)
30     GRADF(M)=( GRADF(M)*GR+FR*ALFF(J,Q,1,SX) )*AREA
   CONTS(J,Q)=CONTS(J,Q)+GRADF(1)*N(1)+GRADF(2)*N(2)+GRADF(3)*N(3)
c write(*,100) gradf(1),gradf(2),gradf(3),n(1),n(2),n(3),Sx
20    CONTINUE
10 CONTINUE
c write(*,*) '------XXXXXXXXXXXXXXXXXXXXX----------'
100 format(6f12.8,112)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETAVERAGER(COR,R,FI,TET)
REAL COR(4,3),R,FI,TET
C
IMPLICIT NONE
```

```
REAL DUMP1(3),DUMP2(3),DUMP3(3),MID1(3),MID2(3)
C
CALL ONEDIM(COR(1,1),COR(1,2),COR(1,3),DUMP1)
CALL ONEDIM(COR(2,1),COR(2,2),COR(2,3),DUMP2)
CALL ONEDIM(COR(3,1),COR(3,2),COR(3,3),DUMP3)
CALL MIDPOI(DUMP1,DUMP2,MID1)
CALL MIDPOI(MID1,DUMP3,MID2)
CALL CONVCARTOSPE(MID2(1),MID2(2),MID2(3),R,FI,TET)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CONVCARTOSPE(CX,CY,CZ,R,FI,TET)
REAL CX,CY,CZ,R,FI,TET
C
IMPLICIT NONE
C
  R=SQRT(CX**2+CY**2+CZ**2)
  TET=ACOS(CZ/R)
  IF (R.EQ.0) TET=0.0
  FI=ATAN(CY/CX)
  IF (CX.EQ.0) FI=0.0
  IF ((CX.EQ.0).AND.(CY.GT.0)) FI=1.570796327
  IF ((CX.EQ.0).AND.(CY.LT.0)) FI=-1.570796327
  IF ((CX.GT.0).AND.(CY.EQ.0)) FI=0.0
  IF ((CX.LT.0).AND.(CY.EQ.0)) FI=3.1415926
  IF ((CX.LT.0).AND.(CY.LT.0)) FI=FI+3.1415926
  IF ((CX.LT.0).AND.(CY.GT.0)) FI=FI+3.1415926
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETCONTALFNORM(BCONTS,NORM,NSID,AREA,SIMPLEX,IN10)
COMPLEX BCONTS(10),NSID(6)
REAL NORM(10,10,1,4),AREA
c REAL NORM(10,4,1,1),AREA
INTEGER SIMPLEX,IN10(6)
C
IMPLICIT NONE
INTEGER K,J,L,K1
DO 10 J=1,6
K=IN10(J)
  DO 20 L=1,6
    K1=IN10(L)
C     IF (NSID(J).NE.999.0)
c          BCONTS(K)=BCONTS(K)-NORM(K1,SIMPLEX,1,1)*AREA*NSID(L)
C           BCONTS(L)=BCONTS(L)-NORM(L,SIMPLEX,1,1)*AREA*NSID(J)
           BCONTS(K)=BCONTS(K)-NORM(K,K1,1,SIMPLEX)*AREA*NSID(L)
20  CONTINUE
10 CONTINUE
END
CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC
SUBROUTINE GETCONTFIELDNORM(CONTS,FIELDN,AREA,B,C,D,IN,NV)
INTEGER IN(6)
COMPLEX CONTS(10,10)
REAL FIELDN(10,10,1,4),AREA,NV(3),FACX,FACY,FACZ,B(4),C(4),D(4)
C
IMPLICIT NONE
INTEGER J,K,L
DO 10 J=1,10
  DO 20 K=1,10
    FACX=0.0
    FACY=0.0
    FACZ=0.0
    DO 30 L=1,4
    FACX=FACX+FIELDN(J,K,1,L)*B(L)
    FACY=FACY+FIELDN(J,K,1,L)*C(L)
30    FACZ=FACZ+FIELDN(J,K,1,L)*D(L)
20  CONTS(J,K)=CONTS(J,K)+(FACX*NV(1)+FACY*NV(2)+FACZ*NV(3))*AREA
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETDIR(DIR,DIRN,ALLDIR,ALLDIRN,N,ND,NODE)
INTEGER ALLDIRN(N),NODE(10),DIRN(10),ND,N
COMPLEX DIR(10),ALLDIR(N)
C
IMPLICIT NONE
INTEGER I,J
C
J=1
DO WHILE (J.LE.ND)
  DO 10 I=1,10
  IF (ALLDIRN(J).EQ.NODE(I)) THEN
  IF (REAL(ALLDIR(J)).NE.999.0) THEN
  DIR(I)=ALLDIR(J)
  DIRN(I)=I
END IF
    END IF
10   CONTINUE
  J=J+1
END DO
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE READDIR(ALLDIR,ALLDIRN,N,MAXDIR,FIL2,AX)
INTEGER FIL2,ALLDIRN(N),MAXDIR,N,AX
COMPLEX ALLDIR(N)
C
IMPLICIT NONE
INTEGER J,IND
REAL X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
C
REWIND(FIL2)
READ(FIL2,*) MAXDIR
DO 10 J=1,MAXDIR
  READ(FIL2,*) IND,X1X,X1Y,Y1X,Y1Y,Z1X,Z1Y
  IF (AX.EQ.1)  THEN
```

```
ALLDIR(J)=CMPLX(X1X,X1Y)
      ALLDIRN(J)=IND
END IF
   IF (AX.EQ.2) THEN
ALLDIR(J)=CMPLX(Y1X,Y1Y)
  ALLDIRN(J)=IND
END IF
 IF (AX.EQ.3) THEN
  ALLDIR(J)=CMPLX(Z1X,Z1Y)
  ALLDIRN(J)=IND
END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


SUBROUTINE BUILDS(A,AI,AJ,RHS,N,NZ,SIZE,NZERO,FIL2,AX,ALLDIR,
   * ALLNOI,ALLDIRN,ALLNOIN,MAXN,
   * BSPYR,SIMP,NOR,NODES,MXFAC,CO,ALF,H2OS,SS,SOB,NPI,EPSILON,BET,
   * NPYR,PYRIND,NPS,NPOS,H,DH,HNODE,ABC,SPC,UPD)
REAL*8 RHS(N),A(NZ)
REAL NOR(MXFAC,3),SS(NPI,4,4),SOB(NPI,4,4),BET
COMPLEX ALLDIR(MAXN),ALLNOI(MAXN),EPSILON(10)
INTEGER AI(NZ),AJ(N),SIZE,FIL2,NZERO,AX,MAXN,ALLNOIN(MAXN),NPOS
INTEGER ALLDIRN(MAXN),NODES(MXFAC,10),MXFAC,CO,ALF(10,5,2),NPI,NPYR
INTEGER PYRIND(MXFAC),SIMP(MXFAC),BSPYR(MXFAC),NPS
REAL H(MXFAC,NPI,10),DH(MXFAC,NPI,10),H2OS(NPI,4,10)
INTEGER HNODE(MXFAC,10),ABC,NZ,N,SPC,UPD
C
IMPLICIT NONE
COMMON/OPERATEUR/ LAPLACIAN,TRANSLAPLACIAN,GRADGRAD,ALFF
   *        ,NORM,FIELDN,TRANSFIELDN,TANLAPLAC,SURFALFF
INTEGER SCP,I,NODE(10),NP,SCE,SCF,FIL1
REAL LAPLACIAN(10,10,1,4),TRANSLAPLACIAN(10,10,1,4),NORM(10,10,1,4)
c REAL LAPLACIAN(10,10,1,4),TRANSLAPLACIAN(10,10,1,4),NORM(10,4,1,1)
REAL ALFF(10,10,1,1),FIELDN(10,10,1,4),GRADGRAD(10,10,4,4)
REAL TRANSFIELDN(10,10,1,4),TANLAPLAC(10,10,4,4),SURFALFF(10,10,1,4)
COMPLEX CONTS(10,10),BCONTS(10),DIR(10),DIRN(10),VAL(6,2000)
REAL SCONTS(20,20),BSG(20),DIRSG(20)
INTEGER NODESG(20),DIRNSG(20),VALNOD(2000),TOTNONZERO
INTEGER MAXDIR,MAXNOI,NVAL,CRF(2000),MXVND,VALNODSG(2000)
C
TOTNONZERO=0
MXVND=2000 !MAXIMUM NODES IN ABC EXPRESSION VAL
CALL READBASICINFO(NP,SCE,SCF,SCP,FIL1)
CALL OPERATORS(ALF,1)
CALL INITIALIZE4D(RHS,N,1)
CALL READDIR(ALLDIR,ALLDIRN,MAXN,MAXDIR,FIL2,AX)
CALL READNOI(ALLNOI,ALLNOIN,MAXN,MAXNOI,FIL2,AX)
DO 5 I=1,CO
5 CALL CREATVALNOD(VALNOD,NVAL,I,HNODE,CRF,MXVND,MXFAC,VALNODSG,NP,SCE)
     CALL SETAAIAJ(A,AJ,AI,SCE,NP,N,NZ,NZERO,SPC,AX,UPD,VALNODSG,
    * NVAL,MXVND,ABC)
DO 10 I=1,SCP
  CALL INITIALIZE2(DIRN,10,1)
  CALL INITIALIZE2(DIRNSG,20,1)
  CALL INITIALIZE2(NODESG,20,1)
  CALL INITIALIZE(DIR,10,1)
  CALL INITIALIZE(DIRSG,20,1)
  CALL INITIALIZE(CONTS,10,10)
  CALL INITIALIZE(BCONTS,10,1)
  CALL INITIALIZE4(SCONTS,20,20)
  CALL INITIALIZE4(BSG,20,1)
  CALL GETCONTS(CONTS,BCONTS,I,NODE,NP,SCE,SCF,EPSILON,FIL1,ALLDIR,
    *    ALLNOI,ALLDIRN,ALLNOIN,MAXN,MAXDIR,MAXNOI,DIR,DIRN,AX,BET,
    *    BSPYR,SIMP,NOR,NODES,MXFAC,CO,ALF,H2OS,SS,SOB,NPI,NPYR,PYRIND,
    *    NPS,NPOS,H,DH,HNODE,VAL,VALNOD,NVAL,CRF,MXVND,ABC,
    *    NZERO,A,AI,AJ,N,NZ,TOTNONZERO,VALNODSG)
  CALL CONVCONTSCONTS(CONTS,BCONTS,SCONTS,BSG,NODE,NODESG,DIRN,
    *    DIRNSG,DIR,DIRSG,NP,SCE,AX)
  CALL ADSCONTSTOS(RHS,BSG,NODESG,DIRNSG,SCONTS,DIRSG,SIZE,A,
    *    AI,AJ,N,NZ,NZERO,SCE,NP,TOTNONZERO)
  Write(*,100) I,scp,totnonzero,nzero
c   if (i.eq.550) goto 80
10 CONTINUE
100 FORMAT('DOING PYRAMID ',I6,' OUT OF ',I6,' PYRAMIDS. NZERO=',
    * I10,2x,I10)
CLOSE(FIL1)
WRITE(*,*) 'JUST FINNISHED PYRAMIDS FOR THIS RUN'
80 CALL CLEANAAIAJ(A,AJ,AI,SCE,NP,N,NZ,NZERO)
IF ((AX.EQ.1).AND.(UPD.EQ.1))
    * CALL STORESPACING(NZERO,AI,AJ,N,NZ,SCE,NP)
c stop
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE STORESPACING(NZERO,AI,AJ,N,NZ,SCE,NP)
INTEGER NZERO,NZ,AI(NZ),AJ(N),SCE,NP,N
C
IMPLICIT NONE
INTEGER I,REK
OPEN(UNIT=16,FILE='spacing.dat',STATUS='UNKNOWN',
    * ACCESS='DIRECT',FORM='UNFORMATTED',RECL=1,RECORDTYPE='FIXED')
DO 10 I=1,(SCE+NP)*2
  REK=I
10 WRITE(16,REC=REK) AJ(I)
REK=REK+1
WRITE(16,REC=REK) NZERO
DO 20 I=1,NZERO
  REK=REK+1
20 WRITE(16,REC=REK) AI(I)
CLOSE(16)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE SETAAIAJ(A,AJ,AI,SCE,NP,N,NZ,NZERO,SPC,AX,UPD,NOD,
    * NVAL,MX,ABC)
INTEGER AJ(N),AI(NZ),N,NZERO,SCE,NP,SPC,AX,UPD,NZ,NOD(MX),NVAL,MX,ABC
REAL*8 A(NZ)
C
IMPLICIT NONE
```

```
C
INTEGER I,TOT,J,REK,ESPA,K
C
TOT=(SCE+NP)*2
NZERO=0
DO 10 I=1,NZ
10 A(I)=0.0D0
IF ((SPC.EQ.999999).OR.(AX.NE.1).AND.(UPD.EQ.1)) THEN
  OPEN(UNIT=16,FILE='spacing.dat',STATUS='UNKNOWN',
     *    ACCESS='DIRECT',FORM='UNFORMATTED',RECL=1,RECORDTYPE='FIXED')
  DO 40 I=1,TOT
    REK=I
40  READ(16,REC=REK) AJ(I)
  REK=REK+1
  READ(16,REC=REK) NZERO
  DO 50 I=1,NZERO
    REK=REK+1
50  READ(16,REC=REK) AI(I)
  CLOSE(16)
ELSE
  DO 20 I=1,TOT
    ESPA=SPC
    NZERO=NZERO+1
    AJ(I)=NZERO
    AI(NZERO)=I
    DO 60 K=1,NVAL*2
60    IF ((NOD(K).EQ.I).AND.(ABC.EQ.1)) ESPA=1250
    DO 30 J=1,ESPA
      NZERO=NZERO+1
30    AI(NZERO)=-999
20  CONTINUE
END IF
AJ(TOT+1)=NZERO+1
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CLEANAAIAJ(A,AJ,AI,SCE,NP,N,NZ,NZERO)
INTEGER AJ(N),AI(NZ),N,NZERO,SCE,NP,NZ
REAL*8 A(NZ)
C
IMPLICIT NONE
C
INTEGER I,TOT,J,NONZERO,START,FIN,SHIFTDEC,NLAST
C
NONZERO=0
TOT=(SCE+NP)*2
SHIFTDEC=0
NLAST=0
DO 5 I=AJ(TOT),NZERO
5 IF (A(I).NE.0.0) NLAST=NLAST+1
NLAST=NLAST+AJ(TOT)
DO 10 I=1,TOT
  START=AJ(I)
  AJ(I)=AJ(I)-SHIFTDEC
  FIN=AJ(I+1)-1
  IF (I.EQ.TOT) FIN=NLAST
  J=START-1
  DO WHILE( J.LT.FIN )
    J=J+1
    IF (AI(J).NE.-999) THEN
      NONZERO=NONZERO+1
      AI(NONZERO)=AI(J)
      A(NONZERO)=A(J)
    ELSE
      SHIFTDEC=SHIFTDEC+1
    END IF
  END DO
10 CONTINUE
NZERO=NONZERO
AJ(TOT+1)=NZERO+1
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ADVALSGTOS(VALSG,VALNODSG,MXVND,NVAL,A,AI,AJ,N,NZ,
     *  NZERO,B12,TOTNONZERO,SCE,NP)
INTEGER VALNODSG(MXVND),NZ,AI(NZ),AJ(N),MXVND,NVAL,NZERO,B12(12)
INTEGER TOTNONZERO,SCE,NP,N
REAL VALSG(12,MXVND)
REAL*8 A(NZ)
C
IMPLICIT NONE
INTEGER I,INDX,INDY,J,K,WHERE,NEW,COLSTART,DEL
C
  DO 40 K=1,(2*NVAL)
     INDY=VALNODSG(K)
     COLSTART=AJ(INDY)
     DEL=AJ(INDY+1)-COLSTART-1
    DO 50 J=1,12
     IF (VALSG(J,K).NE.0) THEN
       INDX=B12(J)
WHERE=0
NEW=1
DO 60 I=COLSTART,(COLSTART+DEL)
  IF (AI(I).EQ.INDX) THEN
    WHERE=I
    NEW=0
  END IF
  IF (NEW.EQ.0) GOTO 60
50 CONTINUE
60  IF (WHERE.EQ.0) TOTNONZERO=TOTNONZERO+1
IF ((WHERE.NE.0).AND.(A(WHERE).EQ.0.0))
     &        TOTNONZERO=TOTNONZERO+1
IF (TOTNONZERO.GE.NZ) THEN
WRITE(*,*)
    *'NEXT PASS THE TOTAL SIZE ALLOCATED FOR A(NONZERO) WILL EXCEEDED'
WRITE(*,*) 'IF I AM NOT STOPED I M GONNA CRASH'
WRITE(*,*) 'TOTNONZERO=',TOTNONZERO
END IF
IF (NEW.EQ.1) THEN
  DO 70 I=(COLSTART+DEL),COLSTART,-1
70  IF (AI(I).EQ.-999) WHERE=I
```

```
  IF (WHERE.EQ.0) WHERE=COLSTART+DEL
  IF (AI(WHERE).NE.-999) CALL
     *          SHIFTANDINSERT(A,AI,AJ,N,NZ,NZERO,WHERE,SCE,NP)
  AI(WHERE)=INDX
END IF
      IF (ABS(VALSG(J,K)-999.999).GT.2.0) THEN
        A(WHERE)=A(WHERE)+VALSG(J,K)
      ELSE
        A(WHERE)=1.0
END IF
    END IF
50    CONTINUE
40  CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ADSCONTSTOS(RHS,BSG,NODESG,DIRNSG,SCONTS,DIRSG,SIZE,A,
     *  AI,AJ,N,NZ,NZERO,SCE,NP,TOTNONZERO)
INTEGER NODESG(20),DIRNSG(20),N,AI(NZ),AJ(N),SIZE,NZERO,SCE,NP
INTEGER TOTNONZERO,NZ
REAL SCONTS(20,20),BSG(20),DIRSG(20)
REAL*8 RHS(N),A(NZ)
C
IMPLICIT NONE
INTEGER I,INDX,INDY,J,K,WHERE,NEW,COLSTART,DEL
C
DO 20 J=1,20
      RHS(NODESG(J))=RHS(NODESG(J))+BSG(J)
  IF (DIRNSG(J).NE.0) THEN
    DO 30 K=1,20
30    SCONTS(J,K)=0.0
    SCONTS(J,J)=999.999
    RHS(NODESG(J))=DIRSG(DIRNSG(J))
  END IF
  IF (NODESG(J).GT.SIZE) SIZE=NODESG(J)
20  CONTINUE
  DO 40 K=1,20
     INDY=NODESG(K)
     COLSTART=AJ(INDY)
     DEL=AJ(INDY+1)-COLSTART-1
    DO 50 J=1,20
     IF (SCONTS(J,K).NE.0) THEN
       INDX=NODESG(J)
WHERE=0
NEW=1
DO 60 I=COLSTART,(COLSTART+DEL)
  IF (AI(I).EQ.INDX) THEN
    WHERE=I
    NEW=0
  END IF
  IF (NEW.EQ.0) GOTO 80
60 CONTINUE
80   IF (WHERE.EQ.0) TOTNONZERO=TOTNONZERO+1
IF ((WHERE.NE.0).AND.(A(WHERE).EQ.0.0))
     *          TOTNONZERO=TOTNONZERO+1
IF (TOTNONZERO.GE.NZ) THEN
WRITE(*,*)
    *'NEXT PASS THE TOTAL SIZE ALLOCATED FOR A(NONZERO) WILL EXCEEDED'
WRITE(*,*) 'IF I AM NOT STOPED I M GONNA CRASH'
WRITE(*,*) 'TOTNONZERO=',TOTNONZERO
END IF
IF (NEW.EQ.1) THEN
  DO 70 I=(COLSTART+DEL),COLSTART,-1
70   IF (AI(I).EQ.-999) WHERE=I
  IF (WHERE.EQ.0) WHERE=COLSTART+DEL
  IF (AI(WHERE).NE.-999) CALL
     *          SHIFTANDINSERT(A,AI,AJ,N,NZ,NZERO,WHERE,SCE,NP)
  AI(WHERE)=INDX
END IF
      IF (ABS(SCONTS(J,K)-999.999).GT.2.0) THEN
        A(WHERE)=A(WHERE)+SCONTS(J,K)
      ELSE
        A(WHERE)=1.0
END IF
    END IF
50    CONTINUE
40  CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE SHIFTANDINSERT(A,AI,AJ,N,NZ,NZERO,WHERE,SCE,NP)
INTEGER AI(NZ),AJ(N),N,NZERO,WHERE,SCE,NP,NZ
REAL*8 A(NZ)
C
IMPLICIT NONE
C
INTEGER I,TOT
C
TOT=(SCE+NP)*2
IF ((NZ-NZERO).LE.1) WRITE(*,*) 'A VECTOR IS FULL, I M GONNA CRASH'
DO 10 I=NZERO,WHERE,-1
  A(I+1)=A(I)
10   AI(I+1)=AI(I)
DO 20 I=1,TOT
20 IF (AJ(I).GT.WHERE) AJ(I)=AJ(I)+1
NZERO=NZERO+1
WHERE=WHERE+1
A(WHERE)=0.0D0
AJ(TOT+1)=NZERO+1
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CREATVALNOD(VALNOD,NVAL,BS,HNODE,CRF,MXVND,MXFAC,NODSG,
     *  NP,SCE)
INTEGER MXVND,CRF(MXVND),VALNOD(MXVND),NVAL,BS,HNODE(MXFAC,10),MXFAC
INTEGER NODSG(MXVND),NP,SCE
C
IMPLICIT NONE
INTEGER I,IND,INDC,TOT,OFFSET,J
C
```

```
IF (BS..:Q.1) THEN
  DO 20 I=1,MXVND
20   CRF(I)=0
  NVAL=0
END IF
DO 10 I=1,10
  IND=HNODE(BS,I)
  IF (CRF(IND).EQ.0) NVAL=NVAL+1
  IF (NVAL.GE.MXVND) WRITE(*,*) 'MAXIMUM NUMBER OF NODES IN GLOBAL
     * BOUNDARY (MXVND) IS EXCEED. NVAL=',NVAL
  IF (CRF(IND).EQ.0) VALNOD(NVAL)=IND
  IF (CRF(IND).EQ.0) CRF(IND)=NVAL
10 CONTINUE
INDC=0
TOT=NP+SCE
DO 30 I=1,2
  OFFSET=TOT*(I-1)
  DO 40 J=1,NVAL
    IND=J+INDC
    NODSG(IND)=VALNOD(J)+OFFSET
40   CONTINUE
  INDC=IND
30 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CONVVALVALSG(VAL,VALSG,NODSG,MXVND,NVAL,NP,SCE,AX,
     * B6ND,B12ND)
COMPLEX VAL(6,MXVND)
REAL VALSG(12,MXVND)
INTEGER MXVND,NODSG(MXVND)
INTEGER NP,SCE,AX,NVAL,B6ND(6),B12ND(12)
C
IMPLICIT NONE
INTEGER I,J,IND,INDC,TOT,OFFSET
INTEGER INDCX,INDCY,INDX,INDY,INDXI,INDYI
REAL VALR,VALI
C
INDC=0
TOT=NP+SCE
DO 10 I=1,2
  OFFSET=TOT*(I-1)
  DO 20 J=1,NVAL
    IND=J+INDC
    IF (J.LE.6) B12ND(J+6)=B6ND(J)+OFFSET
    IF (J.LE.6) B12ND(J)=B6ND(J)
20   CONTINUE
  INDC=IND
10 CONTINUE
C
INDCX=0
INDCY=0
  DO 40 I=1,6
    INDX=I+INDCX
    INDXI=INDX+6
    DO 50 J=1,NVAL
      INDY=J+INDCY
      INDYI=INDY+NVAL
      VALR=REAL(VAL(I,J))
      VALI=AIMAG(VAL(I,J))
      VALSG(INDX,INDY)=VALR
      VALSG(INDXI,INDYI)=VALR
      VALSG(INDX,INDYI)=-VALI
      VALSG(INDXI,INDY)=VALI
50     CONTINUE
40   CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CONVCONTBCONTS(CONTS,BCONTS,SCONTS,BSG,NODE,NODESG,DIRN,
     * DIRNSG,DIR,DIRSG,NP,SCE,AX)
COMPLEX CONTS(10,10),BCONTS(10),DIR(10)
REAL SCONTS(20,20),BSG(20),DIRSG(20)
INTEGER NODE(10),NODESG(20),NP,SCE,DIRN(10),DIRNSG(20),AX
C
IMPLICIT NONE
INTEGER I,J,IND,INDC,TOT,VAL,OFFSET
INTEGER INDCX,INDCY,INDX,INDY,INDXI,INDYI
REAL VALR,VALI
C
INDC=0
TOT=NP+SCE
DO 10 I=1,2
  OFFSET=TOT*(I-1)
  DO 20 J=1,10
    IND=J+INDC
    VAL=NODE(J)+OFFSET
    IF (MOD(I,2).EQ.0) DIRSG(IND)=AIMAG(DIR(J))
    IF (MOD(I,2).NE.0) DIRSG(IND)=REAL(DIR(J))
    IF (DIRN(J).NE.0) DIRNSG(IND)=DIRN(J)+INDC
    NODESG(IND)=VAL
20   CONTINUE
  INDC=IND
10 CONTINUE
C
INDCX=0
INDCY=0
  DO 40 I=1,10
    INDX=I+INDCX
    INDXI=INDX+10
    DO 50 J=1,10
      INDY=J+INDCY
      INDYI=INDY+10
      VALR=REAL(CONTS(I,J))
      VALI=AIMAG(CONTS(I,J))
      SCONTS(INDX,INDY)=VALR
      SCONTS(INDXI,INDYI)=VALR
      SCONTS(INDX,INDYI)=-VALI
      SCONTS(INDXI,INDY)=VALI
50     CONTINUE
    BSG(INDX)=REAL(BCONTS(I))
```

```
   BSG(INDXI)=AIMAG(BCONTS(I))
40  CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INITIALIZE(RHS,N1,N2)
COMPLEX RHS(N1,N2)
INTEGER N1,N2
C
IMPLICIT NONE
COMPLEX NIX
INTEGER I,J
C
NIX=CMPLX(0.0,0.0)
DO 10 I=1,N1
  DO 20 J=1,N2
20   RHS(I,J)=NIX
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INITIALIZE4(RHS,N1,N2)
REAL RHS(N1,N2)
INTEGER N1,N2
C
IMPLICIT NONE
INTEGER I,J
C
DO 10 I=1,N1
  DO 20 J=1,N2
20   RHS(I,J)=0.0
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INITIALIZE4D(RHS,N1,N2)
REAL RHS(N1,N2)
INTEGER N1,N2
C
IMPLICIT NONE
INTEGER I,J
C
DO 10 I=1,N1
  DO 20 J=1,N2
20   RHS(I,J)=0.0
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE READNOI(ALLNOI,ALLNOIN,MAXN,MAXNOI,FIL2,AX)
INTEGER ALLNOIN(MAXN),MAXN,MAXNOI,FIL2,AX
COMPLEX ALLNOI(MAXN)
C
IMPLICIT NONE
INTEGER ND,J,IND
REAL X1,X2,Y1,Y2,Z1,Z2
COMPLEX X(3)
C
REWIND(FIL2)
READ(FIL2,*) ND
READ(FIL2,100)
READ(FIL2,*) MAXNOI
DO 10 J=1,MAXNOI
  READ(FIL2,*) IND,X1,X2,Y1,Y2,Z1,Z2
  X(1)=CMPLX(X1,X2)
  X(2)=CMPLX(Y1,Y2)
  X(3)=CMPLX(Z1,Z2)
  ALLNOI(J)=X(AX)
  ALLNOIN(J)=IND
10 CONTINUE
100 FORMAT(<ND-1>(/))
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETNSID(NSID,IN,ALLNOI,ALLNOIN,N,MAXNOI)
INTEGER IN(6),ALLNOIN(N),N,MAXNOI
COMPLEX NSID(6),ALLNOI(N)
C
IMPLICIT NONE
INTEGER I,J
COMPLEX VAL
C
DO 8 I=1,6
8   NSID(I)=CMPLX(999.0,0.0)
I=1
VAL=CMPLX(999.0,0.0)
DO WHILE (I.LE.MAXNOI)
  DO 10 J=1,6
        IF (IN(J).EQ.ALLNOIN(I)) NSID(J)=ALLNOI(I)
10   CONTINUE
  I=I+1
END DO
DO 30 I=1,6
IF (NSID(I).EQ.CMPI X(999.0,0.0)) THEN
      WRITE(*,*) 'NODE ',IN(I),' IS SUPPOSED TO BE A NEUMANN VALUE.'
  WRITE(*,*) 'COULD NOT FIND IT! CHECK IT OUT, FOR ME LIFE GOES ON.'
END IF
30 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE SOLWRITER(BSIDE,N,SIZE,AX)
REAL*8 BSIDE(N)
INTEGER N,SIZE,AX
C
IMPLICIT NONE
```

```
COMPLEX VAL
INTEGER I,BLK,REK,OFFREK
C
OPEN (UNIT=2,FILE='unsolution.dat',STATUS='UNKNOWN',ACCESS='DIRECT'
     *   ,FORM='UNFORMATTED',RECL=2,RECORDTYPE='FIXED')
C
BLK=SIZE/2
IF (AX.EQ.1) THEN
  DO 10 I=1,BLK
    VAL=DCMPLX(BSIDE(I),BSIDE(I+BLK))
        WRITE(2,REC=I) VAL
10   CONTINUE
ELSE IF (AX.EQ.2) THEN
  OFFREK=BLK
  DO 20 I=1,BLK
    REK=OFFREK+I
    VAL=DCMPLX(BSIDE(I),BSIDE(I+BLK))
    WRITE(2,REC=REK) VAL
20   CONTINUE
ELSE IF (AX.EQ.3) THEN
  OFFREK=BLK*2
  DO 30 I=1,BLK
    REK=OFFREK+I
    VAL=DCMPLX(BSIDE(I),BSIDE(I+BLK))
    WRITE(2,REC=REK) VAL
30   CONTINUE
  END IF
CLOSE(2)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
SUBROUTINE VFILLGUESS(SIZ,SOL,N,AX)
REAL*8 SOL(N)
INTEGER SIZ,N,AX
C
IMPLICIT NONE
INTEGER  I,BLK,II,OFFREK,REK
COMPLEX VAL
C
OPEN(UNIT=4,FILE='unsolution.dat',STATUS='UNKNOWN',
     *    ACCESS='DIRECT',FORM='UNFORMATTED',RECL=2,RECORDTYPE='FIXED')
BLK=SIZ/2
IF (AX.EQ.1) OFFREK=0
IF (AX.EQ.2) OFFREK=BLK
IF (AX.EQ.3) OFFREK=SIZ
DO 10 I=1,BLK
 REK=OFFREK+I
 READ(4,REC=REK) VAL
 II=BLK+I
 SOL(I)=REAL(VAL)
 SOL(II)=AIMAG(VAL)
10 CONTINUE
CLOSE(4)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE VFILLDIR(SIZ,SOL,RHS,N)
REAL*8 SOL(N),RHS(N)
INTEGER SIZ,N
C
IMPLICIT NONE
INTEGER  I
C
DO 10 I=1,SIZ
 SOL(I)=RHS(I)
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE REORDERNODES(NODES,BSIDE,SIZ,N)
COMPLEX BSIDE(N)
INTEGER NODES(N),N,SIZ
C
IMPLICIT NONE
INTEGER I,TEMP,ROW,IR
COMPLEX DUM
C
  DO 15 ROW=1,SIZ
    DO 20 I=1,SIZ
20      IF (NODES(I).EQ.ROW) IR=I
    DUM=BSIDE(ROW)
    BSIDE(ROW)=BSIDE(IR)
    BSIDE(IR)=DUM
    TEMP=NODES(ROW)
    NODES(ROW)=NODES(IR)
    NODES(IR)=TEMP
15   CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

MESHER3DSUB.F

```
SUBROUTINE MESHWRITER(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,EXTRAPYR)
INTEGER NP,SCP,SCF,SCE,PID(30000),EDGE(60000,3),FACE(65000,4),NPS
INTEGER PYR(40000,5),EXTRAPYR(40000,8)
REAL P(30000,3)
C
IMPLICIT NONE
  INTEGER OFFREC,REK,I,IN1,IN2,IN3
OPEN(UNIT=1,FILE='edge.dat',STATUS='UNKNOWN')
OPEN(UNIT=2,FILE='mesh.dat',STATUS='UNKNOWN',ACCESS='SEQUENTIAL',
     *      FORM='FORMATTED')
OPEN(UNIT=3,FILE='unmesh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
     *      FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
WRITE(3,FMT=100) NPS,'  THE NUMBER OF POINTS ON THE SURFACE'
```

```
      WRITE(2,FMT=100) NP,'    THE NUMBER OF ALL POINTS'
      WRITE(2,FMT=100) SCP,'    THE NUMBER OF PYRAMIDS'
      WRITE(2,FMT=100) SCF,'    THE NUMBER OF FACES'
      WRITE(2,FMT=100) SCE,'    THE NUMBER OF EDGES'
      WRITE(2,FMT=120) '------ON THE SURFACE------'
      WRITE(3,REC=1) NPS
      WRITE(3,REC=2) NP
      WRITE(3,REC=3) SCP
      WRITE(3,REC=4) SCF
      WRITE(3,REC=5) SCE
      OFFREC=5
      DO 60 I=1,NPS
        REK=OFFREC+I
        WRITE(2,110) P(I,1),P(I,2),P(I,3),PID(I)
   60   WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
      OFFREC=REK
      WRITE(2,FMT=120) '------INSIDE SURFACE------'
      DO 70 I=NPS+1,NP
        REK=OFFREC+I-NPS
        WRITE(2,110) P(I,1),P(I,2),P(I,3),PID(I)
   70   WRITE(3,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
      WRITE(2,FMT=120) '------EDGES------'
      OFFREC=REK
      DO 20 I=1,SCE
        REK=OFFREC+I
      WRITE(2,FMT=105) I,EDGE(I,1),EDGE(I,2),EDGE(I,3)
   20 WRITE(3,REC=REK) EDGE(I,1),EDGE(I,2),EDGE(I,3)
      OFFREC=REK
      WRITE(2,FMT=120) '---FACES------'
      DO 30 I=1,SCF
        REK=OFFREC+I
      WRITE(2,FMT=115) I,FACE(I,1),FACE(I,2),FACE(I,3),FACE(I,4)
   30 WRITE(3,REC=REK) FACE(I,1),FACE(I,2),FACE(I,3),FACE(I,4)
      OFFREC=REK
      WRITE(2,FMT=120) '---PYRAMIDS----'
      DO 40 I=1,SCP
        REK=OFFREC+I
      WRITE(2,FMT=130) PYR(I,1),PYR(I,2),PYR(I,3),PYR(I,4),
     *      PYR(I,5),EXTRAPYR(I,1),EXTRAPYR(I,2),EXTRAPYR(I,3),
     *      EXTRAPYR(I,4),EXTRAPYR(I,5),EXTRAPYR(I,6)
   40 WRITE(3,REC=REK) PYR(I,1),PYR(I,2),PYR(I,3),PYR(I,4),
     *      PYR(I,5),EXTRAPYR(I,1),EXTRAPYR(I,2),EXTRAPYR(I,3),
     *      EXTRAPYR(I,4),EXTRAPYR(I,5),EXTRAPYR(I,6)
C
      WRITE(1,*) SCF
      DO 50 I=1,SCF
        CALL BASE3NODE(EDGE,FACE,I,IN1,IN2,IN3)
        WRITE(1,*) P(IN1,1)
        WRITE(1,*) P(IN1,2)
        WRITE(1,*) P(IN1,3)
        WRITE(1,*) P(IN2,1)
            WRITE(1,*) P(IN2,2)
        WRITE(1,*) P(IN2,3)
        WRITE(1,*) P(IN3,1)
        WRITE(1,*) P(IN3,2)
   50   WRITE(1,*) P(IN3,3)
C
   90 FORMAT(<NP>(/))
  100  FORMAT(I5,(A))
  110 FORMAT(3X,F9.5,3X,F9.5,3X,F9.5,3X,I7)
  105 FORMAT(X,I6,3X,I6,3X,I6,3X,I6)
  115 FORMAT(X,I6,3X,I6,3X,I6,3X,I6,3X,I6,3X,I6)
  120 FORMAT((A))
  130  FORMAT(1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,I6,1X,
     *          I6,1X,I6)
C
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE POINTREADER(NPS,NPV,PID,P)
      INTEGER NPS,NPV,PID(30000)
      REAL P(30000,3)
C
      IMPLICIT NONE
      INTEGER REK,OFFREC,I,SCP,SCF,SCE,NP
C
      OPEN(UNIT=2,FILE='unmsh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
     *      FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
C
      READ(2,REC=1) NPS
      READ(2,REC=2) NP
      READ(2,REC=3) SCP
      READ(2,REC=4) SCF
      READ(2,REC=5) SCE
      NPV=NP-NPS
      OFFREC=5
      DO 10 I=1,NPS
        REK=OFFREC+I
   10 READ(2,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
      OFFREC=REK
      OFFREC=OFFREC-NPS
      DO 20 I=NPS+1,NP
        REK=OFFREC+I
   20 READ(2,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
C
      CLOSE(2)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE DETERMINANT(MAT,N,NN,DET)
      INTEGER N,NN
      REAL MAT(N,NN),DET
C
      IMPLICIT NONE
      INTEGER I,ROW,COL,K,J
      REAL COFAC,DMAT(10,10),DET1
      DET1=0.0
      DET=0.0
      IF (NN.GT.2) THEN
        DO 10 I=1,NN
          COFAC=MAT(1,I)*(-1)**(I+1)
```

```
   DO 15 K=2,NN
    ROW=K-1
    DO 20 J=1,NN-1
     COL=J
IF (J.GE.I) COL=J+1
20      DMAT(ROW,J)=MAT(K,COL)
15    CONTINUE
 CALL DETERMINANT(DMAT,10,NN-1,DET1)
 DET=DET+DET1*COFAC
10 CONTINUE
ELSE
 DET=MAT(1,1)*MAT(2,2)-MAT(2,1)*MAT(1,2)
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION DIST(DUM1,DUM2)
REAL DUM1(3),DUM2(3)
C
DIST=SQRT( (DUM1(1)-DUM2(1))**2+(DUM1(2)-DUM2(2))**2+
     *           (DUM1(3)-DUM2(3))**2                    )
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MIDPOI(DUM1,DUM2,DUM3)
REAL DUM1(3),DUM2(3),DUM3(3)
C
INTEGER I
DO 10 I=1,3
10 DUM3(I)=(DUM1(I)+DUM2(I))/2
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ONEDIM(P1,P2,P3,DUM)
REAL P1,P2,P3,DUM(3)
C
DUM(1)=P1
DUM(2)=P2
DUM(3)=P3
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ONEDIM4(P1,P2,P3,P4,DUM)
REAL P1,P2,P3,P4,DUM(4)
C
DUM(1)=P1
DUM(2)=P2
DUM(3)=P3
DUM(4)=P4
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MAKEMAT3(DUM1,DUM2,DUM3,MAT)
REAL DUM1(3),DUM2(3),DUM3(3),MAT(3,3)
C
INTEGER I
DO 10 I=1,3
 MAT(1,I)=DUM1(I)
 MAT(2,I)=DUM2(I)
10  MAT(3,I)=DUM3(I)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MAKEMAT4(DUM1,DUM2,DUM3,DUM4,MAT)
REAL DUM1(4),DUM2(4),DUM3(4),DUM4(4),MAT(4,4)
C
INTEGER I
DO 10 I=1,4
 MAT(1,I)=DUM1(I)
 MAT(2,I)=DUM2(I)
 MAT(3,I)=DUM3(I)
10  MAT(4,I)=DUM4(I)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
INTEGER FUNCTION FACEID(P1,P2,P3)
INTEGER P1,P2,P3
C
IMPLICIT NONE
INTEGER D1(6),D2(6),D3(6),DP1,DP2,DP3,VAL,I,J,FACD
DP1=P1
DP2=P2
DP3=P3
DO 10 I=1,6
 VAL=INT(DP1/10)*10
 D1(I)=DP1-VAL
10 DP1=VAL/10
 DO 20 I=1,6
 VAL=INT(DP2/10)*10
 D2(I)=DP2-VAL
20 DP2=VAL/10
 DO 30 I=1,6
 VAL=INT(DP3/10)*10
 D3(I)=DP3-VAL
30 DP3=VAL/10
FACD=0
DO 40 I=1,6
 J=I-1
 IF ((D1(I).EQ.1).OR.(D2(I).EQ.1).OR.(D3(I).EQ.1)) THEN
   IF (D1(I).EQ.4) D1(I)=1
   IF (D2(I).EQ.4) D2(I)=1
   IF (D3(I).EQ.4) D3(I)=1
```

```
  ELSE IF ((D1(I).EQ.3).OR.(D2(I).EQ.2).OR.(D3(I).EQ.2)) THEN
    IF (D1(I).EQ.4) D1(I)=2
    IF (D2(I).EQ.4) D2(I)=2
    IF (D3(I).EQ.4) D3(I)=2
   END IF
  IF ( (D1(I).EQ.D3(I)).AND.(D2(I).EQ.D3(I)) )
     *       FACD=FACD+D1(I)*(10**J)
40 CONTINUE
IF ((D1(6).EQ.5).AND.(D2(6).EQ.5).AND.(D3(6).EQ.5)) FACD=FACD+500000
FACEID=FACD
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION DISTP2PLAN(DCEN,DAP,DBP,DCP,DDP)
REAL DCEN(3),DAP,DBP,DCP,DDP
C
REAL DEN,NUM
DEN=SQRT( DAP**2+DBP**2+DCP**2)
NUM=DCEN(1)*DAP+DCEN(2)*DBP+DCEN(3)*DCP+DDP
DISTP2PLAN=ABS(NUM/DEN)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE PLANEPAR3(DUM1,DUM2,DUM3,DAP,DBP,DCP,DDP)
REAL DUM1(3),DUM2(3),DUM3(3),DAP,DBP,DCP,DDP
C
IMPLICIT NONE
REAL DM1(3),DM2(3),DM3(3),MAT(3,3)
INTEGER I
C
DO 10 I=1,3
  DM1(I)=DUM1(I)
  DM2(I)=DUM2(I)
10   DM3(I)=DUM3(I)
CALL MAKEMAT3(DM1,DM2,DM3,MAT)
CALL DETERMINANT(MAT,3,3,DDP)
DDP=-DDP
DM1(3)=1.0
DM2(3)=1.0
DM3(3)=1.0
CALL MAKEMAT3(DM1,DM2,DM3,MAT)
CALL DETERMINANT(MAT,3,3,DCP)
DM1(2)=DUM1(3)
DM2(2)=DUM2(3)
DM3(2)=DUM3(3)
CALL MAKEMAT3(DM1,DM2,DM3,MAT)
CALL DETERMINANT(MAT,3,3,DBP)
DBP=-DBP
DM1(1)=DUM1(2)
DM2(1)=DUM2(2)
DM3(1)=DUM3(2)
CALL MAKEMAT3(DM1,DM2,DM3,MAT)
CALL DETERMINANT(MAT,3,3,DAP)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
SUBROUTINE CENTERSPHERE(DUM1,DUM2,DUM3,DUM4,CENT)
REAL DUM1(3),DUM2(3),DUM3(3),DUM4(3),CENT(3)
C
IMPLICIT NONE
REAL DM1(4),DM2(4),DM3(4),DM4(4),MAT(4,4)
REAL DAS,DDS,DES,DFS
INTEGER I
LOGICAL COPL,COPLANAR
C
COPL=COPLANAR(DUM1,DUM2,DUM3,DUM4)
IF (.NOT.(COPL)) THEN
  DO 10 I=1,3
    DM1(I)=DUM1(I)
    DM2(I)=DUM2(I)
    DM3(I)=DUM3(I)
10    DM4(I)=DUM4(I)
  DM1(4)=1.0
  DM2(4)=1.0
  DM3(4)=1.0
  DM4(4)=1.0
  CALL MAKEMAT4(DM1,DM2,DM3,DM4,MAT)
  CALL DETERMINANT(MAT,4,4,DAS)
  DM1(1)=DUM1(1)**2+DUM1(2)**2+DUM1(3)**2
  DM2(1)=DUM2(1)**2+DUM2(2)**2+DUM2(3)**2
  DM3(1)=DUM3(1)**2+DUM3(2)**2+DUM3(3)**2
  DM4(1)=DUM4(1)**2+DUM4(2)**2+DUM4(3)**2
  CALL MAKEMAT4(DM1,DM2,DM3,DM4,MAT)
  CALL DETERMINANT(MAT,4,4,DDS)
  DDS=-DDS
  DM1(2)=DUM1(1)
  DM2(2)=DUM2(1)
  DM3(2)=DUM3(1)
  DM4(2)=DUM4(1)
  CALL MAKEMAT4(DM1,DM2,DM3,DM4,MAT)
  CALL DETERMINANT(MAT,4,4,DES)
  DM1(3)=DUM1(2)
  DM2(3)=DUM2(2)
  DM3(3)=DUM3(2)
  DM4(3)=DUM4(2)
  CALL MAKEMAT4(DM1,DM2,DM3,DM4,MAT)
  CALL DETERMINANT(MAT,4,4,DFS)
  DFS=-DFS
  CENT(1)=-DDS/DAS/2
  CENT(2)=-DES/DAS/2
  CENT(3)=-DFS/DAS/2
ELSE
  CENT(1)=9999.9
  CENT(2)=9999.9
  CENT(3)=9999.9
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C
SUBROUTINE PYRAMIDGENERATOR(NP,NPV,SCF,SCE,SCP,P,PID,EDGE,FACE,PYR)
INTEGER NPV,SCF,SCE,SCP,FACE(65000,4),EDGE(50000,3),PYR(40000,5)
INTEGER PID(30000),NP
REAL P(30000,3)
C
IMPLICIT NONE
REAL DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3),VOLUME,VOL,RAT
REAL AP,BP,CP,DP,DIST,MID1(3),COOR(4,3),COSINUS
REAL MIN1,RATREF,MINREF1
REAL DIR1(3),DIR2(3),SURFACE,AREA,NEWP(3)
INTEGER VICIN(60),NV,NO,SCOLD,SCFOL,SCPOL,K,MAX
INTEGER I,J,SC,LSTAK(65000),SCE1,SCE2,SCE3,SCF4,F1,OLDSCP
INTEGER IN1,IN2,IN3,OCCUR,SCF1,SCF2,SCF3,IND,II,PYRCOM,NODE(4)
INTEGER NPYR,VICPYR(400),NEW,SNGFAS(65000),SNGFASC,ORDPOINT(30000,30)
INTEGER VICEDG(1000),VICFAC(2000),NEDG,NFAC,VICIN1(60),NV1
LOGICAL CANTUSE,PYRLAP,PYRAMOVERLAP,NOGD2
LOGICAL DONTUSE,NOGD1,NOTVALID,CENTERVALIDV,NOTVALID1,CENTERVALIDV1
CHARACTER*24 FDATE
C
C
NO=0
LSTAK(1)=1
SC=1
c CALL FILLSTAK(LSTAK,SC,SCF)
SNGFASC=0
VOL=0.0
SCE=0
SCF=0
MAX=30
CALL ORDER(ORDPOINT,NP,NPV,P,MAX)
WRITE(*,*) 'I JUST FINISHED ORDER'
WRITE(*,*) FDATE()
120 DO WHILE (SC.GT.0)
  SCF1=LSTAK(1)
  CALL UPDATESTAK(SC,LSTAK)
  DONTUSE=.FALSE.
100  IF (.NOT.DONTUSE) THEN
   CALL BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
   CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
   CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
   CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
   CALL PLANEPAR3(DUMP1,DUMP2,DUMP3,AP,BP,CP,DP)
   CALL GETCORNER(DUMP1,DUMP2,DUMP3,IN1,IN2,IN3,IND,MID1)
   CALL VICINITY(VICIN,NV1,IND,MAX,ORDPOINT)
   NV=30
   IF (MAX.LT.NV) NV=MAX
c this is the distance between 2 points
15    RATREF=0.25
   MINREF1=10000.0
   IND=0
   CALL GETVICPYR(VICPYR,NPYR,EDGE,FACE,PYR,SCP,SCF,SCE,SCF1,
    *     NFAC,VICFAC,VICIN,NV1,P)
   CALL FACEVICINITY(NPYR,VICPYR,PYR,NFAC,VICFAC)
   CALL EDGEVICINITY(NFAC,VICFAC,FACE,NEDG,VICEDG)
   CALL OCCURENCEPVIC(SCF1,NPYR,OCCUR,PYR,PYRCOM,VICPYR)
   DO 30 II=1,NV
    I=VICIN(II)
    IF ((IN1.EQ.I).OR.(IN2.EQ.I).OR.(IN3.EQ.I)) GOTO 30
    CALL ONEDIM(P(I,1),P(I,2),P(I,3),DUMP4)
c the last 3 parameters change with size of problem
    NOTVALID=CENTERVALIDV(EDGE,FACE,SCF1,DUMP4,P,0.5,0.5,1.0)
    IF (NOTVALID) GOTO 30
    PYRLAP=PYRAMOVERLAP(EDGE,FACE,PYR,AP,BP,CP,DP,SCF1,PYRCOM,I,P)
    IF (PYRLAP) GOTO 30
    CALL INSIDEPYR(IN1,IN2,IN3,I,DUMP1,DUMP2,DUMP3,DUMP4,
    *     P,NV1,VICIN1,NO)
    IF (NO.NE.0) GOTO 30
    CALL OVERLAPWITHFACE(FACE,EDGE,VICEDG,VICFAC,NEDG,NFAC,SCF1,
    *     I,P,NOGD1,NOGD2,NV,VICIN)
    IF (NOGD1) GOTO 30
    IF (NOGD2) GOTO 30
     CALL GETNORMALTOFACE(DUMP1,DUMP2,DUMP3,DUMP4,DIR1)
    DIR1(1)=-DIR1(1)
    DIR1(2)=-DIR1(2)
    DIR1(3)=-DIR1(3)
    CALL DIRECTION(DUMP4,MID1,DIR2)
    RAT=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
    MIN1=DIST(DUMP4,MID1)
    IF ((RAT+0.000001).LT.RATREF) GOTO 30
c     IF ((RAT.LT.RATREF).AND.(MIN1.GE.MINREF1)) GOTO 30
    IF (MIN1.GT.(1.5*MINREF1)) GOTO 30
    CALL NOOVERLAP
    *     (EDGE,FACE,SCF1,DUMP4,P,PYR,SCP,CANTUSE,VICPYR,NPYR)
    IF (CANTUSE) GOTO 30
   IND =I
   RATREF=RAT
   MINREF1=MIN1
30    CONTINUE
   SCOLD=SC
   IF (IND.NE.0) THEN
   CALL EXISTENCEEDGE(IN1,IN2,IND,NEDG,SCE1,SCE2,SCE3,VICEDG,EDGE)
   CALL CHECKEDGE(SCE1,SCE,SC,IN1,IN2,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE2,SCE,SC,IN2,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE3,SCE,SC,IN1,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL EXISTENCEFACE(SCE1,SCE2,SCE3,NFAC,SCFOL,VICFAC,FACE)
   CALL CHECKFACE(SCE1,SCE2,SCE3,IN1,IN2,IND,SCF,SCFOL,FACE,PID,
    *                    1,LSTAK,SC,NFAC,VICFAC)
 SCF1=SCFOL
   CALL EXISTENCEEDGE(IN1,IN3,IND,NEDG,SCE1,SCE2,SCE3,VICEDG,EDGE)
   CALL CHECKEDGE(SCE1,SCE,SC,IN1,IN3,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE2,SCE,SC,IN3,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE3,SCE,SC,IN1,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL EXISTENCEFACE(SCE1,SCE3,SCE3,NFAC,SCFOL,VICFAC,FACE)
   CALL CHECKFACE(SCE1,SCE2,SCE3,IN1,IND,IN3,SCF,SCFOL,FACE,PID,
    *                    1,LSTAK,SC,NFAC,VICFAC)
 SCF2=SCFOL
   CALL EXISTENCEEDGE(IN2,IN3,IND,NEDG,SCE1,SCE2,SCE3,VICEDG,EDGE)
   CALL CHECKEDGE(SCE1,SCE,SC,IN2,IN3,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE2,SCE,SC,IN3,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL CHECKEDGE(SCE3,SCE,SC,IN2,IND,PID,EDGE,0,LSTAK,NEDG,VICEDG)
   CALL EXISTENCEFACE(SCE1,SCE2,SCE3,NFAC,SCFOL,VICFAC,FACE)
   CALL CHECKFACE(SCE1,SCE3,SCE3,IND,IN2,IN3,SCF,SCFOL,FACE,PID,
```

```
     *                     1,LSTAK,SC,NFAC,VICFAC)
SCF3=SCFOL
    CALL EXISTENCEEDGE(IN1,IN2,IN3,NEDG,SCE1,SCE2,SCE3,VICEDG,EDGE)
    CALL CHECKEDGE(SCE1,SCE,SC,IN1,IN2,PID,EDGE,0,LSTAK,NEDG,VICEDG)
    CALL CHECKEDGE(SCE2,SCE,SC,IN2,IN3,PID,EDGE,0,LSTAK,NEDG,VICEDG)
    CALL CHECKEDGE(SCE3,SCE,SC,IN1,IN3,PID,EDGE,0,LSTAK,NEDG,VICEDG)
         CALL EXISTENCEFACE(SCE1,SCE2,SCE3,NFAC,SCFOL,VICFAC,FACE)
    CALL CHECKFACE(SCE1,SCE2,SCE3,IN1,IN2,IN3,SCF,SCFOL,FACE,PID,
    *                          1,LSTAK,SC,NFAC,VICFAC)
SCF4=SCFOL
CALL EXISTENCEPYR(SCF1,SCF2,SCF3,SCF4,PYR,SCPOL,NPYR,VICPYR)
    IF (SCPOL.EQ.0) THEN
  CALL CHECKPYR(SCF1,SCF2,SCF3,SCF4,IN1,IN2,IN3,IND,
    *                              SCP,SCPOL,PID,PYR)
  call pyr4node(edge,face,pyr,scp,node)
        DO 81 K=1,4
          COOR(K,1)=P(NODE(K),1)
          COOR(K,2)=P(NODE(K),2)
81           COOR(K,3)=P(NODE(K),3)
     VOL=VOL+VOLUME(COOR)
   WRITE(*,1110) scp,node(1),node(2),node(3),node(4),volume(coor),vol
  IF (SCOLD.NE.SC) SC=SC-1
       ELSE
          DONTUSE=.TRUE.
          GOTO 100
END IF
    ELSE
      DONTUSE=.TRUE.
      GOTO 100
    END IF
ELSE
110   SC=SC-1
c  NO=0
END IF
END DO
C
AREA=0.0
F1=0
DO 40 I=1,SCP
   CALL OCCURENCEP(I,SCP,OCCUR,PYR,PYRCOM)
   CALL BASE3NODE(EDGE,FACE,I,IN1,IN2,IN3)
c     write(*,*) I,in1,in2,in3,occur
  IF (OCCUR.EQ.1) THEN
    CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
    CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
    CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
    AREA=AREA+SURFACE(DUMP1,DUMP2,DUMP3)
    F1=F1+1
  END IF
   IF (OCCUR.GE.2) CALL FACEIN(FACE(I,4))
40 CONTINUE
VOL=0.0
DO 1000 I=1,SCP
   CALL PYR4NODE(EDGE,FACE,PYR,I,NODE)
   DO 80 K=1,4
     COOR(K,1)=P(NODE(K),1)
     COOR(K,2)=P(NODE(K),2)
80       COOR(K,3)=P(NODE(K),3)
   VOL=VOL+VOLUME(COOR)
c      WRITE(*,1110) I,node(1),node(2),node(3),node(4),volume(coor),vol
1000 CONTINUE
   WRITE(*,*) 'AREA IS:',AREA,'FOR ',F1, 'FACES'
   WRITE(*,*)  'VOLUME IS :',VOL
1100 FORMAT(5I8,3x,f12.7)
1110 FORMAT(5I8,3x,f12.7,2x,f12.7)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETCORNER(DUMP1,DUMP2,DUMP3,IN1,IN2,IN3,IND,MID1)
REAL DUMP1(3),DUMP2(3),DUMP3(3),MID1(3)
INTEGER IN1,IN2,IN3,IND
C
IMPLICIT NONE
REAL DIR1(3),DIR2(3),ANG(3),COSINUS
C
CALL DIRECTION(DUMP2,DUMP1,DIR1)
CALL DIRECTION(DUMP3,DUMP1,DIR2)
ANG(1)=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
CALL DIRECTION(DUMP1,DUMP2,DIR1)
CALL DIRECTION(DUMP3,DUMP2,DIR2)
ANG(2)=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
CALL DIRECTION(DUMP2,DUMP3,DIR1)
CALL DIRECTION(DUMP1,DUMP3,DIR2)
ANG(3)=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
IF ( (ANG(1).LE.ANG(2)).AND.(ANG(1).LE.ANG(3)) ) IND=IN1
IF ( (ANG(2).LE.ANG(1)).AND.(ANG(2).LE.ANG(3)) ) IND=IN2
IF ( (ANG(3).LE.ANG(1)).AND.(ANG(3).LE.ANG(2)) ) IND=IN3
IF (IND.EQ.IN1) CALL MIDPOI(DUMP2,DUMP3,MID1)
IF (IND.EQ.IN2) CALL MIDPOI(DUMP1,DUMP3,MID1)
IF (IND.EQ.IN3) CALL MIDPOI(DUMP2,DUMP1,MID1)
IF (IND.EQ.IN1) CALL MIDPOI(MID1,DUMP1,MID1)
IF (IND.EQ.IN2) CALL MIDPOI(MID1,DUMP2,MID1)
IF (IND.EQ.IN3) CALL MIDPOI(MID1,DUMP3,MID1)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETNORMALTOFACE(DUMP1,DUMP2,DUMP3,DUMP4,ORT)
REAL DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3),ORT(3)
C
IMPLICIT NONE
REAL MID(3),MID1(3),DIR(3),FAC,MAG,COSINUS,VAL
C
FAC=1.0
CALL DIRECTION(DUMP2,DUMP1,MID)
CALL DIRECTION(DUMP3,DUMP1,MID1)
CALL DIRECTION(DUMP4,DUMP1,DIR)
CALL NORMALVEC(MID,ORT,MID1(1),MID1(2),MID1(3))
VAL=COSINUS(ORT,DIR(1),DIR(2),DIR(3))
IF (VAL.GT.0.0) FAC=-1.0
MAG=SQRT(ORT(1)**2+ORT(2)**2+ORT(3)**2)
ORT(1)=ORT(1)*FAC/MAG
```

```
ORT(2)=ORT(2)*FAC/MAG
ORT(3)=ORT(3)*FAC/MAG
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
SUBROUTINE FACEVICINITY(NPYR,VICPYR,PYR,NFAC,VICFAC)
INTEGER NPYR,VICPYR(400),PYR(40000,6),NFAC,VICFAC(2000)
C
IMPLICIT NONE
INTEGER I,F(4),J,IND,K
C
DO 10 I=1,NPYR
   IND=VICPYR(I)
  DO 20 J=1,4
    F(J)=PYR(IND,J)
    DO 30 K=1,NFAC
30     IF (VICFAC(K).EQ.F(J)) GOTO 20
    NFAC=NFAC+1
    VICFAC(NFAC)=F(J)
20    CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EDGEVICINITY(NFAC,VICFAC,FACE,NEDG,VICEDG)
INTEGER NFAC,VICFAC(2000),FACE(65000,4),NEDG,VICEDG(1000)
C
IMPLICIT NONE
INTEGER I,E(3),J,IND,K
C
NEDG=0
DO 10 I=1,NFAC
  DO 20 J=1,3
    IND=VICFAC(I)
    E(J)=FACE(IND,J)
    DO 30 K=1,NEDG
30     IF (VICEDG(K).EQ.E(J)) GOTO 20
    NEDG=NEDG+1
    VICEDG(NEDG)=E(J)
20    CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE UPDATESTAK(SC,LSTAK)
INTEGER SC,LSTAK(5000)
C
IMPLICIT NONE
INTEGER I,J
I=0
J=1
DO WHILE ( J.LE.(SC-1))
I=I+1
J=J+1
LSTAK(I)=LSTAK(J)
IF (LSTAK(J).EQ.0) I=I-1
END DO
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OVERLAPWITHFACE(FACE,EDGE,VICEDG,VICFAC,NEDG,NFAC,
     * SCF1,IND,P,NOGOOD1,NOGOOD2,NV,VICIN)
INTEGER FACE(65000,4),EDGE(50000,3),NEDG,SCF1,IND,NFAC,NV
INTEGER VICFAC(2000),VICEDG(1000),VICIN(50)
REAL P(30000,3)
LOGICAL NOGOOD1,NOGOOD2
C
IMPLICIT NONE
INTEGER IN11,IN22,IN33,I,J,E,NF,AF(300),IN1,IN2,IN3,IN(3),IN4,K
INTEGER NE,EDJ(1000)
REAL AP,BP,CP,DP,DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3),dz,dy
LOGICAL SAME,SAMESIDE,COPL,COPLANAR,INTERSECT,INTERSLINE
C
NOGOOD1=.FALSE.
NOGOOD2=.FALSE.
CALL ONEDIM(P(IND,1),P(IND,2),P(IND,3),DUMP4)
CALL BASE3NODE(EDGE,FACE,SCF1,IN(1),IN(2),IN(3))
CALL ONEDIM(P(IN(1),1),P(IN(1),2),P(IN(1),3),DUMP1)
CALL ONEDIM(P(IN(2),1),P(IN(2),2),P(IN(2),3),DUMP2)
CALL ONEDIM(P(IN(3),1),P(IN(3),2),P(IN(3),3),DUMP3)
CALL PLANEPAR3(DUMP1,DUMP2,DUMP3,AP,BP,CP,DP)
DO 10 I=1,3
  E=FACE(SCF1,I)
  IN11=EDGE(E,1)
  IN22=EDGE(E,2)
  CALL EDGEINCOMMON(VICFAC,NFAC,NF,AF,E,FACE)
  CALL ONEDIM(P(IN11,1),P(IN11,2),P(IN11,3),DUMP1)
  CALL ONEDIM(P(IN22,1),P(IN22,2),P(IN22,3),DUMP2)
  DO 20 J=1,NF
    CALL BASE3NODE(EDGE,FACE,AF(J),IN1,IN2,IN3)
    IF ((IN1.NE.IN11).AND.(IN1.NE.IN22)) IN33=IN1
    IF ((IN2.NE.IN11).AND.(IN2.NE.IN22)) IN33=IN2
    IF ((IN3.NE.IN11).AND.(IN3.NE.IN22)) IN33=IN3
    CALL ONEDIM(P(IN33,1),P(IN33,2),P(IN33,3),DUMP3)
    SAME=SAMESIDE(EDGE,FACE,AP,BP,CP,DP,SCF1,IND,IN33,P)
    COPL=COPLANAR(DUMP1,DUMP2,DUMP3,DUMP4)
    IF (SAME.AND.COPL.AND.(IND.NE.IN33)) NOGOOD1=.TRUE.
    IF (NOGOOD1) GOTO 70
20   CONTINUE
10 CONTINUE
C
DO 40 I=1,NV
  IN11=VICIN(I)
  CALL EDGEINCOMMONFROMPOINTS(FACE,EDGE,NEDG,NFAC,VICEDG,VICFAC,
     *    IN11,SCF1,EDJ,NE)
  DO 50 J=1,NE
    IN4=IND
    IN2=EDGE(EDJ(J),1)
```

```
      IN3=EDGE(EDJ(J),2)
      DO 60 K=1,3
        IN1=IN(K)
        CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
        CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
        CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
        CALL ONEDIM(P(IN4,1),P(IN4,2),P(IN4,3),DUMP4)
        INTERSECT=INTERSLINE(DUMP1,DUMP2,DUMP3,DUMP4)
        IF (INTERSECT) NOGOOD2=.TRUE.
        IF (NOGOOD2) THEN
          dx=dy
continue
continue
continue
continue
GOTO 70
      END IF
60      CONTINUE
50    CONTINUE
40 CONTINUE
70 CONTINUE
continue
continue
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION INTERSLINE(DUMP1,DUMP2,DUMP3,DUMP4)
REAL DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3)
C
IMPLICIT NONE.
REAL DIR1(3),DIR2(3),T,P1(3),P2(3),DIFX,DIFY,DIFZ
INTERSLINE=.FALSE.
CALL DIRECTION(DUMP2,DUMP3,DIR1)
CALL DIRECTION(DUMP4,DUMP1,DIR2)
CALL PARAMETERINT(DUMP3,DUMP1,DIR1,DIR2,T)
P1(1)=DUMP3(1)+T*DIR1(1)
P1(2)=DUMP3(2)+T*DIR1(2)
P1(3)=DUMP3(3)+T*DIR1(3)
P2(1)=DUMP1(1)+T*DIR2(1)
P2(2)=DUMP1(2)+T*DIR2(3)
P2(3)=DUMP1(3)+T*DIR2(3)
DIFX=ABS(P1(1)-P2(1))
DIFY=ABS(P1(2)-P2(2))
DIFZ=ABS(P1(3)-P2(3))
IF ((DIFX.LT.0.000001).AND.(DIFY.LT.0.000001).AND.(DIFZ.LT.0.000001)
   * .AND.(T.LT.0.99999).AND.(T.GT.0.00001)) INTERSLINE=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETVICPYR(VICPYR,NPYR,EDGE,FACE,PYR,SCP,SCF,SCE,SCF1,
   *                   NFAC,VICFAC,VICIN,NV,P)
INTEGER VICPYR(400),NPYR,EDGE(50000,3),FACE(65000,4),VICIN(50),NV
INTEGER PYR(40000,5),SCP,SCF1,SCE,SCF,NFAC,VICFAC(2000)
REAL P(30000,3)
C
IMPLICIT NONE
INTEGER I,J,K,ND(4),TIMES
REAL DUM(3),DUM1(3),DIST,FAR,MAX
C
NPYR=0
NFAC=0
FAR=0.0
CALL BASE3NODE(EDGE,FACE,SCF1,VICIN(NV+1),VICIN(NV+2),VICIN(NV+3))
CALL ONEDIM(P(VICIN(1),1),P(VICIN(1),2),P(VICIN(1),3),DUM)
DO 5 I=2,NV
  CALL ONEDIM(P(VICIN(NV),1),P(VICIN(NV),2),P(VICIN(NV),3),DUM1)
  MAX=DIST(DUM,DUM1)
  IF (MAX.GT.FAR) FAR=MAX
5 CONTINUE
FAR=3*FAR+0.005
DO 10 I=1,SCP
  DO 20 J=1,NPYR
20   IF (VICPYR(J).EQ.I) GOTO 10
  CALL PYR4NODE(EDGE,FACE,PYR,I,ND)
  CALL ONEDIM(P(ND(1),1),P(ND(1),2),P(ND(1),3),DUM1)
  MAX=DIST(DUM,DUM1)
  IF (MAX.GT.FAR) GOTO 10
  TIMES=0
  DO 30 J=1,NV+3
    K=VICIN(J)
    IF ( (K.EQ.ND(1)).OR.(K.EQ.ND(2)).OR.(K.EQ.ND(3)).OR.
   *          (K.EQ.ND(4)) ) TIMES=TIMES+1
    IF (TIMES.GE.2) THEN
      NPYR=NPYR+1
      VICPYR(NPYR)=I
      GOTO 10
    END IF
30 CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EDGEINCOMMONSPECH(FACE,SCF,NF,AF,E)
INTEGER FACE(65000,4),SCF,NF,AF(500),E(3)
C
IMPLICIT NONE
INTEGER I,E1,E2,E3,J
LOGICAL SAME(3)
NF=0
DO 10 I=1,SCF
  E1=FACE(I,1)
  E2=FACE(I,2)
  E3=FACE(I,3)
  DO 20 J=1,3
    SAME(J)=.FALSE.
20     SAME(J)=(E(J).EQ.E1).OR.(E(J).EQ.E2).OR.(E(J).EQ.E3)
  IF ( SAME(1).OR.SAME(2).OR.SAME(3) ) THEN
  DO 30 J=1,NF
30  IF (AF(J).EQ.I) GOTO 10
NF=NF+1
```

```
AF(NF)=I
  END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
SUBROUTINE EDGEINCOMMONFROMPOINTS(FACE,EDGE,NEDG,NFAC,VICEDG,VICFAC,
     *IND,SCF1,EDJ,NE)
INTEGER FACE(65000,4),EDGE(50000,3),IND,SCF1,NE,EDJ(1000)
INTEGER NEDG,NFAC,VICEDG(1000),VICFAC(2000)
C
IMPLICIT NONE
INTEGER E1,E2,E3,P1,P2,I,J,K,NF1,AF1(300),NEOLD,L,E
C
E1=FACE(SCF1,1)
E2=FACE(SCF1,2)
E3=FACE(SCF1,3)
NE=0
DO 10 I=1,NEDG
  P1=EDGE(VICEDG(I),1)
  P2=EDGE(VICEDG(I),2)
  IF ((P1.EQ.IND).OR.(P2.EQ.IND)) THEN
    IF ((I.NE.E1).AND.(I.NE.E2).AND.(I.NE.E3)) THEN
NE=NE+1
EDJ(NE)=VICEDG(I)
    END IF
  END IF
10 CONTINUE
C
NEOLD=NE
DO 20 L=1,NEOLD
  CALL EDGEINCOMMON(VICFAC,NFAC,NF1,AF1,EDJ(L),FACE)
C
DO 50 I=1,NF1
  DO 60 J=1,3
    E=FACE(AF1(I),J)
    IF ((I.EQ.E1).OR.(I.EQ.E2).OR.(I.EQ.E3)) GOTO 60
    DO 70 K=1,NE
      IF (E.EQ.EDJ(K)) GOTO 60
70    CONTINUE
    NE=NE+1
    EDJ(NE)=E
60  CONTINUE
50 CONTINUE
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EDGEINCOMMONFROMPOINTSSPECH(FACE,EDGE,SCE,SCF,SCF1,EDJ,NE)
INTEGER FACE(65000,4),EDGE(50000,3),SCF1,NE,EDJ(1000),SCE,SCF
C
IMPLICIT NONE
INTEGER P1,P2,I,K,NF1,AF1(300),NEOLD,L,E,EDJ1(1000),NE1
INTEGER E1,E2,E3
LOGICAL COTE1,COTE2,COTE3
C
NE1=0
NE=0
CALL BASE3NODE(EDGE,FACE,SCF1,E1,E2,E3)
DO 10 I=1,SCE
  P1=EDGE(I,1)
  P2=EDGE(I,2)
  COTE1=(E1.EQ.P1).OR.(E1.EQ.P2)
  COTE2=(E2.EQ.P1).OR.(E2.EQ.P2)
  COTE3=(E3.EQ.P1).OR.(E3.EQ.P2)
  IF (COTE1.OR.COTE2.OR.COTE3) THEN
      DO 40 L=1,NE1
40      IF (EDJ1(L).EQ.I) GOTO 10
NE1=NE1+1
EDJ1(NE1)=I
    END IF
10 CONTINUE
C
NEOLD=NE1
DO 50 L=1,NEOLD
  CALL EDGEINCOMMON2D(FACE,SCF,NF1,AF1,EDJ1(L))
  DO 60 I=1,NF1
    E=AF1(I)
    DO 70 K=1,NE
      IF (E.EQ.EDJ(K)) GOTO 60
70    CONTINUE
    NE=NE+1
    EDJ(NE)=E
60 CONTINUE
50 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
LOGICAL FUNCTION SAMESIDE(EDGE,FACE,AP,BP,CP,DP,SCF1,IND1,IND2,P)
INTEGER EDGE(50000,3),FACE(65000,4),IND1,SCF1,IND2
REAL P(30000,3),AP,BP,CP,DP
C
IMPLICIT NONE
REAL VAL1,VAL2
LOGICAL OVER,UNDER
C
SAMESIDE=.FALSE.
VAL1=AP*P(IND1,1)+BP*P(IND1,2)+CP*P(IND1,3)+DP
VAL2=AP*P(IND2,1)+BP*P(IND2,2)+CP*P(IND2,3)+DP
OVER=(VAL1.GT.0).AND.(VAL2.GT.0)
UNDER=(VAL1.LT.0).AND.(VAL2.LT.0)
IF (OVER.OR.UNDER) SAMESIDE=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION SURFACE(DUM1,DUM2,DUM3)
REAL DUM1(3),DUM2(3),DUM3(3)
```

```
C
IMPLICIT NONE
REAL VEC1(3),VEC2(3),NVEC(3)
C
CALL DIRECTION(DUM2,DUM1,VEC1)
CALL DIRECTION(DUM3,DUM1,VEC2)
CALL NORMALVEC(VEC1,NVEC,VEC2(1),VEC2(2),VEC2(3))
SURFACE=0.5*SQRT( NVEC(1)**2+NVEC(2)**2+NVEC(3)**2 )
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE FACEIN(F1)
INTEGER F1
C
IMPLICIT NONE
INTEGER VAL
C IF (F1.GE.500000) THEN
C   F1=3331
C ELSE
  VAL=F1/10*10
  F1=F1-VAL
C END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE INSIDEPYR(IN1,IN2,IN3,IN4,DM1,DM2,DM3,DM4,P,NV,VICIN,NO)
INTEGER IN1,IN2,IN3,IN4,VICIN(50),NV,NO
REAL P(30000,3),DM1(3),DM2(3),DM3(3),DM4(3)
C
IMPLICIT NONE
INTEGER L,I,II
REAL AP(4),BP(4),CP(4),DP(4),POINT(1,3)
LOGICAL VAL(4),ABOVE
C
NO=0
CALL PARPLANES(DM1,DM2,DM3,DM4,AP,BP,CP,DP)
DO 10 II=1,NV
  I=VICIN(II)
  IF ((IN1.EQ.I).OR.(IN2.EQ.I).OR.(IN3.EQ.I).OR.(IN4.EQ.I)) GOTO 10
  POINT(1,1)=P(I,1)
  POINT(1,2)=P(I,2)
  POINT(1,3)=P(I,3)
DO 40 L=1,4
40  VAL(L)=ABOVE(POINT,AP(L),BP(L),CP(L),DP(L))
IF (VAL(1).AND.VAL(2).AND.VAL(3).AND.VAL(4)) NO=NO+1
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION ABOVE(COOR,AP,BP,CP,DP)
REAL COOR(1,3),AP,BP,CP,DP
C
IMPLICIT NONE
REAL VAL
ABOVE=.FALSE.
VAL=AP*COOR(1,1)+BP*COOR(1,2)+CP*COOR(1,3)+DP
IF (VAL.GE.0.0) ABOVE=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION REALLYABOVE(COOR,AP,BP,CP,DP)
REAL COOR(1,3),AP,BP,CP,DP
C
IMPLICIT NONE
REAL VAL
REALLYABOVE=.FALSE.
VAL=AP*COOR(1,1)+BP*COOR(1,2)+CP*COOR(1,3)+DP
IF (VAL.GT.0.0) REALLYABOVE=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE PARPLANES(DUM1,DUM2,DUM3,DUM4,AP,BP,CP,DP)
REAL DUM1(3),DUM2(3),DUM3(3),DUM4(3),AP(4),BP(4),CP(4),DP(4)
C
IMPLICIT NONE
REAL DUM5(3),DUM6(3),DUM7(3),DUM8(3),ANG(4),COSINUS
INTEGER J
C
CALL PLANEPAR3(DUM1,DUM2,DUM3,AP(1),BP(1),CP(1),DP(1))
CALL PLANEPAR3(DUM1,DUM2,DUM4,AP(2),BP(2),CP(2),DP(2))
CALL PLANEPAR3(DUM1,DUM3,DUM4,AP(3),BP(3),CP(3),DP(3))
CALL PLANEPAR3(DUM2,DUM3,DUM4,AP(4),BP(4),CP(4),DP(4))
CALL DUMP4PROJ(DUM4,AP(1),BP(1),CP(1),DP(1),DUM5)
CALL DUMP4PROJ(DUM3,AP(2),BP(2),CP(2),DP(2),DUM6)
CALL DUMP4PROJ(DUM2,AP(3),BP(3),CP(3),DP(3),DUM7)
CALL DUMP4PROJ(DUM1,AP(4),BP(4),CP(4),DP(4),DUM8)
CALL DIRECTION(DUM4,DUM5,DUM5)
CALL DIRECTION(DUM3,DUM6,DUM6)
CALL DIRECTION(DUM2,DUM7,DUM7)
CALL DIRECTION(DUM1,DUM8,DUM8)
ANG(1)=COSINUS(DUM5,AP(1),BP(1),CP(1))
ANG(2)=COSINUS(DUM6,AP(2),BP(2),CP(2))
ANG(3)=COSINUS(DUM7,AP(3),BP(3),CP(3))
ANG(4)=COSINUS(DUM8,AP(4),BP(4),CP(4))
DO  20 J=1,4
  IF (ANG(J).LT.0.0) THEN
     AP(J)=-AP(J)
BP(J)=-BP(J)
CP(J)=-CP(J)
DP(J)=-DP(J)
    END IF
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION  COSINUS(VEC1,X,Y,Z)
REAL VEC1(3),X,Y,Z
```

```
C
IMPLICIT NONE
REAL NUM,DEN
DEN=SQRT(X**2+Y**2+Z**2)*SQRT(VEC1(1)**2+VEC1(2)**2+VEC1(3)**2)
NUM=X*VEC1(1)+Y*VEC1(2)+Z*VEC1(3)
COSINUS=NUM/DEN
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE FACECREATIONSURFACE(NP,NPS,SCF,SCE,P,PID,EDGE,FACE)
INTEGER NP,SCF,SCE,EDGE(50000,3),FACE(65000,4),PID(30000),NPS
REAL P(30000,3)
C
IMPLICIT NONE
REAL MIN,DUMP1(3),DUMP2(3),MID(3),CENT(3),MINO,DUMP3(3),ANG
REAL DIR1(3),ANGREF,DIR2(3)
INTEGER LSTAK(5000),SC,SCE1,SCE2,SCE3,IND,IN1,IN2,OCCUR,I,J
INTEGER EDGEID,SCE1I,SCE2I,SCE3I,IN3,IN11,IN22,IN33,NF,AF(300)
REAL DIST,CENT1(3),DUMP4(3),MAT(3,3),VAL,ORG(3),COSINUS
INTEGER VICIN(50),NV,II,NO,VICIOL(20),SCOLD,SCFOL,ecfold
LOGICAL CENTERVALIDS,VALID,COPL,FACELAP,COPLANAR,FACEOVERLAP
C
C WHEN YOU RUN A PURELY 2D PROBLEM SHIFT ALL BY ONE IN THE Z-DIRECTION
C FOR 3D ORG(3) IS AT THE HEIGTH OF THE COVER OF THE BOX
ORG(1)=0.0
ORG(2)=0.0
ORG(3)=1.0
IND=2
SC=0
ecf=0
ecfold=0
CALL ONEDIM(P(1,1),P(1,2),P(1,3),DUMP1)
CALL ONEDIM(P(IND,1),P(IND,2),P(IND,3),DUMP2)
CALL DIRECTION(ORG,DUMP1,DIR1)
CALL DIRECTION(DUMP2,DUMP1,DIR2)
ANGREF=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
MIN=DIST(DUMP1,DUMP2)
DO 10 I=3,NPS
  CALL ONEDIM(P(I,1),P(I,2),P(I,3),DUMP2)
  CALL DIRECTION(DUMP2,DUMP1,DIR2)
  ANG=COSINUS(DIR1,DIR2(1),DIR2(2),DIR2(3))
  IF (ANG.GT.ANGREF)  THEN
    IF (DIST(DUMP1,DUMP2).LT.(1.5*MIN)) THEN
MIN=DIST(DUMP1,DUMP2)
IND=I
ANGREF=ANG
END IF
  END IF
10 CONTINUE

SC=SC+1
SCE=SCE+1
EDGE(SCE,1)=1
EDGE(SCE,2)=IND
EDGE(SCE,3)=EDGEID(PID(1),PID(IND))
LSTAK(SC)=SCE
C
DO WHILE (SC.GT.0)
  SCE1=LSTAK(SC)
  CALL OCCURENCEF(SCE1,SCF,OCCUR,FACE)
  CALL EDGEINCOMMON2D(FACE,SCF,NF,AF,SCE1)
60   IF  (OCCUR.LT.2) THEN
    MINO=10000.0
    IN1=EDGE(SCE1,1)
    IN2=EDGE(SCE1,2)
    CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
    CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
    CALL MAKEMAT3(DUMP1,DUMP2,ORG,MAT)
    CALL DETERMINANT(MAT,3,3,VAL)
    IF (VAL.LT.0) THEN
    IN1=EDGE(SCE1,2)
    IN2=EDGE(SCE1,1)
    END IF
    IF (NF.NE.0) CALL BASE3NODE(EDGE,FACE,AF(1),IN11,IN22,IN33)
    IF ((IN11.NE.IN1).AND.(IN11.NE.IN2)) IN3=IN11
    IF ((IN22.NE.IN1).AND.(IN22.NE.IN2)) IN3=IN22
    IF ((IN33.NE.IN1).AND.(IN33.NE.IN2)) IN3=IN33
    CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP4)
    CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
    CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
    CALL MIDPOI(DUMP1,DUMP2,MID)
    CALL VICINITY2D(VICIN,NV,MID,NP,NPS,P,20)
    IND=0
    DO 20 II=1,NV
      I=VICIN(II)
      IF ((IN1.EQ.I).OR.(IN2.EQ.I)) GOTO 20
      DO 40 J=1,NO
IF (I.EQ.VICIOL(J)) GOTO 20
40       CONTINUE
      CALL ONEDIM(P(I,1),P(I,2),P(I,3),DUMP3)
      FACELAP=FACEOVERLAP(DUMP1,DUMP2,DUMP3,DUMP4,DUMP3)
      COPL=COPLANAR(DUMP1,DUMP2,DUMP3,DUMP4)
      IF (FACELAP.AND.(NF.NE.0)) GOTO 20
      IF (COPL.AND.FACELAP.AND.(NF.NE.0)) GOTO 20
      CALL CENTERSPACE(DUMP1,DUMP2,DUMP3,CENT,1)
      VALID=CENTERVALIDS(CENT)
c       VALID=.TRUE.
      IF ( VALID ) THEN
        MIN=DIST(CENT,MID)
        IF (MIN.LT.MINO) THEN
    IND=I
 MINO=MIN
 SCE1I=SCE1
 SCE2I=SCE2
 SCE3I=SCE3
 CALL TRANSFER(CENT,CENT1)
       END IF
      END IF
20     CONTINUE
C
    SCOLD=SC
    IF (IND.NE.0) THEN
```

```
 NO=0
        CALL ELIMINATORS(VICIN,NV,CENT1,DUMP1,P,NO,VICIOL)
    IF (NO.NE.0) THEN
      CALL SPECIALORDER(CENT1,IND,IN1,IN2,NO,VICIOL,P)
      IN1=VICIOL(1)
       DO 30 I=2,(NO-1)
 IN2=VICIOL(I)
 IN3=VICIOL(I+1)
       CALL EXISTENCEEDGE2D(IN1,IN2,IN3,SCE,SCE1,SCE2,SCE3,EDGE)
        CALL EXISTENCEFACE2D(SCE1,SCE2,SCE3,SCF,SCFOL,FACE)
        CALL CHECKEDGE2D(SCE1,SCE,SC,IN1,IN2,PID,EDGE,1,LSTAK)
 CALL CHECKEDGE2D(SCE2,SCE,SC,IN2,IN3,PID,EDGE,1,LSTAK)
 CALL CHECKEDGE2D(SCE3,SCE,SC,IN1,IN3,PID,EDGE,1,LSTAK)
    CALL CHECKFACE2D(SCE1,SCE2,SCE3,IN1,IN2,IN3,SCF,SCFOL,FACE,PID,
    *                                             0,LSTAK,SC)
30      CONTINUE
    ELSE
        SCE1=SCE1I
        SCE2=SCE2I
        SCE3=SCE3I
        CALL CHECKEDGE2D(SCE1,SCE,SC,IN1,IN2,PID,EDGE,1,LSTAK)
        CALL CHECKEDGE2D(SCE2,SCE,SC,IN2,IND,PID,EDGE,1,LSTAK)
        CALL CHECKEDGE2D(SCE3,SCE,SC,IND,IN1,PID,EDGE,1,LSTAK)
    CALL CHECKFACE2D(SCE1,SCE2,SCE3,IN1,IN2,IND,SCF,SCFOL,FACE,PID,
    *                                            0,LSTAK,SC)
   END IF
      SC=SC-1
   ELSE
c NO=0
      SC=SC-1
   END IF
  if (scfold.ne.scf) then
    call base3node(edge,face,scf,in1,in2,in3)
c       write(*,*) scf,in1,in2,in3
    scfold=scf
  end if
 ELSE
   SC=SC-1
 END IF
END DO
DO 1000 I=1,SCF
    call base3node(edge,face,I,in1,in2,in3)
1000     WRITE(*,1100) I,in1,in2,in3
1100 FORMAT(7I8)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 SUBROUTINE VICINITY2D(VICIN,NV,POI,NP,NPD,P,NUMB)
INTEGER VICIN(50),NV,NP,NPD,NUMB
REAL POI(3),P(30000,3)
C
IMPLICIT NONE
REAL DUM(3),DIST,LON(30000),FAR
INTEGER I,J,STAR,FIN,IND,K,VOISIN(30000)
C
NV=0
STAR=NP+1
FIN=NPD+NP
CALL ONEDIM(P(STAR,1),P(STAR,2),P(STAR,3),DUM)
LON(1)=DIST(DUM,POI)
VOISIN(1)=STAR
DO 10 I=1,(FIN-STAR)
IND=STAR+I
CALL ONEDIM(P(IND,1),P(IND,2),P(IND,3),DUM)
FAR=DIST(DUM,POI)
c IF (FAR.GT.CRIT) GOTO 10
IF (FAR.LT.LON(1)) THEN
  DO 20 J=1,1,-1
   VOISIN(J+1)=VOISIN(J)
20     LON(J+1)=LON(J)
  LON(1)=FAR
  VOISIN(1)=IND
ELSE
  DO 30 J=1,I
   IF (FAR.LT.LON(J)) THEN
     DO 40 K=I,J,-1
 VOISIN(K+1)=VOISIN(K)
40       LON(K+1)=LON(K)
     LON(J)=FAR
     VOISIN(J)=IND
     GOTO 10
   END IF
30   CONTINUE
  LON(I+1)=FAR
  VOISIN(I+1)=IND
END IF
10 CONTINUE
NV=NUMB
DO 50 I=1,NV
50 VICIN(I)=VOISIN(I)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


C
INTEGER FUNCTION EDGEID(P1,P2)
INTEGER P1,P2
C
IMPLICIT NONE
INTEGER D1(5),D2(5),DP1,DP2,VAL,I,J,EDG
DP1=P1
DP2=P2
IF (DP1.GE.500000) DP1=DP1-500000
IF (DP2.GE.500000) DP2=DP2-500000
  DO 10 I=1,5
  VAL=INT(DP1/10)*10
  D1(I)=DP1-VAL
10 DP1=VAL/10
  DO 20 I=1,5
  VAL=INT(DP2/10)*10
  D2(I)=DP2-VAL
20 DP2=VAL/10
```

```
EDG=0
DO 30 I=1,8
 J=I-1
 IF (D1(I).EQ.4) D1(I)=D2(I)
 IF (D2(I).EQ.4) D2(I)=D1(I)
 IF (D1(I).EQ.D2(I)) EDG=EDG+D1(I)*(10**J)
30 CONTINUE
EDGEID=EDG
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
INTEGER FUNCTION PYRID(P1,P2,P3,P4)
INTEGER P1,P2,P3,P4
C
IMPLICIT NONE
INTEGER D1,D2,D3,D4,DP1,DP2,DP3,DP4
DP1=P1
DP2=P2
DP3=P3
DP4=P4
IF (DP1.GE.500000) DP1=DP1-500000
IF (DP2.GE.500000) DP2=DP2-500000
IF (DP3.GE.500000) DP3=DP3-500000
IF (DP4.GE.500000) DP4=DP4-500000
D1=DP1/10**10
D2=DP2/10**10
D3=DP3/10**10
D4=DP4/10**10
D1=DP1-D1
D2=DP2-D2
D3=DP3-D3
D4=DP4-D4
PYRID=0
IF ((D1.EQ.D2).AND.(D3.EQ.D4).AND.(D2.EQ.D3)) THEN
PYRID=D1
ELSE
 IF ((D1.NE.D2).AND.(D1.NE.D3).AND.(D1.NE.D4).AND.(D1.NE.4)) PYRID=D1
 IF ((D2.NE.D1).AND.(D2.NE.D3).AND.(D2.NE.D4).AND.(D2.NE.4)) PYRID=D2
 IF ((D3.NE.D2).AND.(D3.NE.D1).AND.(D3.NE.D4).AND.(D3.NE.4)) PYRID=D3
 IF ((D4.NE.D2).AND.(D4.NE.D3).AND.(D4.NE.D1).AND.(D4.NE.4)) PYRID=D4
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE FILLSTAK(LSTAK,SC,SCF)
INTEGER LSTAK(8000),SC,SCF
C
IMPLICIT NONE
INTEGER I
SC=SCF
 DO 10 I=1,SCF
  LSTAK(I)=I
10  CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 SUBROUTINE VICINITY(VICIN,NV,IND,NUMB,ORDPOINT)
INTEGER VICIN(60),NV,IND,NUMB,ORDPOINT(30000,30)
C
IMPLICIT NONE
INTEGER I
C
DO 10 I=1,NUMB
10 VICIN(I)=ORDPOINT(IND,I)
NV=NUMB
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 SUBROUTINE ORDER(ORDPOINT,NP,NPD,P,MAX)
INTEGER ORDPOINT(30000,30),NP,NPD,MAX
REAL P(30000,3)
C
IMPLICIT NONE
REAL LON(30000),POI1(3),MIN,FAR
INTEGER I,J,STAR,FIN,IND,K,L,DIF,INDL
C
STAR=NP
FIN=NPD+NP
DIF=FIN-STAR
IF (DIF.LT.MAX) MAX=DIF
DO 8 L=(STAR+1),FIN
 INDL=L
 CALL ONEDIM(P(INDL,1),P(INDL,2),P(INDL,3),POI1)
C
 DO 10 I=1,DIF
  IND=I+STAR
  LON(I)=(P(IND,1)-POI1(1))**2+(P(IND,2)-POI1(2))**2+
   *          (P(IND,3)-POI1(3))**2
10  CONTINUE
K=0
MIN=99999.0
DO  20 I=1,DIF
 IF ((I+STAR).EQ.INDL) GOTO 20
 IF (LON(I).LT.MIN) then
MIN=LON(I)
end if
20 CONTINUE
IF (MIN.EQ.0.0) WRITE(*,*)
   *    'YOUR MESH HAS 2 IDENTICAL POINTS NUMBERED DIFENRENTLY'
FAR=2*MIN+0.00001
DO WHILE (K.LT.MAX)
  FAR=FAR+MIN
  DO 50 I=1,DIF
   IF (LON(I).GT.FAR) GOTO 50
   IND=I+STAR
   IF (IND.EQ.INDL) GOTO 50
   DO 30 J=1,K
30     IF (ORDPOINT(L,J).EQ.IND) GOTO 50
   K=K+1
```

```
      ORDPOINT(L,K)=IND
      IF (K.EQ.MAX) GOTO 8
60    CONTINUE
END DO
8 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE SCNDORDER(SCE,SCP,EDGE,FACE,PYR,EXTRAPYR)
INTEGER SCE,FACE(60000,4),SCP,PYR(40000,5),EXTRAPYR(40000,5)
INTEGER EDGE(60000,3)
C
IMPLICIT NONE
INTEGER SIDE(5),I,K,L,J,THAT,IND(5,2),TEMP,X
LOGICAL THERE
C
DO 20 I=1,SCP
  DO 30 J=1,5
30   SIDE(J)=0
  SIDE(1)=FACE(PYR(I,1),1)
  SIDE(2)=FACE(PYR(I,1),2)
  SIDE(3)=FACE(PYR(I,1),3)
  K=3
  DO 40 L=2,3
    DO 50 J=1,3
      THAT=FACE(PYR(I,L),J)
      IF (THERE(SIDE,THAT,K,5)) THEN
K=K+1
SIDE(K)=THAT
      END IF
50    CONTINUE
40   CONTINUE
DO 70 J=1,5
  IND(J,1)=EDGE(SIDE(J),1)
70   IND(J,2)=EDGE(SIDE(J),2)
DO 80 J=2,5
  IF (((IND(1,1).NE.IND(J,1)).AND.(IND(1,1).NE.IND(J,2))) THEN
    DO 90 L=J+1,5
    IF ((IND(L,1).EQ.IND(1,1)).OR.(IND(L,2).EQ.IND(1,1))) THEN
TEMP=SIDE(J)
SIDE(J)=SIDE(J+1)
SIDE(J+1)=TEMP
DO 95 K=1,5
  IND(K,1)=EDGE(SIDE(K),1)
95   IND(K,2)=EDGE(SIDE(K),2)
GOTO 85
  END IF
90      CONTINUE
85   END IF
80 CONTINUE
IF (IND(2,1).NE.IND(1,1)) THEN
TEMP=IND(2,2)
IND(2,2)=IND(2,1)
IND(2,1)=TEMP
END IF
IF (IND(3,1).NE.IND(1,1)) THEN
TEMP=IND(3,2)
IND(3,2)=IND(3,1)
IND(3,1)=TEMP
END IF
X=1
DO 115 L=4,5
DO 100 J=L,5
  IF (L.EQ.5) X=3
  IF ((IND(X,2).NE.IND(J,1)).AND.(IND(X,2).NE.IND(J,2))) THEN
  IF ((IND(2,2).NE.IND(J,1)).AND.(IND(2,2).NE.IND(J,2))) THEN
TEMP=SIDE(J)
SIDE(J)=SIDE(J+1)
SIDE(J+1)=TEMP
DO 120 K=1,5
  IND(K,1)=EDGE(SIDE(K),1)
120   IND(K,2)=EDGE(SIDE(K),2)
GOTO 115
    END IF
  END IF
100 CONTINUE
115 CONTINUE
C
  DO 60 K=1,5
60   EXTRAPYR(I,K)=SIDE(K)
20 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION THERE(SIDE,THAT,K,N)
INTEGER SIDE(N),THAT,K,N
C
IMPLICIT NONE
INTEGER I
THERE=.TRUE.
DO 10 I=1,K
10  IF (SIDE(I).EQ.THAT) THERE=.FALSE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION DISTLINPOI(CENT,DUM1,DX,DY,DZ)
REAL CENT(3),DX,DY,DZ,DUM1(3)
C
IMPLICIT NONE
REAL E1,E2,E3,E
C
E=DX**2+DY**2+DZ**2
E1=(CENT(1)-DUM1(1))*DY-(CENT(2)-DUM1(2))*DX
E2=(CENT(2)-DUM1(2))*DZ-(CENT(3)-DUM1(3))*DY
E3=(CENT(3)-DUM1(3))*DX-(CENT(1)-DUM1(3))*DZ
DISTLINPOI=SQRT((E1**2+E2**2+E3**2)/E)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C
SUBROUTINE ELIMINATORS(VICIN,NV,CENT,DUMP1,P,NO,VICIOL)
INTEGER VICIOL(20),VICIN(80),NV,NO
REAL CENT(3),DUMP1(3),P(30000,3)
C
IMPLICIT NONE
INTEGER I
REAL R,RD,DIST,DUM(3)
C
NO=0
R=DIST(DUMP1,CENT)
DO 10 I=1,NV
  CALL ONEDIM(P(VICIN(I),1),P(VICIN(I),2),P(VICIN(I),3),DUM)
  RD=DIST(DUM,CENT)
  IF (ABS(RD-R).LT.0.0001) THEN
NO=NO+1
    VICIOL(NO)=VICIN(I)
   END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C THIS SUBROUTINE DEPENDS ON THE PROBLEM
LOGICAL FUNCTION CENTERVALIDS(CENT)
REAL CENT(3)
C
IMPLICIT NONE
REAL RLX,RLY,RLZ
INTEGER DUM(3),LX,LY,LZ
LOGICAL GX,GY,GZ,PX,PY,PZ,PZP,GZP,BZ,BZP,BAS
C
RLX=0.5
RLY=0.5
RLZ=1.0
LX=INT(RLX*10000)
LY=INT(RLY*10000)
LZ=INT(RLZ*10000)
DUM(1)=ABS(INT(CENT(1)*10000))
DUM(2)=ABS(INT(CENT(2)*10000))
DUM(3)=ABS(INT(CENT(3)*10000))
CENTERVALIDS=.TRUE.
GX=DUM(1).GT.LX
GY=DUM(2).GT.LY
GZ=DUM(3).GT.LZ
PX=DUM(1).LT.LX
PY=DUM(2).LT.LY
PZ=DUM(3).LT.0
PZP=DUM(3).GT.0
c PZP=.TRUE.      SWITCH 4 LINES FOR EMI OR ANTE PROB
c PZ=.TRUE.       WITH previous 2 lines
c BAS=((INT(CENT(3)*10000)).EQ.(-LZ))
GZP=DUM(3).LT.LZ
BZP=PZP.AND.GZP
BZ=PZ.OR.GZ
IF ( (PX).AND.(PY).AND.(BZP) ) CENTERVALIDS=.FALSE.
IF ( (GX).AND.(GY).AND.(BZ) )  CENTERVALIDS=.FALSE.
c IF ( (PX).AND.(PY).AND.(BAS) ) CENTERVALIDS=.FALSE.
c IF ( (GX).AND.(GY).AND.(BAS) )  CENTERVALIDS=.FALSE.
END
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
C THIS SUBROUTINE DEPENDS ON THE PROBLEM
LOGICAL FUNCTION CENTERVALIDV(EDGE,FACE,SCF1,DUMP,P,RLX,RLY,RLZ)
REAL P(30000,3),RLX,RLY,RLZ,DUMP(3)
INTEGER EDGE(60000,3),FACE(65000,4),SCF1
C
IMPLICIT NONE
REAL DUM1(3),DUM2(3),DUMP1(3),DUMP2(3),DUMP3(3)
REAL CENT(3),DIR1(3),RLZ1
INTEGER IN1,IN2,IN3
LOGICAL LX1,LX2,LY1,LY2,LZ1,LZ2,L1,L2
C
RLZ1=-RLZ
CALL BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
CALL MIDPOI(DUMP1,DUMP2,CENT)
CALL MIDPOI(CENT,DUMP3,CENT)
CALL DIRECTION(CENT,DUMP,DIR1)
DUM1(1)=ABS(DUMP(1)+0.002*DIR1(1))
DUM1(2)=ABS(DUMP(2)+0.002*DIR1(2))
DUM1(3)=DUMP(3)+0.002*DIR1(3)
DUM2(1)=ABS(DUMP(1)+0.998*DIR1(1))
DUM2(2)=ABS(DUMP(2)+0.998*DIR1(2))
DUM2(3)=DUMP(3)+0.998*DIR1(3)
CENTERVALIDV=.FALSE.
LX1=(DUM1(1).LE.RLX)
LX2=(DUM2(1).LE.RLX)
LY1=(DUM1(2).LE.RLY)
LY2=(DUM2(2).LE.RLY)
c LZ1=(DUM1(3).LE.RLZ).AND.(DUM1(3).GT.0.0)  SWITCH THIS 2 LINE WITH
c LZ2=(DUM2(3).LE.RLZ).AND.(DUM2(3).GT.0.0)  NEXT 2 FOR EMI OR ANT PROB
LZ1=(DUM1(3).LE.RLZ).AND.(DUM1(3).GE.RLZ1)
LZ2=(DUM2(3).LE.RLZ).AND.(DUM2(3).GE.RLZ1)
L1=LX1.AND.LY1.AND.LZ1
L2=LX2.AND.LY2.AND.LZ2
IF ( L1.OR.L2) CENTERVALIDV=.TRUE.
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Cf
LOGICAL FUNCTION CENTERVALIDV1(EDGE,FACE,SCF1,DUMP,P,RLX,RLY,RLZ)
REAL P(30000,3),RLX,RLY,RLZ,DUMP(3)
INTEGER EDGE(60000,3),FACE(65000,4),SCF1
C
IMPLICIT NONE
REAL DUM1(3),DUM2(3),DUMP1(3),DUMP2(3),DUMP3(3)
REAL CENT(3),DIR1(3)
```

```
INTEGER IN1,IN2,IN3
LOGICAL LX1,LX2,LY1,LY2,LZ1,LZ2,L1,L2
C
CALL BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
CALL MIDPOI(DUMP1,DUMP2,CENT)
CALL MIDPOI(CENT,DUMP3,CENT)
CALL DIRECTION(CENT,DUMP,DIR1)
DUM1(1)=ABS(DUMP(1)+0.002*DIR1(1))
DUM1(2)=ABS(DUMP(2)+0.002*DIR1(2))
DUM1(3)=DUMP(3)+0.002*DIR1(3)
DUM2(1)=ABS(DUMP(1)+0.998*DIR1(1))
DUM2(2)=ABS(DUMP(2)+0.998*DIR1(2))
DUM2(3)=DUMP(3)+0.998*DIR1(3)
CENTERVALIDV1=.FALSE.
LX1=(DUM1(1).GT.RLX)
LX2=(DUM2(1).GT.RLX)
LY1=(DUM1(2).GT.RLY)
LY2=(DUM2(2).GT.RLY)
LZ1=(DUM1(3).LT.RLZ)
LZ2=(DUM2(3).LT.RLZ)
L1=(LX1.OR.LY1).AND.LZ1
L2=(LX2.OR.LY2).AND.LZ2
IF ( L1.OR.L2) CENTERVALIDV1=.TRUE.
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE DIRECTION(DUM1,DUM2,DIRE)
REAL DUM1(3),DUM2(3),DIRE(3)
C
c points from dum2 to dum1
IMPLICIT NONE
INTEGER I
DO 10 I=1,3
10 DIRE(I)=DUM1(I)-DUM2(I)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE NORMALVEC(VEC,NVEC,AD,BD,CD)
REAL VEC(3),NVEC(3),AD,BD,CD
C
IMPLICIT NONE
C
NVEC(1)=VEC(2)*CD-VEC(3)*BD
NVEC(2)=VEC(3)*AD-VEC(1)*CD
NVEC(3)=VEC(1)*BD-AD*VEC(2)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CENTERSPACE(DUM1,DUM2,DUM3,DCEN,CHECK)
REAL DUM1(3),DUM2(3),DUM3(3),DCEN(3)
INTEGER CHECK
C
IMPLICIT NONE
REAL MID1(3),MID2(3),DIR1(3),DIR2(3),NDIR1(3)
REAL NDIR2(3),AP,BP,CP,DP,T,MID12(3),MID22(3)
LOGICAL COLINEAR,COLI
C
COLI=.FALSE.
IF (CHECK.EQ.1) COLI=COLINEAR(DUM1,DUM2,DUM3)
IF (.NOT.(COLI)) THEN
  CALL PLANEPAR3(DUM1,DUM2,DUM3,AP,BP,CP,DP)
  CALL MIDPOI(DUM1,DUM2,MID1)
  CALL MIDPOI(DUM1,DUM3,MID2)
  CALL DIRECTION(DUM1,DUM3,DIR1)
  CALL DIRECTION(DUM1,DUM3,DIR2)
  CALL NORMALVEC(DIR1,NDIR1,AP,BP,CP)
  CALL NORMALVEC(DIR2,NDIR2,AP,BP,CP)
  MID12(1)=MID1(1)+NDIR1(1)
  MID12(2)=MID1(2)+NDIR1(2)
  MID12(3)=MID1(3)+NDIR1(3)
  MID22(1)=MID2(1)+NDIR2(1)
  MID22(2)=MID2(2)+NDIR2(2)
  MID22(3)=MID2(3)+NDIR2(3)
  CALL DIRECTION(MID1,MID12,DIR1)
  CALL DIRECTION(MID2,MID22,DIR2)
  CALL PARAMETERINT(MID1,MID2,DIR1,DIR2,T)
  DCEN(1)=MID2(1)+T*DIR2(1)
  DCEN(2)=MID2(2)+T*DIR2(2)
  DCEN(3)=MID2(3)+T*DIR2(3)
ELSE
  DCEN(1)=9999.9
  DCEN(2)=9999.9
  DCEN(3)=9999.9
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE PARAMETERINT(MID1,MID2,DIR1,DIR2,T)
REAL MID1(3),MID2(3),DIR1(3),DIR2(3),T
C
IMPLICIT NONE
REAL C1,C2,C3,A1,B1,A2,B2,A3,B3,D1,D2,D3
  C1=MID2(1)-MID1(1)
  C2=MID2(2)-MID1(2)
  C3=MID2(3)-MID1(3)
  A1=DIR1(1)
  B1=-DIR2(1)
  A2=DIR1(2)
  B2=-DIR2(2)
  A3=DIR1(3)
  B3=-DIR2(3)
  D1=B1*A2-B2*A1
  D2=B1*A3-B3*A1
  D3=B2*A3-B3*A2
  IF (D1.NE.0.0) THEN
  T=( A2*C1-A1*C2 ) / D1
   ELSE IF (D2.NE.0.0) THEN
```

```
 T=( A3*C1-A1*C3 ) / D3
  ELSE
T=( A3*C2-A2*C3 ) / D3
  END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EXISTENCEEDGE2D(IN1,IN2,IN3,SCE,SCE1,SCE2,SCE3,EDGE)
INTEGER IN1,IN2,IN3,SCE,SCE1,SCE2,SCE3,EDGE(80000,3)
C
IMPLICIT NONE
INTEGER K,F1,F2
C
SCE1=0
SCE2=0
SCE3=0
DO 10 K=1,SCE
  F1=EDGE(K,1)
  F2=EDGE(K,2)
  IF ( ((IN1.EQ.F1).OR.(IN1.EQ.F2)).AND.
     *          ((IN2.EQ.F1).OR.(IN2.EQ.F2)) ) SCE1=K
  IF ( ((IN2.EQ.F1).OR.(IN2.EQ.F2)).AND.
     *          ((IN3.EQ.F1).OR.(IN3.EQ.F2)) ) SCE2=K
  IF ( ((IN1.EQ.F1).OR.(IN1.EQ.F2)).AND.
     *          ((IN3.EQ.F1).OR.(IN3.EQ.F2)) ) SCE3=K
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EXISTENCEEDGE(IN1,IN2,IN3,NEDG,SCE1,SCE2,SCE3,VICEDG,EDGE)
INTEGER IN1,IN2,IN3,NEDG,SCE1,SCE2,SCE3,VICEDG(1000),EDGE(80000,3)
C
IMPLICIT NONE
INTEGER K,F1,F2,E
C
SCE1=0
SCE2=0
SCE3=0
DO 10 K=1,NEDG
  E=VICEDG(K)
  F1=EDGE(E,1)
  F2=EDGE(E,2)
  IF ( ((IN1.EQ.F1).OR.(IN1.EQ.F2)).AND.
     *          ((IN2.EQ.F1).OR.(IN2.EQ.F2)) ) SCE1=E
  IF ( ((IN2.EQ.F1).OR.(IN2.EQ.F2)).AND.
     *          ((IN3.EQ.F1).OR.(IN3.EQ.F2)) ) SCE2=E
  IF ( ((IN1.EQ.F1).OR.(IN1.EQ.F2)).AND.
     *          ((IN3.EQ.F1).OR.(IN3.EQ.F2)) ) SCE3=E
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OCCURENCEF(SCE1,SCF,OCCUR,FACE)
INTEGER SCE1,SCF,OCCUR,FACE(60000,4)
C
IMPLICIT NONE
INTEGER K,F1,F2,F3
C
OCCUR=0
DO 10 K=1,SCF
  F1=FACE(K,1)
  F2=FACE(K,2)
  F3=FACE(K,3)
  IF ( (F1.EQ.SCE1).OR.(F2.EQ.SCE1).OR.(F3.EQ.SCE1) ) OCCUR=OCCUR+1
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OCCURENCEPVIC(SCF1,NPYR,OCCUR,PYR,PYRCOM,VICPYR)
INTEGER SCF1,NPYR,OCCUR,PYR(40000,5),PYRCOM,VICPYR(400)
C
IMPLICIT NONE
INTEGER K,F1,F2,F3,F4,F
C
PYRCOM=0
OCCUR=0
DO 10 K=1,NPYR
      F=VICPYR(K)
  F1=PYR(F,1)
  F2=PYR(F,2)
  F3=PYR(F,3)
  F4=PYR(F,4)
  IF ( (F1.EQ.SCF1).OR.(F2.EQ.SCF1).OR.
     *          (F3.EQ.SCF1).OR.(F4.EQ.SCF1) ) THEN
OCCUR=OCCUR+1
PYRCOM=F
    END IF
c IF (PYRCOM.NE.0) GOTO 20
10 CONTINUE
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OCCURENCEP(SCF1,SCP,OCCUR,PYR,PYRCOM)
INTEGER SCF1,SCP,OCCUR,PYR(40000,5),PYRCOM
C
IMPLICIT NONE
INTEGER K,F1,F2,F3,F4
C
PYRCOM=0
OCCUR=0
DO 10 K=1,SCP
  F1=PYR(K,1)
  F2=PYR(K,2)
  F3=PYR(K,3)
  F4=PYR(K,4)
  IF ( (F1.EQ.SCF1).OR.(F2.EQ.SCF1).OR.
     *          (F3.EQ.SCF1).OR.(F4.EQ.SCF1) ) THEN
```

```
OCCUR=OCCUR+1
PYRCOM=K
    END IF
e IF (PYRCOM.NE.0) GOTO 20
10 CONTINUE
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CREATEFACE(SCE1,SCE2,SCE3,SCF,FACE,IN1,IN2,IN3,PID)
INTEGER SCE1,SCE2,SCE3,SCF,FACE(60000,4),PID(30000),IN1,IN2,IN3
IMPLICIT NONE
C
INTEGER FACEID
FACE(SCF,1)=SCE1
FACE(SCF,2)=SCE2
FACE(SCF,3)=SCE3
FACE(SCF,4)=FACEID(PID(IN1),PID(IN2),PID(IN3))
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CREATEEDGE(SCE,IND1,IND2,PID,EDGE)
INTEGER SCE,IND1,IND2,PID(30000),EDGE(60000,3)
C
IMPLICIT NONE
INTEGER EDGEID
EDGE(SCE,1)=IND1
EDGE(SCE,2)=IND2
EDGE(SCE,3)=EDGEID(PID(IND1),PID(IND2))
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CHECKEDGE(DSCE,SCE,SC,IN1,IN2,PID,EDGE,N,LSTAK,NEDG,VICEDG)
INTEGER DSCE,SCE,SC,IN1,IN2,PID(30000),EDGE(60000,3),N
INTEGER LSTAK(6000),NEDG,VICEDG(1000)
IMPLICIT NONE
C
IF (DSCE.EQ.0) THEN
  SCE=SCE+1
  DSCE=SCE
  NEDG=NEDG+1
  VICEDG(NEDG)=SCE
  CALL CREATEEDGE(DSCE,IN1,IN2,PID,EDGE)
  IF (N.EQ.1) THEN
LSTAK(SC)=SCE
SC=SC+1
      END IF
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CHECKEDGE2D(DSCE,SCE,SC,IN1,IN2,PID,EDGE,N,LSTAK)
INTEGER DSCE,SCE,SC,IN1,IN2,PID(30000),EDGE(60000,3),N
INTEGER LSTAK(6000)
IMPLICIT NONE
C
IF (DSCE.EQ.0) THEN
  SCE=SCE+1
  DSCE=SCE
  CALL CREATEEDGE(DSCE,IN1,IN2,PID,EDGE)
  IF (N.EQ.1) THEN
LSTAK(SC)=SCE
SC=SC+1
      END IF
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
REAL FUNCTION ANGLE(VEC1,VEC2,VEC)
REAL VEC1(3),VEC2(3),VEC(3)
C
IMPLICIT NONE
INTEGER I
REAL SUM1,SUM2,MAG1,MAG2,COSINE2
REAL MAG,SUM,COSINE3,ANG
C
SUM1=0
MAG1=0
MAG2=0
SUM2=0
SUM=0
MAG=0
DO 10 I=1,3
  SUM1=SUM1+VEC1(I)*VEC1(I)
  SUM2=SUM2+VEC2(I)*VEC1(I)
  SUM=SUM+VEC2(I)*VEC(I)
  MAG=VEC(I)**2+MAG
  MAG1=VEC1(I)**2+MAG1
10   MAG2=VEC2(I)**2+MAG2
COSINE2=SUM2/SQRT(MAG2)/SQRT(MAG1)
COSINE3=SUM/SQRT(MAG)/SQRT(MAG2)
ANG=ATAN2(COSINE3,COSINE2)*180.0/3.141592654
ANG=INT(ANG*100)/100
DO WHILE (ANG.LT.0.0)
  ANG=ANG+360.0
END DO
ANGLE=ANG
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE SPECIALORDER(CENT,IND,IN1,IN2,NO,VICIOL,P)
INTEGER IND,IN1,IN2,NO,VICIOL(20)
REAL P(30000,3),CENT(3)
C
IMPLICIT NONE
REAL DDIR(3),TET(20),DUM(3),DIR1(3),DIR(20,3),ANGLE
```

```
REAL AP,BP,CP,DP,DIR2(3),NDIR1(3),TEMP1
INTEGER I,J,I1,TEMP
C
DO 10 I=1,NO
  I1=VICIOL(I)
  CALL ONEDIM(P(I1,1),P(I1,2),P(I1,3),DUM)
  CALL DIRECTION(DUM,CENT,DDIR)
  DIR(I,1)=DDIR(1)
  DIR(I,2)=DDIR(2)
  DIR(I,3)=DDIR(3)
10 CONTINUE
CALL ONEDIM(DIR(1,1),DIR(1,2),DIR(1,3),DIR1)
CALL ONEDIM(DIR(2,1),DIR(2,2),DIR(2,3),DIR2)
CALL PLANEPAR3(DIR2,DDIR,DIR1,AP,BP,CP,DP)
CALL NORMALVEC(DIR1,NDIR1,AP,BP,CP)
DO 20 I=1,NO
  CALL ONEDIM(DIR(I,1),DIR(I,2),DIR(I,3),DUM)
20 TET(I)=ANGLE(DIR1,DUM,NDIR1)
C
DO 30 J=1,(NO-1)
  DO 40 I=(J+1),NO
    IF (TET(I).LT.TET(J)) THEN
    TEMP1=TET(J)
    TET(J)=TET(I)
    TET(I)=TEMP1
    TEMP=VICIOL(J)
    VICIOL(J)=VICIOL(I)
    VICIOL(I)=TEMP
      END IF
40   CONTINUE
30 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE TRANSFER(VEC1,VEC2)
REAL VEC1(3),VEC2(3)
C
IMPLICIT NONE
INTEGER I
DO 10 I=1,3
10 VEC2(I)=VEC1(I)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION COLINEAR(DUM1,DUM2,DUM3)
REAL DUM1(3),DUM2(3),DUM3(3)
C
IMPLICIT NONE
REAL NUM1,NUM2,NUM3,DEN1,DEN2,DEN3,RAT1,RAT2,RAT3,X(3)
INTEGER I
C
DO 10 I=1,3
  X(I)=0
  IF ( (DUM1(I).EQ.DUM2(I)).AND.(DUM1(I).EQ.DUM3(I)) ) X(I)=1
10 CONTINUE
COLINEAR=.FALSE.
NUM1=DUM2(1)-DUM1(1)
NUM2=DUM2(2)-DUM1(2)
NUM3=DUM2(3)-DUM1(3)
DEN1=DUM3(1)-DUM1(1)
DEN2=DUM3(2)-DUM1(2)
DEN3=DUM3(3)-DUM1(3)
RAT1=NUM1/DEN1
RAT2=NUM2/DEN2
RAT3=NUM3/DEN3
IF ((X(1)+X(2)+X(3)).GT.1) THEN
   COLINEAR=.TRUE.
ELSE IF (X(1).EQ.1) THEN
  IF (ABS(RAT2-RAT3).LT.0.0001) COLINEAR=.TRUE.
ELSE IF (X(2).EQ.1) THEN
  IF (ABS(RAT1-RAT3).LT.0.0001) COLINEAR=.TRUE.
ELSE IF (X(3).EQ.1) THEN
  IF (ABS(RAT2-RAT1).LT.0.0001) COLINEAR=.TRUE.
ELSE IF ((ABS(RAT1-RAT2).LT.0.0001)
     *          .AND.(ABS(RAT1-RAT3).LT.0.0001)) THEN
 COLINEAR=.TRUE.
END IF
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
INTEGER EDGE(50000,3),FACE(65000,4),SCF1,IN1,IN2,IN3
C
IMPLICIT NONE
INTEGER TRY,I,J
LOGICAL FOUND,NOUV
C
IN1=0
IN2=0
IN3=0
I=1
J=1
FOUND=((IN1.NE.0).AND.(IN2.NE.0).AND.(IN3.NE.0))
IN1=EDGE(FACE(SCF1,I),J)
DO WHILE (.NOT.(FOUND))
 IF (J.EQ.3) THEN
   J=0
   I=I+1
 END IF
 J=J+1
 TRY=EDGE(FACE(SCF1,I),J)
 NOUV=((TRY.NE.IN1).AND.(TRY.NE.IN2).AND.(TRY.NE.IN3))
 IF (NOUV) THEN
   IF (IN2.EQ.0) THEN
IN2=TRY
   ELSE
IN3=TRY
   END IF
```

```
 END IF
 FOUND=((IN1.NE.0).AND.(IN2.NE.0).AND.(IN3.NE.0))
END DO
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION COPLANAR(DUM1,DUM2,DUM3,DUM4)
REAL DUM1(3),DUM2(3),DUM3(3),DUM4(3)
C
IMPLICIT NONE
REAL MAT(4,4),DET,DM1(4),DM2(4),DM3(4),DM4(4)
INTEGER I
C
  DO 10 I=1,3
    DM1(I)=DUM1(I)
    DM2(I)=DUM2(I)
    DM3(I)=DUM3(I)
10    DM4(I)=DUM4(I)
  DM1(4)=1.0
  DM2(4)=1.0
  DM3(4)=1.0
  DM4(4)=1.0
COPLANAR=.FALSE.
CALL MAKEMAT4(DM1,DM2,DM3,DM4,MAT)
CALL DETERMINANT(MAT,4,4,DET)
IF (ABS(DET).LT.0.00001) COPLANAR=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EXISTENCEPYR(SCF1,SCF2,SCF3,SCF4,PYR,SCPOL,NPYR,VICPYR)
INTEGER SCF1,SCF2,SCF3,SCF4,NPYR,PYR(40000,5),SCPOL,VICPYR(400)
C
IMPLICIT NONE
INTEGER K,P1,P2,P3,P4,P
LOGICAL VAL1,VAL2,VAL3,VAL4
SCPOL=0
DO 10 K=1,NPYR
  P=VICPYR(K)
  P1=PYR(P,1)
  P2=PYR(P,2)
  P3=PYR(P,3)
  P4=PYR(P,4)
  VAL1=(P1.EQ.SCF1).OR.(P1.EQ.SCF2).OR.(P1.EQ.SCF3).OR.(P1.EQ.SCF4)
  VAL2=(P2.EQ.SCF1).OR.(P2.EQ.SCF2).OR.(P2.EQ.SCF3).OR.(P2.EQ.SCF4)
  VAL3=(P3.EQ.SCF1).OR.(P3.EQ.SCF2).OR.(P3.EQ.SCF3).OR.(P3.EQ.SCF4)
  VAL4=(P4.EQ.SCF1).OR.(P4.EQ.SCF2).OR.(P4.EQ.SCF3).OR.(P4.EQ.SCF4)
        IF ( VAL1.AND.VAL2.AND.VAL3.AND.VAL4 )   SCPOL=P
  IF (SCPOL.NE.0) GOTO 20
10 CONTINUE
20 END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CHECKPYR(SCF1,SCF2,SCF3,SCF4,IN1,IN2,IN3,IN4,SCP,
    *                               SCPOL,PID,PYR)
INTEGER SCF1,SCF2,SCF3,SCF4,SCPOL,SCP,PYR(40000,5),PID(30000)
INTEGER IN1,IN2,IN3,IN4
C
IMPLICIT NONE
INTEGER PYRID
IF (SCPOL.EQ.0) THEN
  SCP=SCP+1
  PYR(SCP,1)=SCF1
  PYR(SCP,2)=SCF2
  PYR(SCP,3)=SCF3
  PYR(SCP,4)=SCF4
  PYR(SCP,5)=PYRID(PID(IN1),PID(IN2),PID(IN3),PID(IN4))
END IF
END
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EXISTENCEFACE2D(SCE1,SCE2,SCE3,SCF,SCFOL,FACE)
INTEGER SCE1,SCE2,SCE3,SCFOL,FACE(65000,4),SCF
C
IMPLICIT NONE
INTEGER K,F1,F2,F3
SCFOL=0
DO 10 K=1,SCF
  F1=FACE(K,1)
  F2=FACE(K,2)
  F3=FACE(K,3)
  IF ( ((SCE1.EQ.F1).OR.(SCE1.EQ.F2).OR.(SCE1.EQ.F3)).AND.
    *        ((SCE2.EQ.F1).OR.(SCE2.EQ.F2).OR.(SCE2.EQ.F3)).AND.
    *   ((SCE3.EQ.F1).OR.(SCE3.EQ.F2).OR.(SCE3.EQ.F3)) )  SCFOL=K
  IF (SCFOL.NE.0) GOTO 20
10 CONTINUE
20  END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EXISTENCEFACE(SCE1,SCE2,SCE3,NFAC,SCFOL,VICFAC,FACE)
INTEGER SCE1,SCE2,SCE3,SCFOL,VICFAC(2000),NFAC,FACE(65000,4)
C
IMPLICIT NONE
INTEGER K,F1,F2,F3,F
SCFOL=0
DO 10 K=1,NFAC
  F=VICFAC(K)
  F1=FACE(F,1)
  F2=FACE(F,2)
  F3=FACE(F,3)
  IF ( ((SCE1.EQ.F1).OR.(SCE1.EQ.F2).OR.(SCE1.EQ.F3)).AND.
    *        ((SCE2.EQ.F1).OR.(SCE2.EQ.F2).OR.(SCE2.EQ.F3)).AND.
    *   ((SCE3.EQ.F1).OR.(SCE3.EQ.F2).OR.(SCE3.EQ.F3)) )  SCFOL=F
  IF (SCFOL.NE.0) GOTO 20
10 CONTINUE
20  END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C
SUBROUTINE CHECKFACE(SCE1,SCE2,SCE3,IN1,IN2,IN3,SCF,SCFOL,
     *                FACE,PID,N,LSTAK,SC,NFAC,VICFAC)
INTEGER SCE1,SCE2,SCE3,SCF,SCFOL,FACE(65000,4),PID(30000),N
INTEGER LSTAK(5000),SC,IN1,IN2,IN3,NFAC,VICFAC(2000)
C
IMPLICIT NONE
INTEGER I,J
IF (SCFOL.EQ.0) THEN
  SCF=SCF+1
  SCFOL=SCF
  NFAC=NFAC+1
  VICFAC(NFAC)=SCF
  CALL CREATEFACE(SCE1,SCE2,SCE3,SCF,FACE,IN1,IN2,IN3,PID)
  IF (N.EQ.1) THEN
    LSTAK(SC)=SCF
    SC=SC+1
  END IF
ELSE
  DO 10 I=1,SC
    IF (LSTAK(I).EQ.SCFOL) THEN
      DO 20 J=I,SC-1
20        LSTAK(J)=LSTAK(J+1)
      SC=SC-1
    END IF
10  CONTINUE
END IF
30 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE CHECKFACE2D(SCE1,SCE2,SCE3,IN1,IN2,IN3,SCF,SCFOL,
     *                        FACE,PID,N,LSTAK,SC)
INTEGER SCE1,SCE2,SCE3,SCF,SCFOL,FACE(65000,4),PID(30000),N
INTEGER LSTAK(5000),SC,IN1,IN2,IN3
C
IMPLICIT NONE
IF (SCFOL.EQ.0) THEN
  SCF=SCF+1
  SCFOL=SCF
  CALL CREATEFACE(SCE1,SCE2,SCE3,SCF,FACE,IN1,IN2,IN3,PID)
  IF (N.EQ.1) THEN
    LSTAK(SC)=SCF
    SC=SC+1
  END IF
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


C
SUBROUTINE MODIFYPOINTSUR(NPS,PID)
INTEGER NPS,PID(30000)
C
IMPLICIT NONE
INTEGER I
DO 10 I=1,NPS
10 PID(I)=PID(I)-INT(PID(I)/10)*10
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE NOOVERLAP(EDGE,FACE,SCF1,DUMP,P,PYR,SCP,CANTUSE,
     *                    VICPYR,NPYR)
INTEGER EDGE(50000,3),FACE(65000,4),SCF1,PYR(40000,5),SCP
INTEGER VICPYR(400),NPYR
REAL P(30000,3),DUMP(3)
LOGICAL CANTUSE
C
IMPLICIT NONE
INTEGER I,K,IN1,IN2,IN3
INTEGER NODE(4),L
REAL DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3)
REAL AAP(4),BP(4),CP(4),DP(4),POINT(44,3)
REAL DIR1(3),CENT(3),POIN(1,3)
LOGICAL VAL(4),ABOVE
C
CANTUSE=.FALSE.
CALL BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
CALL MIDPOI(DUMP1,DUMP2,CENT)
CALL MIDPOI(CENT,DUMP3,CENT)
CALL DIRECTION(CENT,DUMP,DIR1)
DO 50 I=0,10
  K=I+1
  POINT(K,1)=DUMP(1)+(0.002+I*0.0996)*DIR1(1)
  POINT(K,2)=DUMP(2)+(0.002+I*0.0996)*DIR1(2)
50   POINT(K,3)=DUMP(3)+(0.002+I*0.0996)*DIR1(3)
CALL MIDPOI(DUMP1,DUMP,CENT)
CALL MIDPOI(CENT,DUMP2,CENT)
CALL DIRECTION(CENT,DUMP3,DIR1)
DO 51 I=0,10
  K=I+12
  POINT(K,1)=DUMP3(1)+(0.002+I*0.0996)*DIR1(1)
  POINT(K,2)=DUMP3(2)+(0.002+I*0.0996)*DIR1(2)
51   POINT(K,3)=DUMP3(3)+(0.002+I*0.0996)*DIR1(3)
CALL MIDPOI(DUMP1,DUMP,CENT)
CALL MIDPOI(CENT,DUMP3,CENT)
CALL DIRECTION(CENT,DUMP2,DIR1)
DO 52 I=0,10
  K=I+23
  POINT(K,1)=DUMP2(1)+(0.002+I*0.0996)*DIR1(1)
  POINT(K,2)=DUMP2(2)+(0.002+I*0.0996)*DIR1(2)
52   POINT(K,3)=DUMP2(3)+(0.002+I*0.0996)*DIR1(3)
CALL MIDPOI(DUMP2,DUMP,CENT)
CALL MIDPOI(CENT,DUMP3,CENT)
CALL DIRECTION(CENT,DUMP1,DIR1)
DO 53 I=0,10
  K=I+34
  POINT(K,1)=DUMP1(1)+(0.002+I*0.0996)*DIR1(1)
```

```
  POINT(K,2)=DUMP1(2)+(0.002+I*0.0996)*DIR1(2)
83   POINT(K,3)=DUMP1(3)+(0.002+I*0.0996)*DIR1(3)
     DO 45 K=1,NPYR
  CALL PYRANODE(EDGE,FACE,PYR,VICPYR(K),NODE)
  CALL ONEDIM(P(NODE(1),1),P(NODE(1),2),P(NODE(1),3),DUMP1)
  CALL ONEDIM(P(NODE(2),1),P(NODE(2),2),P(NODE(2),3),DUMP2)
  CALL ONEDIM(P(NODE(3),1),P(NODE(3),2),P(NODE(3),3),DUMP3)
  CALL ONEDIM(P(NODE(4),1),P(NODE(4),2),P(NODE(4),3),DUMP4)
  CALL PARPLANES(DUMP1,DUMP2,DUMP3,DUMP4,AAP,BP,CP,DP)
  DO 10 I=1,44
    POIN(1,1)=POINT(I,1)
    POIN(1,2)=POINT(I,2)
    POIN(1,3)=POINT(I,3)
    DO 60 L=1,6
60       VAL(L)=ABOVE(POIN,AAP(L),BP(L),CP(L),DP(L))
    IF (VAL(1).AND.VAL(2).AND.VAL(3).AND.VAL(4)) CANTUSE=.TRUE.
  IF (CANTUSE) GOTO 70
10   CONTINUE
45 CONTINUE
70 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION FACEOVERLAP(DUMP1,DUMP2,DUMP3,DUMP4)
REAL DUMP1(3),DUMP2(3),DUMP3(3),DUMP4(3)
C
IMPLICIT NONE
REAL VAL1,VAL2,MAT(3,3),DUM1(3),DUM2(3),DUM3(3),DUM4(3)
INTEGER I
LOGICAL OVER,UNDER
C
FACEOVERLAP=.FALSE.
DO 10 I=1,3
  DUM1(I)=DUMP1(I)+0.0056474
  DUM2(I)=DUMP2(I)+0.0056474
  DUM3(I)=DUMP3(I)+0.0056474
10   DUM4(I)=DUMP4(I)+0.0056474
CALL MAKEMAT3(DUM1,DUM2,DUM3,MAT)
CALL DETERMINANT(MAT,3,3,VAL1)
CALL MAKEMAT3(DUM1,DUM2,DUM4,MAT)
CALL DETERMINANT(MAT,3,3,VAL2)
c I commented this and it worked!
c IF (VAL2.GT.0) FACEOVERLAP=.TRUE.
OVER=(VAL1.GT.0).AND.(VAL2.GE.0)
UNDER=(VAL1.LT.0).AND.(VAL2.LE.0)
IF (OVER.OR.UNDER)  FACEOVERLAP=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EDGEINCOMMON(VICFAC,NFAC,NF,AF,E,FACE)
INTEGER VICFAC(2000),NFAC,NF,AF(300),E,FACE(65000,4)
C
IMPLICIT NONE
INTEGER I,E1,E2,E3
LOGICAL SAME
NF=0
C DO 10 I=1,SCF
DO 10 I=1,NFAC
  E1=FACE(VICFAC(I),1)
  E2=FACE(VICFAC(I),2)
  E3=FACE(VICFAC(I),3)
  SAME=(E.EQ.E1).OR.(E.EQ.E2).OR.(E.EQ.E3)
  IF ( SAME) THEN
NF=NF+1
AF(NF)=VICFAC(I)
END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE EDGEINCOMMON2D(FACE,SCF,NF,AF,E)
INTEGER FACE(65000,4),SCF,NF,AF(300),E
C
IMPLICIT NONE
INTEGER I,E1,E2,E3
LOGICAL SAME
NF=0
DO 10 I=1,SCF
  E1=FACE(I,1)
  E2=FACE(I,2)
  E3=FACE(I,3)
  SAME=(E.EQ.E1).OR.(E.EQ.E2).OR.(E.EQ.E3)
  IF ( SAME) THEN
NF=NF+1
AF(NF)=I
END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION EQUALCENTER(CENT,CENT1)
REAL CENT(3),CENT1(3)
C
IMPLICIT NONE
REAL VAL(3)
INTEGER I
EQUALCENTER=.FALSE.
DO 10 I=1,3
10 VAL(I)=ABS(CENT(I)-CENT1(I))
IF ((VAL(1).LT.0.001).AND.(VAL(2).LT.0.001).AND.(VAL(3).LT.0.001))
     *                        EQUALCENTER=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GETDUMPS(P,IN1,IN2,IN3,DUMP1,DUMP2,DUMP3)
INTEGER IN1,IN2,IN3
REAL P(30000,3),DUMP1(3),DUMP2(3),DUMP3(3)
C
```

```
IMPLICIT NONE
CALL ONEDIM(P(IN1,1),P(IN1,2),P(IN1,3),DUMP1)
CALL ONEDIM(P(IN2,1),P(IN2,2),P(IN2,3),DUMP2)
CALL ONEDIM(P(IN3,1),P(IN3,2),P(IN3,3),DUMP3)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION EQUALFACE(F,FACE,EDGE,IN11,IN22,IN33)
INTEGER F,FACE(55000,4),EDGE(50000,3),IN11,IN22,IN33
C
IMPLICIT NONE
LOGICAL VAL1,VAL2,VAL3
INTEGER IN1,IN2,IN3
EQUALFACE=.FALSE.
CALL BASE3NODE(EDGE,FACE,F,IN1,IN2,IN3)
VAL1=(IN11.EQ.IN1).OR.(IN11.EQ.IN2).OR.(IN11.EQ.IN3)
VAL2=(IN22.EQ.IN1).OR.(IN22.EQ.IN2).OR.(IN22.EQ.IN3)
VAL3=(IN33.EQ.IN1).OR.(IN33.EQ.IN2).OR.(IN33.EQ.IN3)
IF ((VAL1).AND.(VAL2).AND.(VAL3)) EQUALFACE=.TRUE.
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION VOLUME(COOR)
REAL COOR(4,3)
C
IMPLICIT NONE
REAL COOR1(4),COOR2(4),COOR3(4),COOR4(4),VOL,MAT4(4,4)
C
CALL ONEDIM4(COOR(1,1),COOR(1,2),COOR(1,3),1.0,COOR1)
CALL ONEDIM4(COOR(2,1),COOR(2,2),COOR(2,3),1.0,COOR2)
CALL ONEDIM4(COOR(3,1),COOR(3,2),COOR(3,3),1.0,COOR3)
CALL ONEDIM4(COOR(4,1),COOR(4,2),COOR(4,3),1.0,COOR4)
CALL MAKEMAT4(COOR1,COOR2,COOR3,COOR4,MAT4)
CALL DETERMINANT(MAT4,4,4,VOL)
VOLUME=ABS(VOL)/6
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE DUMP4PROJ(DUM,AP,BP,CP,DP,DUMP5)
REAL DUM(3),AP,BP,CP,DP,DUMP5(3)
C
IMPLICIT NONE
REAL DEN,NUM,T
C
DEN=AP**2+BP**2+CP**2
NUM=DUM(1)*AP+DUM(2)*BP+DUM(3)*CP+DP
T=-NUM/DEN
DUMP5(1)=DUM(1)+T*AP
DUMP5(2)=DUM(2)+T*BP
DUMP5(3)=DUM(3)+T*CP
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
LOGICAL FUNCTION PYRAMOVERLAP(EDGE,FACE,PYR,AP,BP,CP,DP,SCF1,
     *                        PYRCOM,IND1,P)
INTEGER EDGE(50000,3),FACE(55000,4),IND1,SCF1
INTEGER PYRCOM,PYR(40000,5)
REAL P(30000,3),AP,BP,CP,DP
C
IMPLICIT NONE
INTEGER NODE(4),I,IN1,IN2,IN3,IND2
REAL VAL1,VAL2
LOGICAL OVER,UNDER
C
PYRAMOVERLAP=.FALSE.
IF (PYRCOM.EQ.0) GOTO 20
CALL BASE3NODE(EDGE,FACE,SCF1,IN1,IN2,IN3)
CALL  PYR4NODE(EDGE,FACE,PYR,PYRCOM,NODE)
DO 10 I=1,4
10 IF ((NODE(I).NE.IN1).AND.(NODE(I).NE.IN2).AND.(NODE(I).NE.IN3))
     *                          IND2=NODE(I)
VAL1=AP*P(IND1,1)+BP*P(IND1,2)+CP*P(IND1,3)+DP
VAL2=AP*P(IND2,1)+BP*P(IND2,2)+CP*P(IND2,3)+DP
OVER=(VAL1.GE.0).AND.(VAL2.GT.0)
UNDER=(VAL1.LE.0).AND.(VAL2.LT.0)
IF (OVER.OR.UNDER)  PYRAMOVERLAP=.TRUE.
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE PYR4NODE(EDGE,FACE,PYR,PYRCOM,NODE)
INTEGER EDGE(50000,3),FACE(55000,4),PYRCOM,NODE(4),PYR(40000,5)
C
IMPLICIT NONE
INTEGER F1,F2,IN1,IN2,IN3,IN4,IN5,IN6,IN7
F1=PYR(PYRCOM,1)
F2=PYR(PYRCOM,2)
CALL BASE3NODE(EDGE,FACE,F1,IN1,IN2,IN3)
CALL BASE3NODE(EDGE,FACE,F2,IN4,IN5,IN6)
IF ((IN4.NE.IN1).AND.(IN4.NE.IN2).AND.(IN4.NE.IN3)) IN7=IN4
IF ((IN5.NE.IN1).AND.(IN5.NE.IN2).AND.(IN5.NE.IN3)) IN7=IN5
IF ((IN6.NE.IN1).AND.(IN6.NE.IN2).AND.(IN6.NE.IN3)) IN7=IN6
NODE(1)=IN1
NODE(2)=IN2
NODE(3)=IN3
NODE(4)=IN7
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE FACEINCOMMON(PYR,SCP,FAS,SCF1,NP,AP)
INTEGER PYR(40000,5),SCP,FAS,NP,AP(18),SCF1
IMPLICIT NONE
INTEGER I,F1,F2,F3,F4
NP=0
C
DO 10 I=1,SCP
  F1=PYR(I,1)
```

```
  F2=PYR(I,2)
  F3=PYR(I,3)
  F4=PYR(I,4)
  IF ((FAS.EQ.F1).OR.(FAS.EQ.F2).OR.(FAS.EQ.F3).OR.(FAS.EQ.F4)) THEN
IF (FAS.NE.SCF1) THEN
NP=NP+1
AP(NP)=I
    END IF
  END IF
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MESHREADER(NP,NPS,SCP,SCF,SCE,P,PID,EDGE,FACE,PYR,
   *         EXTRAPYR,FIL)
INTEGER NP,SCP,SCF,SCE,PID(30000),EDGE(50000,3),FACE(65000,4),NPS
INTEGER PYR(40000,5),EXTRAPYR(40000,6),FIL
REAL P(30000,3)
C
IMPLICIT NONE
  INTEGER OFFREC,REK,I
READ(FIL,REC=1) NPS
READ(FIL,REC=2) NP
READ(FIL,REC=3) SCP
READ(FIL,REC=4) SCF
READ(FIL,REC=5) SCE
OFFREC=5
DO 60 I=1,NPS
  REK=OFFREC+I
60   READ(FIL,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
OFFREC=REK
DO 70 I=NPS+1,NP
  REK=OFFREC+I-NPS
70   READ(FIL,REC=REK) P(I,1),P(I,2),P(I,3),PID(I)
OFFREC=REK
DO 20 I=1,SCE
   REK=OFFREC+I
20 READ(FIL,REC=REK) EDGE(I,1),EDGE(I,2),EDGE(I,3)
OFFREC=REK
DO 30 I=1,SCF
  REK=OFFREC+I
30 READ(FIL,REC=REK) FACE(I,1),FACE(I,2),FACE(I,3),FACE(I,4)
OFFREC=REK
DO 40 I=1,SCP
  REK=OFFREC+I
40 READ(FIL,REC=REK) PYR(I,1),PYR(I,2),PYR(I,3),PYR(I,4),
   *      PYR(I,5),EXTRAPYR(I,1),EXTRAPYR(I,2),EXTRAPYR(I,3),
   *      EXTRAPYR(I,4),EXTRAPYR(I,5),EXTRAPYR(I,6)
C
CLOSE(FIL)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

MESHCOMPSUPSG.F

```
SUBROUTINE OPERATORGRADGRAD(GRADGRAD,ALF)
REAL GRADGRAD(10,10,4,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,ALFR(2,5),ALFC(2,5),ALFCD(2,5),SIMP1,SIMP2
INTEGER  I,J,SIMP,DALFR(4,2,5),DALFC(4,2,5),ALFRD(2,5),ALFAD(4,5)
REAL INTEGRATIONVOL
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30      ALFR(I,J)=ALF(ROW,I,J)
20   CONTINUE
  DO 60 SIMP=1,4
    CALL DERIVATE(ALFR,ALFRD,SIMP,1)
    DO 70 I=1,2
      DO 80 J=1,5
80        DALFR(SIMP,I,J)=ALFRD(I,J)
70      CONTINUE
60   CONTINUE
  DO 90 COL=1,10
    DO 100 I=1,2
      DO 110 J=1,5
110        ALFC(I,J)=ALF(COL,I,J)
100     CONTINUE
    DO 120 SIMP=1,4
      CALL DERIVATE(ALFC,ALFCD,SIMP,1)
      DO 130 I=1,2
        DO 140 J=1,5
140          DALFC(SIMP,I,J)=ALFCD(I,J)
130       CONTINUE
120     CONTINUE
    DO 150 SIMP1=1,4
      DO 160 I=1,2
        DO 170 J=1,5
170          ALFRD(I,J)=DALFR(SIMP1,I,J)
160       CONTINUE
      DO 180 SIMP2=1,4
        DO 190 I=1,2
          DO 200 J=1,5
200            ALFCD(I,J)=DALFC(SIMP2,I,J)
190         CONTINUE
        CALL MULTIPLYALFA(ALFRD,ALFCD,ALFAD)
        GRADGRAD(ROW,COL,SIMP1,SIMP2)=INTEGRATIONVOL(ALFAD)
180       CONTINUE
150     CONTINUE
90   CONTINUE
10  CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C
SUBROUTINE ALFFIELD(ALFF,ALF)
REAL ALFF(10,10)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,ALFR(2,5),ALFC(2,5),I,J,ALFAD(4,5)
REAL INTEGRATIONVOL
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30    ALFR(I,J)=ALF(ROW,I,J)
20  CONTINUE
  DO 40 COL=1,10
    DO 50 I=1,2
      DO 60 J=1,5
60      ALFC(I,J)=ALF(COL,I,J)
50    CONTINUE
    CALL MULTIPLYALFA(ALFR,ALFC,ALFAD)
    ALFF(ROW,COL)=INTEGRATIONVOL(ALFAD)
40  CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ALFFOVSURF(ALFF,ALF)
REAL ALFF(10,10,1,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,ALFR(2,5),ALFC(2,5),I,J,ALFAD(4,5),SIMP
REAL INTEGRATIONSURF
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30    ALFR(I,J)=ALF(ROW,I,J)
20  CONTINUE
  DO 40 COL=1,10
    DO 50 I=1,2
      DO 60 J=1,5
60      ALFC(I,J)=ALF(COL,I,J)
50    CONTINUE
    CALL MULTIPLYALFA(ALFR,ALFC,ALFAD)
    DO 70 SIMP=1,4
70    ALFF(ROW,COL,1,SIMP)=INTEGRATIONSURF(ALFAD,SIMP)
40  CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE MULTIPLYALFA(ALFER1,ALFER2,ALFAD)
INTEGER ALFER1(2,5),ALFER2(2,5),ALFAD(4,5)
C
IMPLICIT NONE
INTEGER I,J,K,L
L=0
DO 20 I=1,2
  DO 30 J=1,2
    L=L+1
    ALFAD(L,5)=ALFER1(I,5)*ALFER2(J,5)
    DO 40 K=1,4
      ALFAD(L,K)=ALFER1(I,K)+ALFER2(J,K)
40      IF (ALFAD(L,5).EQ.0) ALFAD(L,K)=0
30  CONTINUE
20 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE DERIVATE(ALFJ,ALFJD,L,DER)
INTEGER ALFJ(2,5),ALFJD(2,5),L,DER
C
IMPLICIT NONE
INTEGER I,J,K
DO 10 J=1,2
  DO 20 K=1,5
20  ALFJD(J,K)=ALFJ(J,K)
10 CONTINUE
DO 30 J=1,DER
  DO 40 I=1,2
    ALFJD(I,5)=ALFJD(I,5)*ALFJD(I,L)
40  ALFJD(I,L)=ALFJD(I,L)-1
30 CONTINUE
DO 50 I=1,2
  IF (ALFJD(I,L).LT.0) THEN
  DO 60 J=1,5
60    ALFJD(I,J)=0
END IF
50 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OPERATORFIELDN(FIELDN,ALF)
REAL FIELDN(10,10,1,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,ALFR(2,5),ALFC(2,5),ALFCD(2,5),ALFAD(4,5),I,J
INTEGER SIMPX
REAL INTEGRATIONSURF
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30    ALFR(I,J)=ALF(ROW,I,J)
20  CONTINUE
  DO 40 COL=1,10
```

```
   DO 50 I=1,2
    DO 60 J=1,5
60      ALFC(I,J)=ALF(COL,I,J)
50    CONTINUE
  DO 70 SIMPX=1,4
    CALL DERIVATE(ALFC,ALFCD,SIMPX,1)
    CALL MULTIPLYALFA(ALFR,ALFCD,ALFAD)
   FIELDN(ROW,COL,1,SIMPX)=INTEGRATIONSURF(ALFAD,SIMPX)
70  CONTINUE
40  CONTINUE
10 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OPERATORALFNORMoriginal(NORM,ALF)
REAL NORM(10,4,1,1)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,I,J,ALFC(2,5),ALFR(2,5),ALFAD(4,5)
REAL INTEGRATIONSURF
C
DO 10 I=1,2
  DO 20 J=1,5
20  ALFC(I,J)=0
10 CONTINUE
ALFC(1,5)=1
DO 30 ROW=1,10
  DO 40 I=1,2
    DO 50 J=1,5
50    ALFR(I,J)=ALF(ROW,I,J)
40  CONTINUE
  CALL MULTIPLYALFA(ALFR,ALFC,ALFAD)
  DO 60 I=1,4
60   NORM(ROW,1,1,1)=INTEGRATIONSURF(ALFAD,I)
30 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OPERATORALFNORM(NORM,ALF)
REAL NORM(10,10,1,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,I,J,ALFC(2,5),ALFR(2,5),ALFAD(4,5),COL
REAL INTEGRATIONSURF
C
DO 5 ROW=1,10
  DO 10 I=1,2
    DO 20 J=1,5
20    ALFC(I,J)=ALF(ROW,I,J)
10  CONTINUE
  DO 30 COL=1,10
    DO 40 I=1,2
      DO 50 J=1,5
50      ALFR(I,J)=ALF(COL,I,J)
40    CONTINUE
    CALL MULTIPLYALFA(ALFR,ALFC,ALFAD)
    DO 60 I=1,4
60    NORM(ROW,COL,1,I)=INTEGRATIONSURF(ALFAD,I)
30  CONTINUE
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C


REAL FUNCTION INTEGRATIONVOL(ALFAD)
INTEGER ALFAD(4,5)
C
IMPLICIT NONE
REAL SUM,INTEGRATEVOL
INTEGER I
SUM=0.0
DO 10 I=1,4
SUM=SUM+INTEGRATEVOL(ALFAD(I,1),ALFAD(I,2),
     *                ALFAD(I,3),ALFAD(I,4))*ALFAD(I,5)
10 CONTINUE
INTEGRATIONVOL=SUM
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION INTEGRATEVOL(A1,A2,A3,A4)
INTEGER A1,A2,A3,A4
C
IMPLICIT NONE
INTEGER SUM,FAC
SUM=A1++AA4+3
INTEGRATEVOL=6.0*FAC(A1)*FAC(A2)*FAC(A3)*FAC(A4)/FAC(SUM)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
INTEGER FUNCTION FAC(VAL)
INTEGER VAL
C
IMPLICIT NONE
INTEGER SUM,I
SUM=1
DO 10 I=VAL,1,-1
10 SUM=SUM*I
FAC=SUM
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE COEFFICIENT(DUM1,DUM2,DUM3,AD,BD,CD,DD,DEN)
REAL DUM1(3),DUM2(3),DUM3(3),AD,BD,CD,DD,DEN
C
```

```
IMPLICIT NONE
REAL MAT3(3,3),DM1(3),DM2(3),DM3(3)
INTEGER I
C
DO 10 I=1,3
  DM1(I)=DUM1(I)
  DM2(I)=DUM2(I)
10 DM3(I)=DUM3(I)
CALL MAKEMAT3(DM1,DM2,DM3,MAT3)
CALL DETERMINANT(MAT3,3,3,AD)
AD=AD/DEN
DM1(1)=1.0
DM2(1)=1.0
DM3(1)=1.0
CALL MAKEMAT3(DM1,DM2,DM3,MAT3)
CALL DETERMINANT(MAT3,3,3,BD)
BD=-BD/DEN
  DM1(2)=DUM1(1)
DM2(2)=DUM2(1)
DM3(2)=DUM3(1)
CALL MAKEMAT3(DM1,DM2,DM3,MAT3)
CALL DETERMINANT(MAT3,3,3,CD)
CD=CD/DEN
  DM1(3)=DUM1(2)
DM2(3)=DUM3(2)
DM3(3)=DUM3(2)
CALL MAKEMAT3(DM1,DM2,DM3,MAT3)
CALL DETERMINANT(MAT3,3,3,DD)
DD=-DD/DEN
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
REAL FUNCTION INTEGRATIONSURF(ALFAD,SIMP)
INTEGER ALFAD(4,5),SIMP
C
IMPLICIT NONE
REAL SUM,INTEGRATESURF
INTEGER I,ALFAS(4,4)
SUM=0.0
CALL ALFSURF(ALFAD,4,ALFAS,SIMP)
DO 10 I=1,4
10 SUM=SUM+INTEGRATESURF( ALFAS(I,1),ALFAS(I,2),ALFAS(I,3) )*ALFAS(I,4)
INTEGRATIONSURF=SUM
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
  REAL FUNCTION INTEGRATESURF(A1,A2,A3)
INTEGER A1,A2,A3
C
IMPLICIT NONE
INTEGER SUM,FAC
SUM=A1+A2+A3+2
INTEGRATESURF=2.0*FAC(A1)*FAC(A2)*FAC(A3)/FAC(SUM)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ALFSURF(ALFAD,N,ALFAS,SIMP)
INTEGER ALFAD(N,5),SIMP,ALFAS(4,4),N
C
IMPLICIT NONE
INTEGER I,J
DO 10 I=1,N
ALFAS(I,4)=ALFAD(I,5)
10 IF ((ALFAD(I,SIMP).NE.0).OR.(ALFAD(I,5).EQ.0)) ALFAS(I,4)=0
DO 20 I=1,N
  DO 30 J=1,N-1
    IF (J.LT.SIMP) THEN
      ALFAS(I,J)=ALFAD(I,J)
    ELSE
      ALFAS(I,J)=ALFAD(I,J+1)
    END IF
30   CONTINUE
20 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE TRANSPOSE(MAT1,MAT2,L,S)
REAL MAT1(L,L,1,S),MAT2(L,L,1,S)
INTEGER L,S
C
IMPLICIT NONE
INTEGER I,J,K
DO 10 I=1,L
  DO 20 J=1,L
    DO 30 K=1,S
30            MAT2(J,I,1,K)=MAT1(I,J,1,K)
20   CONTINUE
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE OPERATORLAPLACIAN(LAPLACE4,ALF)
REAL LAPLACE4(10,10,1,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,SIMP,ALFR(2,5),ALFC(2,5),ALFCD(2,5)
INTEGER ALFAD(4,5),J,J
REAL INTEGRATIONVOL
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30     ALFR(I,J)=ALF(ROW,I,J)
20   CONTINUE
```

```
  DO 40 COL=1,10
    DO 50 I=1,2
      DO 60 J=1,5
60        ALFC(I,J)=ALF(COL,I,J)
50      CONTINUE
    DO 70 SIMP=1,4
      CALL DERIVATE(ALFC,ALFCD,SIMP,2)
      CALL MULTIPLYALFA(ALFR,ALFCD,ALFAD)
      LAPLACE4(ROW,COL,1,SIMP)=INTEGRATIONVOL(ALFAD)
70      CONTINUE
40    CONTINUE
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE BTLAPLACTAN(TANLAPLAC,ALF)
REAL TANLAPLAC(10,10,4,4)
INTEGER ALF(10,2,5)
C
IMPLICIT NONE
INTEGER ROW,COL,SIMP,ALFR(2,5),ALFC(2,5),ALFCD(2,5)
INTEGER ALFAD(4,5),I,J,SMPX
REAL INTEGRATIONSURF
C
DO 10 ROW=1,10
  DO 20 I=1,2
    DO 30 J=1,5
30      ALFR(I,J)=ALF(ROW,I,J)
20    CONTINUE
  DO 40 COL=1,10
    DO 50 I=1,2
      DO 60 J=1,5
60        ALFC(I,J)=ALF(COL,I,J)
50      CONTINUE
    DO 70 SIMP=1,4
      CALL DERIVATE(ALFC,ALFCD,SIMP,2)
      CALL MULTIPLYALFA(ALFR,ALFCD,ALFAD)
      DO 80 SMPX=1,4
80        TANLAPLAC(ROW,COL,SIMP,SMPX)=INTEGRATIONSURF(ALFAD,SMPX)
70      CONTINUE
40    CONTINUE
10 CONTINUE
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE ALLCOEFFICIENT(COOR,A,B,C,D)
REAL COOR(4,3),A(4),B(4),C(4),D(4)
C
IMPLICIT NONE
REAL DUM1(3),DUM2(3),DUM3(3),MAT4(4,4),AA,BB,CC,DD,DEN
REAL DUM14(4),DUM24(4),DUM34(4),DUM44(4)
C
CALL ONEDIM4(1.0,COOR(1,1),COOR(1,2),COOR(1,3),DUM14)
CALL ONEDIM4(1.0,COOR(2,1),COOR(2,2),COOR(2,3),DUM24)
CALL ONEDIM4(1.0,COOR(3,1),COOR(3,2),COOR(3,3),DUM34)
CALL ONEDIM4(1.0,COOR(4,1),COOR(4,2),COOR(4,3),DUM44)
CALL MAKEMAT4(DUM14,DUM24,DUM34,DUM44,MAT4)
CALL DETERMINANT(MAT4,4,4,DEN)
CALL ONEDIM(COOR(2,1),COOR(2,2),COOR(2,3),DUM1)
CALL ONEDIM(COOR(3,1),COOR(3,2),COOR(3,3),DUM2)
CALL ONEDIM(COOR(4,1),COOR(4,2),COOR(4,3),DUM3)
CALL COEFFICIENT(DUM1,DUM2,DUM3,AA,BB,CC,DD,DEN)
A(1)=AA
B(1)=BB
C(1)=CC
D(1)=DD
CALL ONEDIM(COOR(1,1),COOR(1,2),COOR(1,3),DUM1)
CALL COEFFICIENT(DUM1,DUM2,DUM3,AA,BB,CC,DD,DEN)
A(2)=-AA
B(2)=-BB
C(2)=-CC
D(2)=-DD
CALL ONEDIM(COOR(2,1),COOR(2,2),COOR(2,3),DUM2)
CALL COEFFICIENT(DUM1,DUM2,DUM3,AA,BB,CC,DD,DEN)
A(3)=AA
B(3)=BB
C(3)=CC
D(3)=DD
CALL ONEDIM(COOR(3,1),COOR(3,2),COOR(3,3),DUM3)
CALL COEFFICIT(DUM1,DUM2,DUM3,AA,BB,CC,DD,DEN)
A(4)=-AA
B(4)=-BB
C(4)=-CC
D(4)=-DD
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
INTEGER FUNCTION NORMALSIMPTOSURFACE(ND1,ND2,ND3,VERT)
INTEGER ND1,ND2,ND3,VERT(4)
C
IMPLICIT NONE
INTEGER I,V(4)
NORMALSIMPTOSURFACE=0
DO 10 I=1,4
  V(I)=0
10 IF ((VERT(I).EQ.ND1).OR.(VERT(I).EQ.ND2).OR.(VERT(I).EQ.ND3)) V(I)=1
DO 20 I=1,4
20 IF (V(I).EQ.0) NORMALSIMPTOSURFACE=1
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE FANCTIONS(ALF,EPSILON,BET)
INTEGER ALF(10,2,5)
COMPLEX EPSILON(10)
```

```
      REAL BET
C
      IMPLICIT NONE
      INTEGER I,N,IND
      REAL DX,DY,FREQ,MU,PI,EP
C
      OPEN(UNIT=3,FILE='functions.dat',STATUS='UNKNOWN',ACCESS=
     *     'SEQUENTIAL',FORM='FORMATTED')
      PI=ACOS(-1.0)
      EP=8.85E-12
      READ(3,100)
      DO 10 I=1,10
        READ(3,*) ALF(I,1,1),ALF(I,1,2),ALF(I,1,3),ALF(I,1,4),ALF(I,1,5)
        READ(3,*) ALF(I,2,1),ALF(I,2,2),ALF(I,2,3),ALF(I,2,4),ALF(I,2,5)
   10 CONTINUE
      READ(3,*)
      READ(3,*) FREQ
      READ(3,*) MU
      READ(3,*)
      READ(3,*) N
      BET=2*PI*FREQ*SQRT(MU*EP)
      DO 20 I=1,N
        READ(3,*) DX,DY,IND
        EPSILON(IND)=CMPLX(DX,DY)*BET**2
   20 CONTINUE
      CLOSE(3)
  100 FORMAT(5(/))
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE READBASICINFO(NP,SCE,SCF,SCP,FIL)
      INTEGER NP,SCE,SCF,SCP,FIL
C
      IMPLICIT NONE
      OPEN(UNIT=2,FILE='unmesh.dat',STATUS='UNKNOWN',ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=11,RECORDTYPE='FIXED')
      READ(2,REC=2) NP
      READ(2,REC=3) SCP
      READ(2,REC=4) SCF
      READ(2,REC=5) SCE
      FIL=2
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE READINFO(I,NP,SCE,SCF,EDGE,POINT,MEDP,
     *                    MEDP,MEDE,INDP,COOR,INDEX,FIL)
      REAL COOR(4,3)
      INTEGER I,NP,SCE,SCF,EDGE(4,3),POINT(4,3,2),MEDP,MEDP(4)
      INTEGER MEDE(4,3),INDP(6),INDEX(4),FIL
C
      IMPLICIT NONE
      INTEGER OFFSET,IND,J,K,FACE(4),IND1,IND2,ND
C
      IND=0
      OFFSET=NP+SCE+SCF+I+5
      READ(FIL,REC=OFFSET) FACE(1),FACE(2),FACE(3),FACE(4),MEDP,INDP(1),
     *     INDP(2),INDP(3),INDP(4),INDP(5),INDP(6)
      DO 10 J=1,4
        OFFSET=NP+SCE+FACE(J)+5
        READ(FIL,REC=OFFSET) EDGE(J,1),EDGE(J,2),EDGE(J,3),MEDF(J)
        DO 20 K=1,3
          OFFSET=NP+EDGE(J,K)+5
          READ(FIL,REC=OFFSET) POINT(J,K,1),POINT(J,K,2),MEDE(J,K)
   20   CONTINUE
   10 CONTINUE
      ND=0
      DO 40 J=1,4
       OFFSET=NP+5+INDP(J)
       READ(FIL,REC=OFFSET) IND1,IND2
       DO 50 K=1,ND
   50    IF (INDEX(K).EQ.IND1) GOTO 55
         ND=ND+1
         INDEX(ND)=IND1
   55  DO 60 K=1,ND
   60    IF (INDEX(K).EQ.IND2) GOTO 40
         ND=ND+1
         INDEX(ND)=IND2
       IF (ND.EQ.4) GOTO 45
   40 CONTINUE
   45 DO 70 J=1,4
        DO 80 K=1,3
          IND1=POINT(J,K,1)
          IND2=POINT(J,K,2)
          IF ( ((INDEX(1).EQ.IND1).AND.(INDEX(2).EQ.IND2)).OR.
     *       ((INDEX(1).EQ.IND2).AND.(INDEX(2).EQ.IND1)) ) INDP(1)=EDGE(J,K)
          IF ( ((INDEX(1).EQ.IND1).AND.(INDEX(3).EQ.IND2)).OR.
     *       ((INDEX(1).EQ.IND2).AND.(INDEX(3).EQ.IND1)) ) INDP(2)=EDGE(J,K)
          IF ( ((INDEX(1).EQ.IND1).AND.(INDEX(4).EQ.IND2)).OR.
     *       ((INDEX(1).EQ.IND2).AND.(INDEX(4).EQ.IND1)) ) INDP(3)=EDGE(J,K)
          IF ( ((INDEX(2).EQ.IND1).AND.(INDEX(3).EQ.IND2)).OR.
     *       ((INDEX(2).EQ.IND2).AND.(INDEX(3).EQ.IND1)) ) INDP(4)=EDGE(J,K)
          IF ( ((INDEX(3).EQ.IND1).AND.(INDEX(4).EQ.IND2)).OR.
     *       ((INDEX(3).EQ.IND2).AND.(INDEX(4).EQ.IND1)) ) INDP(5)=EDGE(J,K)
          IF ( ((INDEX(4).EQ.IND1).AND.(INDEX(2).EQ.IND2)).OR.
     *       ((INDEX(4).EQ.IND2).AND.(INDEX(2).EQ.IND1)) ) INDP(6)=EDGE(J,K)
   80   CONTINUE
   70 CONTINUE
      DO 30 J=1,4
        OFFSET=INDEX(J)+5
   30   READ(FIL,REC=OFFSET)  ,COOR(J,2),COOR(J,3)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE OPERATORS(ALF,COMPUTE)
      INTEGER ALF(10,2,5),COMPUTE
C
      IMPLICIT NONE
      REAL LAPLACIAN(10,10,1,4),TRANSLAPLACIAN(10,10,1,4),FIELDN(10,10,1,4)
      REAL ALFF(10,10,1,1),GRADGRAD(10,10,4,4),NORM(10,10,1,4)
c     REAL ALFF(10,10,1,1),GRADGRAD(10,10,4,4),NORM(10,4,1,1)
```

```
REAL TRANSFIELDN(10,10,1,4),TANLAPLAC(10,10,4,4),SURFALFF(10,10,1,4)
COMMON/OPERATEUR/ LAPLACIAN,TRANSLAPLACIAN,GRADGRAD,ALFF
     *        ,NORM,FIELDN,TRANSFIELDN,TANLAPLAC,SURFALFF
C
OPEN(UNIT=4,FILE='operateur.dat',STATUS='UNKNOWN',ACCESS=
     *    'SEQUENTIAL',FORM='FORMATTED')
IF (COMPUTE.EQ.1) THEN
 CALL OPERATORLAPLACIAN(LAPLACIAN,ALF)
 CALL WRITER(LAPLACIAN,10,10,1,4,'LAPLACIAN  ',4)
 CALL TRANSPOSE(LAPLACIAN,TRANSLAPLACIAN,10,4)
 CALL WRITER(TRANSLAPLACIAN,10,10,1,4,'TRANSLAPLN',4)
 CALL OPERATORGRADGRAD(GRADGRAD,ALF)
 CALL WRITER(GRADGRAD,10,10,4,4,'GRADIENT  ',4)
 CALL ALFFIELD(ALFF,ALF)
 CALL WRITER(ALFF,10,10,1,1,'ALFFIELD  ',4)
 CALL OPERATORALFNORM(NORM,ALF)
 CALL WRITER(NORM,10,10,1,4,'ALF*const',4)
c  CALL WRITER(NORM,10,4,1,1,'ALF*const',4)
 CALL OPERATORFIELDN(FIELDN,ALF)
 CALL WRITER(FIELDN,10,10,1,4,'ALFdFI/dn',4)
 CALL TRANSPOSE(FIELDN,TRANSFIELDN,10,4)
 CALL WRITER(TRANSFIELDN,10,10,1,4,'FIdALF/dn',4)
 CALL STLAPLACTAN(TANLAPLAC,ALF)
 CALL WRITER(TANLAPLAC,10,10,4,4,'TANLAPLAC',4)
 CALL ALFFOVSURF(SURFALFF,ALF)
 CALL WRITER(SURFALFF,10,10,1,4,'ALFFOVSURF',4)
ELSE
 CALL READER(LAPLACIAN,10,10,1,4,4)
 CALL READER(TRANSLAPLACIAN,10,10,1,4,4)
 CALL READER(GRADGRAD,10,10,4,4,4)
 CALL READER(ALFF,10,10,1,1,4)
 CALL READER(NORM,10,10,1,4,4)
 CALL READER(FIELDN,10,10,1,4,4)
END IF
CLOSE(4)
C
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

SOURCESUBS.F

```
CLIST OF SUBROUTINES USED COMMONLY BY SEVERAL PROGRAMS
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE 3 CARTESIAN COMPONENETS OF THE MAGNETIC FIELD
SUBROUTINE GETREALH(P,F,NP,NS,K)
REAL K,P(NP,3)
INTEGER NP,NS
COMPLEX F(NP,3)
C
IMPLICIT NONE
INTEGER I,AX
COMPLEX DIRICHLETVALX,DIRICHLETVALY,DIRICHLETVALZ
REAL FREQ,PI
C
PI=ACOS(-1.0)
FREQ=K/(2.0*PI*SQRT(8.85E-12*1.25664E-6))
DO 5 AX=1,3
  DO 10 I=1,N'J
    IF (AX.EQ.1) F(I,AX)=DIRICHLETVALX(P(I,1),P(I,2),P(I,3),FREQ)
    IF (AX.EQ.2) F(I,AX)=DIRICHLETVALY(P(I,1),P(I,2),P(I,3),FREQ)
    IF (AX.EQ.3) F(I,AX)=DIRICHLETVALZ(P(I,1),P(I,2),P(I,3),FREQ)
10   CONTINUE
5 CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE X-COMPONENT OF THE MAGNETIC FIELD
COMPLEX FUNCTION DIRICHLETVALX(X,Y,Z,FREQ)
REAL X,Y,Z,FREQ
C
IMPLICIT NONE
REAL MEASURERROR,PHASE,ANG,MAG,FAC
COMPLEX FILCONT,VAL
VAL=FILCONT(X,Y,Z,0.0,0.0, 0.7,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.0,0.0,-0.7,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.2,0.2, 0.6,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.2,0.2,-0.6,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.1,0.0, 0.7,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.1,0.0,-0.7,1,FREQ,1,3)
     VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.7,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,-0.2,0.0,-0.7,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.0,-0.2, 0.8,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.0,-0.2,-0.8,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.2,-0.18, 0.4,1,FREQ,1,3)+
    *      FILCONT(X,Y,Z,0.2,-0.18,-0.4,1,FREQ,1,3)
VAL=VAL+FILCONT(X,Y,Z,0.2,0.0, 0.8,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,0.2,0.0,-0.8,1,FREQ,-1,2)+
    *      FILCONT(X,Y,Z,0.0,0.2, 0.7,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,0.0,0.2,-0.7,1,FREQ,-1,2)+
    *      FILCONT(X,Y,Z,0.2,0.2, 0.4,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,0.2,0.2,-0.4,1,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,-0.2,-0.28, 0.4,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,-0.2,-0.28,-0.4,1,FREQ,-1,2)+
    *      FILCONT(X,Y,Z,0.0,0.0, 0.0,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,0.0,0.0,-0.0,1,FREQ,-1,2)+
    *      FILCONT(X,Y,Z,0.2,-0.1, 0.48,1,FREQ,1,2)+
    *      FILCONT(X,Y,Z,0.2,-0.1,-0.48,1,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,0.1,0.18, 0.7,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,0.1,0.18,-0.7,1,FREQ,-1,1)+
    *      FILCONT(X,Y,Z,0.1,0.0, 0.65,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,0.1,0.0,-0.65,1,FREQ,-1,1)+
    *      FILCONT(X,Y,Z,-0.1,-0.1, 0.8,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,-0.1,-0.1,-0.8,1,FREQ,-1,1)
VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.4,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,-0.2,0.0,-0.4,1,FREQ,-1,1)+
    *      FILCONT(X,Y,Z,0.1,-0.1, 0.7,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,0.1,-0.1,-0.7,1,FREQ,-1,1)+
    *      FILCONT(X,Y,Z,0.2,-0.2, 0.8,1,FREQ,1,1)+
    *      FILCONT(X,Y,Z,0.2,-0.2,-0.8,1,FREQ,-1,1)
```

```
C FAC=MEASURERROR(1)
FAC=1.0
ANG=PHASE(VAL)*FAC*3.1415926/180.0
MAG=CABS(VAL)*FAC
VAL=MAG*CMPLX(COS(ANG),SIN(ANG))
DIRICHLETVALX=10.0*VAL
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE Y-COMPONENT OF THE MAGNETIC FIELD
COMPLEX FUNCTION DIRICHLETVALY(X,Y,Z,FREQ)
REAL X,Y,Z,FREQ
C
IMPLICIT NONE
REAL MEASURERROR,PHASE,ANG,MAG,FAC
COMPLEX FILCONT,VAL
VAL=FILCONT(X,Y,Z,0.0,0.0, 0.7,2,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.0,0.0,-0.7,2,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.2,0.2, 0.6,2,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.2,0.2,-0.6,2,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.1,0.0, 0.7,2,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.1,0.0,-0.7,2,FREQ,1,3)
   VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.7,2,FREQ,1,3)+
  *        FILCONT(X,Y,Z,-0.2,0.0,-0.7,2,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.0,-0.2, 0.5,2,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.0,-0.2,-0.5,2,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.2,-0.15, 0.4,2,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.2,-0.15,-0.4,2,FREQ,1,3)
VAL=VAL+FILCONT(X,Y,Z,0.2,0.0, 0.6,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,0.0,-0.6,2,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.0,0.2, 0.7,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.0,0.2,-0.7,2,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.2,0.2, 0.4,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,0.2,-0.4,2,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,-0.2,-0.25, 0.4,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,-0.2,-0.25,-0.4,2,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.0,0.0, 0.0,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.0,0.0,-0.0,2,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.2,-0.1, 0.45,2,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,-0.1,-0.45,2,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,0.1,0.15, 0.7,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,0.15,-0.7,2,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.1,0.0, 0.65,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,0.0,-0.65,2,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,-0.1,-0.1, 0.5,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,-0.1,-0.1,-0.5,2,FREQ,-1,1)
VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.4,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,-0.2,0.0,-0.4,2,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.1,-0.1, 0.7,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,-0.1,-0.7,2,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.2,-0.2, 0.6,2,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.2,-0.2,-0.6,2,FREQ,-1,1)
C FAC=MEASURERROR(1)
FAC=1.0
ANG=PHASE(VAL)*FAC*3.1415926/180.0
MAG=CABS(VAL)*FAC
VAL=MAG*CMPLX(COS(ANG),SIN(ANG))
DIRICHLETVALY=10.0*VAL
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE Z-COMPONENT OF THE MAGNETIC FIELD
COMPLEX FUNCTION DIRICHLETVALZ(X,Y,Z,FREQ)
REAL X,Y,Z,FREQ
C
IMPLICIT NONE
REAL MEASURERROR,PHASE,ANG,MAG,FAC
COMPLEX FILCONT,VAL
VAL=FILCONT(X,Y,Z,0.0,0.0, 0.7,3,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.0,0.0,-0.7,3,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.2,0.2, 0.6,3,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.2,0.2,-0.6,3,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.1,0.0, 0.7,3,FREQ,1,3)+
  *     FILCONT(X,Y,Z,0.1,0.0,-0.7,3,FREQ,1,3)
   VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.7,3,FREQ,1,3)+
  *        FILCONT(X,Y,Z,-0.2,0.0,-0.7,3,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.0,-0.2, 0.5,3,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.0,-0.2,-0.5,3,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.2,-0.15, 0.4,3,FREQ,1,3)+
  *        FILCONT(X,Y,Z,0.2,-0.15,-0.4,3,FREQ,1,3)
VAL=VAL+FILCONT(X,Y,Z,0.2,0.0, 0.6,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,0.0,-0.6,3,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.0,0.2, 0.7,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.0,0.2,-0.7,3,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.2,0.2, 0.4,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,0.2,-0.4,3,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,-0.2,-0.25, 0.4,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,-0.2,-0.25,-0.4,3,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.0,0.0, 0.0,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.0,0.0,-0.0,3,FREQ,-1,2)+
  *        FILCONT(X,Y,Z,0.2,-0.1, 0.45,3,FREQ,1,2)+
  *        FILCONT(X,Y,Z,0.2,-0.1,-0.45,3,FREQ,-1,2)
VAL=VAL+FILCONT(X,Y,Z,0.1,0.15, 0.7,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,0.15,-0.7,3,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.1,0.0, 0.65,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,0.0,-0.65,3,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,-0.1,-0.1, 0.5,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,-0.1,-0.1,-0.5,3,FREQ,-1,1)
VAL=VAL+FILCONT(X,Y,Z,-0.2,0.0, 0.4,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,-0.2,0.0,-0.4,3,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.1,-0.1, 0.7,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.1,-0.1,-0.7,3,FREQ,-1,1)+
  *        FILCONT(X,Y,Z,0.2,-0.2, 0.6,3,FREQ,1,1)+
  *        FILCONT(X,Y,Z,0.2,-0.2,-0.6,3,FREQ,-1,1)
C FAC=MEASURERROR(1)
FAC=1.0
ANG=PHASE(VAL)*FAC*3.1415926/180.0
MAG=CABS(VAL)*FAC
VAL=MAG*CMPLX(COS(ANG),SIN(ANG))
DIRICHLETVALZ=10.0*VAL
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CRANDOM NUMBER GENERATOR SCALED FOR 25% ERROR(+-2dB)
  REAL FUNCTION MEASURERROR(X)
  INTEGER X
C
  IMPLICIT NONE
  INTEGER RAND
  MEASURERROR=(RAND(X)/71233.0)+0.79
  END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT ACTUALLY COMPUTETHE DIFFERENT CONTRIBUTIONS TO THE FIELD
CRETURNS APROPRIATE VALUE ONLY
CX,Y,Z IS THE OBSERVATION POINT
COX,OY,OZ IS THE SOURCE POINT
CAX IS THE COMPONENT CONTRIBUTION DESIRED
CDIR IS THE ORIENTATION OF THE IDEAL DIPOLE 1 IN SAX DIRCETION,-1 IN -SAX
CSAX IS THE AXIS ALONG WHICH THE DIPOLE IS DIRECTED
CALL VALUES ARE COMPUTED FROM A TRANSFORMATION, OF THE FIELD VALUES
CGENERATED BY A Z-DIRECTED DIPOLE.
COMPLEX FUNCTION FILCONT(X,Y,Z,OX,OY,OZ,AX,FREQ,DIR,SAX)
REAL X,Y,Z,OX,OY,OZ,FREQ
INTEGER AX,DIR,SAX
C
IMPLICIT NONE
REAL R,TET,FI
COMPLEX VALZ(3),VALX(3),VALY(3)
C
VALZ(AX)=CMPLX(0.0,0.0)
VALX(AX)=CMPLX(0.0,0.0)
VALY(AX)=CMPLX(0.0,0.0)
IF (SAX.EQ.3) THEN
  CALL CONVCARTOSFEZ(X,Y,Z,OX,OY,OZ,R,TET,FI)
  CALL POLE(R,TET,FI,FREQ,VALZ)
  VALZ(1)=VALZ(1)*SIN(TET)*SIN(FI)*DIR
  VALZ(2)=-VALZ(2)*SIN(TET)*COS(FI)*DIR
  VALZ(3)=VALZ(3)*0.0*DIR
ELSE IF (SAX.EQ.1) THEN
  CALL CONVCARTOSFEZ(X,Y,Z,OX,OY,OZ,R,TET,FI)
  CALL POLE(R,TET,FI,FREQ,VALX)
  VALX(1)=VALX(1)*0.0*DIR
  VALX(2)=VALX(2)*COS(TET)*DIR
  VALX(3)=-VALX(3)*SIN(TET)*SIN(FI)*DIR
ELSE
  CALL CONVCARTOSFEZ(X,Y,Z,OX,OY,OZ,R,TET,FI)
  CALL POLE(R,TET,FI,FREQ,VALY)
  VALY(1)=-VALY(1)*COS(TET)*DIR
  VALY(2)=VALY(2)*0.0*DIR
  VALY(3)=VALY(3)*SIN(TET)*COS(FI)*DIR
END IF
FILCONT=VALZ(AX)+VALX(AX)+VALY(AX)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES DHDN FROM THE 2 POINT FORWARD DERIVATIVE
SUBROUTINE GETREALDHDN1(P,GFN,NOR,NP,NS,K)
REAL P(NP,3),NOR(3),K
COMPLEX GFN(NP,3)
INTEGER NP,NS
C
IMPLICIT NONE
COMPLEX GHD(1,3),GH(1,3)
INTEGER I,AX,J,L
REAL PD(1,3),MN,P1(3),P2(3)
C
DO 10 I=1,NS
  DO 15 J=0,1,1
    DO 20 L=1,3
    PD(1,L)=P(I,L)+0.001*J*NOR(L)
    IF (J.EQ.0) P1(L)=PD(1,L)
20     IF (J.EQ.1) P2(L)=PD(1,L)
    IF (J.EQ.0) CALL GETREALH(PD,GH,1,1,K)
    IF (J.EQ.1) CALL GETREALH(PD,GHD,1,1,K)
15   CONTINUE
  MN=SQRT( (P2(1)-P1(1))**2+(P2(2)-P1(2))**2+(P2(3)-P1(3))**2)
  DO 5 AX=1,3
    GFN(I,AX)=(GHD(1,AX)-GH(1,AX))/MN
5   CONTINUE
10   CONTINUE
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CALCULATES THE FIELD VALUES DUE TO A Z-DIRECTED DIPOLE
SUBROUTINE POLE(R,TET,FI,FREQ,VAL3)
REAL FREQ,R,TET,FI
COMPLEX VAL3(3)
C
IMPLICIT NONE
COMPLEX IARGH,VALR,VAL
REAL K,PI
C
PI=ACOS(-1.0)
K=2.0*PI*FREQ*SQRT(8.85E-12*1.25664E-6)
IM=CMPLX(0.0,1.0)
c
ARGH=-IM*K*R
VAL=CEXP(ARGH)/R/4.0/PI
VALR=-(IM*K*VAL+VAL/R)
c      contribution from z oriented  ideal dip
VAL3(1)=VALR
VAL3(2)=VALR
VAL3(3)=VALR
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CONVERTS THE DISTANCE FROM OBSERVATION POINT TO SOURCE POINT
CTO SPHERICAL COORDIANTES
SUBROUTINE CONVCARTOSFEZ(X,Y,Z,OX,OY,OZ,R,TET,FI)
REAL X,Y,Z,R,TET,FI,OX,OY,OZ
C
IMPLICIT NONE
REAL DX,DY,DZ,PI
C
PI=ACOS(-1.0)
```

```
DX=X-OX
DY=Y-OY
DZ=Z-OZ
CALL CONVCARTOSFE(DX,DY,DZ,R,TET,FI,PI)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CONVERTS CARTESIAN COORDINATES TO SPHERICAL COORDINATES
SUBROUTINE CONVCARTOSFE(DX,DY,DZ,R,TET,FI,PI)
REAL DX,DY,DZ,R,TET,FI,PI
C
IMPLICIT NONE
C
 R=SQRT(DX**2+DY**2+DZ**2)
 TET=ACOS(DZ/R)
 IF (R.EQ.0) TET=0.0
 FI=ATAN(DY/DX)
 IF (DX.EQ.0) FI=0.0
 IF ((DX.EQ.0).AND.(DY.GT.0)) FI=PI/2
 IF ((DX.EQ.0).AND.(DY.LT.0)) FI=-PI/2
 IF ((DX.GT.0).AND.(DY.EQ.0)) FI=0.0
 IF ((DX.LT.0).AND.(DY.EQ.0)) FI=PI
 IF ((DX.LT.0).AND.(DY.LT.0)) FI=FI+PI
 IF ((DX.LT.0).AND.(DY.GT.0)) FI=FI+PI
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CORRECTS THE ANGLE RETURN BY THE ATAN FUNCTION
SUBROUTINE ADJUSTTAN(DX,DY,FI,PI)
REAL DX,DY,FI,PI
C
IMPLICIT NONE
C
 IF (DX.EQ.0) FI=0.0
 IF ((DX.EQ.0).AND.(DY.GT.0)) FI=PI/2
 IF ((DX.EQ.0).AND.(DY.LT.0)) FI=-PI/2
 IF ((DX.GT.0).AND.(DY.EQ.0)) FI=0.0
 IF ((DX.LT.0).AND.(DY.EQ.0)) FI=PI
 IF ((DX.LT.0).AND.(DY.LT.0)) FI=FI+PI
 IF ((DX.LT.0).AND.(DY.GT.0)) FI=FI+PI
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CFUNCTION THAT RETURNS THE PHASE OF AN IMAGINARY FIELD
REAL FUNCTION PHASE(VAL)
COMPLEX VAL
C
IMPLICIT NONE
REAL DX,DY,FI,PI,ANG
C
PI=ACOS(-1.0)
DX=REAL(VAL)
DY=AIMAG(VAL)
FI=ATAN(DY/DX)
IF (DX.EQ.0) FI=0.0
IF ((DX.EQ.0).AND.(DY.GT.0)) FI=PI/2
IF ((DX.EQ.0).AND.(DY.LT.0)) FI=-PI/2
IF ((DX.GT.0).AND.(DY.EQ.0)) FI=0.0
IF ((DX.LT.0).AND.(DY.EQ.0)) FI=PI
IF ((DX.LT.0).AND.(DY.LT.0)) FI=FI+PI
IF ((DX.LT.0).AND.(DY.GT.0)) FI=FI+PI
ANG=FI*180.0/PI
IF (ANG.GT.180.0) ANG=ANG-360.0
IF (ANG.LT.-180.0) ANG=ANG+360.0
PHASE=ANG
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CSUBROUTINE THAT CONVERTS AN IMAGINARY FIELD TO POLAR FORM
SUBROUTINE CONVTOMAGFAZ(EX)
COMPLEX EX
C
IMPLICIT NONE
REAL RVAL,IVAL,PHASE
C
RVAL=CABS(EX)
IVAL=PHASE(EX)
EX=CMPLX(RVAL,IVAL)
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CFUNCTION THAT COMPUTES THAT CONTROLS IF A VALUE IS NEEDED FOR X-COMPONENT
COMPLEX FUNCTION DIRICHVALXFL(X,Y,Z,FLG,FREQ)
REAL X,Y,Z,FREQ
INTEGER FLG
C
IMPLICIT NONE
COMPLEX DIRICHLETVALX
IF (FLG.EQ.999) THEN
  DIRICHVALXFL=CMPLX(999.0,0.0)
ELSE
  DIRICHVALXFL=DIRICHLETVALX(X,Y,Z,FREQ)
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CFUNCTION THAT COMPUTES THAT CONTROLS IF A VALUE IS NEEDED FOR Y-COMPONENT
COMPLEX FUNCTION DIRICHVALYFL(X,Y,Z,FLG,FREQ)
REAL X,Y,Z,FREQ
INTEGER FLG
C
IMPLICIT NONE
COMPLEX DIRICHLETVALY
IF (FLG.EQ.999) THEN
  DIRICHVALYFL=CMPLX(999.0,0.0)
ELSE
  DIRICHVALYFL=DIRICHLETVALY(X,Y,Z,FREQ)
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CFUNCTION THAT COMPUTES THAT CONTROLS IF A VALUE IS NEEDED FOR Z-COMPONENT
COMPLEX FUNCTION DIRICHVALZFL(X,Y,Z,FLG,FREQ)
```

```
REAL X,Y,Z,FREQ
INTEGER FLG
C
IMPLICIT NONE
COMPLEX DIRICHLETVALZ
IF (FLG.EQ.999) THEN
  DIRICHVALZFL=CMPLX(999.0,0.0)
ELSE
  DIRICHVALZFL=DIRICHLETVALZ(X,Y,Z,FREQ)
END IF
END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

Bibliography

[1] B.H. McDonald and A. Wexler, "Finite element solution of unbounded field problems", *IEEE Trans. Microwave Theory and Techniques* , Vol. 20, No. 12, Dec. 72 pp. 841-847.

[2] J.P. Webb, G.L. Maile and R.L. Ferrari, "Finite element solution of three-dimensional electromagnetic problem" *IEE Proceedings* , Vol. 130, Pt.H., No. 2, March 83.

[3] J.A.M. Svedin, "A numerically efficient finite-element formulation for the general wavegide problem without spurious modes", *IEEE Trans. Microwave Theory and Techniques* , Vol. 37, No. 11, Nov. 89.

[4] J.P. Webb, "Efficient generation of divergence-free fields for the finite element analysis of 3D cavity resonances", *IEEE Trans. on Magnetics*, Vol. 24, No. 1, Jan. 88.

[5] B.E. MacNeal, L.A. Larkin, J.R. Brauer and A.O. Cifuentes, "Elimination of finite element spurious modes using a modal transformation technique", *IEEE Trans. Magnetics*, Vol. 26, No. 5, Sept. 90.

[6] A. Konrad, "A direct three dimensional finite element method for the solution of electromagnetic fields in cavities", *COMPUMAG 85, IEEE Trans. Magnetics*, Vol. 21, No. 6, pp. 2276-2279, Nov. 85.

[7] G. Mur and A.T. de Hoop, "A finite element method for computing three dimensional electromagnetic fields in homogeneous media", *IEEE Trans. Magnetics*, Vol. 21, No. 6, Nov. 85.

[8] J.C. Nedelec, "Mixed finite elements in R3", *Numer. Math.*, Vol. 35, pp. 315-341, 1980.

[9] D.R. Lynch and K.D. Paulsen, "Origin of vector parasites in numerical Maxwell solutions", *IEEE Trans. Microwave Theory and Techniques*, Vol. 39, No. 3, March 91.

[10] K.D. Paulsen and D.R. Lynch, "Elimination of vector parasites in finite element Maxwell solutions", *IEEE Trans. Microwave Theory and Techniques*, Vol. 39, No. 3, March 91.

[11] A.F. Peterson and R.J. Baca, "Error in the finite element discretization of the scalar Helmholtz equation over electrically large regions," *IEEE Microwave and Guided Wave Letters*, Vol. 1, No. 8, Aug. 1991.

[12] J.M. Zhou, K.R. Shao and J.D. Lavers, "Automatic mesh generation allowing for efficient a priori and a posteriori control of the number and distribution of elements," *IEEE Trans. Magnetics*, Vol. 25, No. 4, July 89.

[13] J.M. Zhou, K.R. Shao, K.D. Zhan and Q.H. Zhan, "Computing constrained triangulation and delaunay triangulation: A new algorithm," *IEEE Trans. Magnetics*, Vol. 26, No. 2, March 90.

[14] J.M. Nelson, "A triangulation algorithm for arbitrary planar domains," *Appl. Math. Modelling*, Vol. 2, Sept 78.

[15] P.P. Sylvester and R.L. Ferrari, *Finite Elements for Electrical Engineers*, 2nd Ed., Cambridge University Press, New York, 1990.

[16] C.H. Wilcox, "An expansion theorem for electromagnetic fields," *Comm. Pure and Appl. Math.*, Vol. 9, pp. 115-134, 1956.

[17] A. Sommerfeld, *Partial Differential Equations in Physics*, Academic Press, New York, 1949.

[18] J.J. Stoker, "On radiation conditions," *Comm. Pure and Appl. Math.*, Vol. 9, pp. 577-595, 1956.

[19] A. Bayliss and E. Turkel, "Radiation boundary condition for wave-like equations," *Comm. Pure and Appl. Math.*, Vol. 33, pp. 707-725, 1980.

[20] F.X. Canning, "On the application of some radiation boundary conditions," *IEEE Trans. Antennas and Propagation*, Vol. 38, No. 5, May 1990.

[21] A. Khebir, A.B. Kouki and R. Mittra, "A higher order boundary condition for finite element mesh truncation in open region scattering problems," *IEEE AP-S Symposium Digest*, pp. 1260-1263, Dallas, Texas 1990.

[22] J. D'Angelo and I.D Mayergoyz, "On the use of local absorbing conditions for RF scattering problems," *IEEE Trans. Magnetics*, Vol. 25, No. 4, July 89.

[23] O.M. Ramahi, A. Khebir and R. Mittra, "Numerically derived absorbing boundary condition for the solution of open region scattering problems," *IEEE Trans. Antennas and Propagation*, Vol. 39, No. 3, March 91.

[24] J.B. Keller and D. Givoli, "Exact non-reflecting boundary conditions," *Journal of Computational Physics*, Vol. 82, pp. 172-192, 1989.

[25] J. Sroka, H. Baggentos and R. Ballisti, "On the coupling of the generalized multipole technique with the finite element method," *IEEE Trans. Magnetics*, Vol. 26, No.2, March 90.

[26] Y. Xingchao, D.R Lynch and J.W. Strohbehn, "Coupling of finite element method and moment methods for electromagnetic scattering from inhomogeneous objects," *IEEE Trans. Antennas and Propagation*, Vol. 38, No. 3, March 90.

[27] K.D. Paulsen, D.R. Lynch and J.W. Strohbehn, "Three dimensional finite, boundary, and hybrid element solutions of the Maxwell equations for lossy dielectric media," *IEEE Trans. Microwave Theory and Techniques*, Vol. 36, No. 4, April 88.

[28] A.J. Poggio and E.K. Miller, "Integral equation solutions to three dimensional scattering problems," in Mittra R., ed., *Computer Techniques for Electromagnetics*, Pergamon Press, Oxford, 1973.

[29] J. Brown and E.V. Jull, "The prediction of aerial radiation patterns from near-field measurements," *IEE*, Paper No. 3649 E, Nov 1961.

[30] R.C. Johnson, H. Allen Ecker and J.S. Hollis, "Determination of far-field antenna patterns from near-field measurements," *IEEE Proceedings*, Vol. 61, No. 12, Dec.73.

[31] R.J. Luebbers, K.S. Kunz, M. Scheinder and F. Hunsberger, "A finite-difference time-domain near zone to far zone transformation," *IEEE Trans. Antennas and Propagation*, Vol. 39, No. 4, April 91.

[32] T.K. Sarkar, S. Ponnapolli and E. Arvas, "An accurate method of computing far-field antenna pattern from near-field measurements," *Antennas and Propagation Symposium Digest* Vol. 1, Dallas, Texas, 1990.

[33] O.M. Bucci, C. Gennarelli and C. Savarese, "Fast and accurate near-field-far-field transformation by sampling interpolation of plane-polar measurements," *IEEE Trans. Antennas and Propagation*, Vol. 39, No. 1, Jan. 91.

[34] W.L. Stutzman and G.A. Thiele, *Antenna Theory and Designs*, John Wiley & Sons, New York, 1981.

[35] K.E. Gustafson, *Introduction to Partial Differential Equations and Hilbert Space Methods*, 2nd ed.,John Wiley & Sons, New York, 1987.

[36] H. Anton, *Elementary Linear Algebra*, 4th ed., John Wiley & Sons, New York, 1984.

[37] W. Fleming, *Functions of Several Variables*, 2nd ed., Springer-Verlag, N.Y., 1977.

[38] M.K. Seager and A. Greenbaum, *The Sparse Linear Algebra Package (SLAP) Version 2.0*, Lawrence Livermore National Laboratory, Cont. W-7405-Eng-48.