SOLUTION OF DIELECTRIC LOADED WAVEGUIDES BY FINITE ELEMENT METHODS

Electrical Zoltan J. Csendes, B. Eng. (Mich.) M. Eng.

ABSTRACT

The problem of determining the electromagnetic fields in a dielectric loaded waveguide is examined in this thesis. It is found that these fields are governed by the vector Helmholtz equation formed from the axial components of the electric and magnetic fields, subject to non-homogeneous boundary conditions. A variational expression is derived for this system and its solution is sought by the use of the Rayleigh-Ritz procedure. The region of integration is divided into triangular sub-elements over each of which the trial functions are assumed to have the same form. The first type of polynomial trial function used are the general high-order interpolation polynomials and the corresponding finite element matrices are evaluated. The second type uses the cutoff modes obtained by finite element analysis as the trial functions. The analysis is implemented by a general computer program and dispersion curves and field plots of several waveguide configurations are presented. Solutions obtained by this program are believed to be the most efficient and accurate presently available for arbitrary dielectric loaded waveguides.

SOLUTION OF DIELECTRIC LOADED WAVEGUIDES

Zoltan J. Csendes, B. Eng. (Mich.)

SOLUTION OF DIELECTRIC LOADED WAVEGUIDES BY FINITE ELEMENT METHODS

by

Zoltan J. Csendes, B. Eng.

Department of Electrical Engineering

McGill University,

Montreal, Quebec.

SOLUTION OF DIELECTRIC LOADED WAVEGUIDES

BY FINITE ELEMENT METHODS

by

Zoltan J. Csendes, B. Eng.

A thesis submitted to the Faculty of Graduate Studies
and Research in partial fulfillment of the requirements
for the degree of Master of Engineering.

Department of Electrical Engineering,
McGill University,
Montreal, Quebec,

March, 1970.

ABSTRACT

The problem of determining the electromagnetic fields in a dielectric loaded waveguide is examined in this thesis. It is found that these fields are governed by the vector Helmholtz equation formed from the axial components of the electric and magnetic fields, subject to non-homogeneous boundary conditions. A variational expression is derived for this system and its solution is sought by the use of the Rayleigh-Ritz procedure. The region of integration is divided into triangular sub-elements over each of which the trial functions are assumed to have the same form. The first type of polynomial trial function used are the general high-order interpolation polynomials and the corresponding finite element matrices are evaluated. The second type uses the cutoff modes obtained by finite element analysis as the trial functions. The analysis is implemented by a general computer program and dispersion curves and field plots of several waveguide configurations are presented. Solutions obtained by this program are believed to be the most efficient and accurate presently available for arbitrary dielectric loaded waveguides.

ACKNOWLEDGEMENTS

The author would like to express his most sincere thanks for the inspiration and encouragement provided by his research supervisor Dr. P. Silvester. His extraordinary perception of the underlying nature of problems and his driving enthusiasm for their solution have made the course of this work an exciting and rewarding experience and his friendly manner has made it a pleasant one as well.

Many thanks are also due Mr. M.V.K. Chari for discussions on the finite element method and related topics and to the entire Electrical Engineering Department for providing a lively and cheerful place to work.

Financial support was provided by the Defence Research Board of Canada and the National Research Council of Canada.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 The Nature of the Problem

In a dielectric slab loaded waveguide, there are, enclosed in a cylindrical metal tube, several homogeneous regions of different permittivity. In each of these homogeneous regions, the electric and magnetic fields satisfy the homogeneous Helmholtz equation, subject to boundary conditions that ensure that Maxwell's equations are satisfied on the boundaries. Consequently, the electro magnetic behavior of this system may be determined by solving the boundary value problem formed by the combination of these regions.

All existing solutions of this problem may be classified into two types. Historically, the first class of solutions are those which attempt to determine analytic expressions to satisfy the requirements posed by the boundary value problem. Although such analytic methods have the advantage of being exact, they are restricted to simple geometrical configurations and do not readily yield numerical values. The second class of solution methods attempts to circumvent these limitations by obtaining a sequence of simple numerical functions that converge to the solution. However, at the present time, all such numerical methods have at least one of the following shortcomings:

(1) The sequence of approximate solutions converges too slowly. Hence excessive numerical computations prohibit accurate solution.

(2) The method cannot be adapted to any particular geometry or requires extensive algebraic calculation for each particular problem.

In this thesis, a numerical method is presented that does not suffer from these deficiencies. First, in this chapter, the existing solution methods of the problem are reviewed. In Chapter 2, the variational formulation of the problem is developed and the general finite element method applied to it. Next, two distinct methods are developed, one using interpolation polynomials directly, the other employing the waveguide cutoff modes as trial functions. The latter will be shown to be an extremely efficient and accurate method for the analysis of dielectric loaded waveguides of arbitrary shape. Finally, the computer programs needed for such analysis are presented in the appendix.

1.2 Analytic Solution Methods

The basis of analytic solution methods is to find a coordinate system in which it is possible to separate the independent variables. Then, with a clever choice of the field quantities, it is sometimes possible to cast the equations in a form from which the solution may be recognized. As a result, only the rectangular waveguide with parallel slabs and circular waveguides with concentric dielectric regions have yielded analytic solutions [1,2]. These solutions are not in an accommodating form, however, because difficult transcendental equations must be solved in order to obtain a value for the propagation constant. For some particular geometric and dielectric configurations, the results of such calculations are tabulated or presented graphically [2] - [6] . For the case of an arbitrary off center E-plane dielectrically loaded waveguide, Eberhardt has also developed a nomogram to solve the transcendental equation of the non-hybrid TE modes by using a graphical procedure [6].

1.3 Numerical Solution Methods

Numerical solution methods have become increasingly popular in recent years because, with the advent of modern computers, numerical calculations

have become very easy to perform. In all of these methods, the essential characteristic is the formation of a matrix eigenvalue equation that is solved for the frequency, propagation constant and fields in the waveguide. The most generally successful of these methods, both with respect to accuracy and problem adaptability, are those which employ variational principles.

The first general method for the solution of dielectric loaded waveguides was reported by Collins and Daly [7]. They used the finite difference method to discretize the Helmholtz equation for inhomogeneous waveguides and tried to obtain the waveguide propagation constant by solving a matrix eigenvalue equation of the form $L\psi = \gamma M\psi$. Subsequently Hannaford [8] refined the method by determining the dominant eigenvalue using the stationary property of the Rayleigh quotient $\gamma = \frac{\langle\psi|L\psi\rangle}{\langle\psi|M\psi\rangle}$. Other workers have used essentially the same techniques to obtain results for dielectric loaded waveguides containing a microstrip conductor [9,10]. However, from a practical point of view, the finite difference method is inefficient and unwieldy when applied to dielectric loaded waveguides. The reason for this is that due to the discontinuous change in dielectric constant, a small mesh size must be used to get acceptable results and the corresponding matrix eigenvalue equation is prohibitively large. Furthermore, when the matrix L is not positive definite, the successive over-relaxation technique used to solve the matrix eigenvalue equation will not converge in general and very slow solution methods must be used.

Another procedure used to obtain numerical solutions of inhomogeneous waveguides is the use of the Rayleigh-Ritz method to minimize variational expressions. This procedure has been theoretically known for many years and several simple cases have been solved by it [1,2,11,12]. Recently, fairly extensive calculations with this method were published by Thomas [13,14]

and by English [15]. The principal part of this work is the application of the trial functions $r^n \begin{Bmatrix} \sin m\theta \\ \cos m\theta \end{Bmatrix}$ to circular waveguides and of the products of the sine and cosine functions to rectangular waveguides. The results indicate that the method produces extremely accurate solutions in those cases where the trial functions match the geometry of the problem. This, however, limits each set of trial functions to a particular geometry and produces an inherent inflexibility in the method because lengthy analytic expressions must be evaluated for each set.

Useful numerical results have also been obtained with the transverse resonance technique. In this method, a complicated waveguide cross-section is divided into sections for which transverse solutions are known. The Fourier components of these solutions are then matched on the interface by minimizing a variational expression. Lavik and Unger [16, 17, 18] have calculated remarkably good values for a rectangular guide with a dielectric insert using this method considering that they used only the first four Fourier components. A serious limitation of this method, other than the obvious geometrical one, is that only half of the eigenvalues are calculated, so that either the dominant mode or the bandwidth of the structure remains unknown.

There are also two published reports on the application of the finite element method to dielectric loaded waveguides in the literature. In one paper, Arlett, Bahrani and Zienkiewicz [19] treat dielectric loaded waveguides in the same manner as homogeneous waveguides. They do not mention that the electric and magnetic fields are uncoupled only at cutoff and that for other values of propagation constant a more complicated problem must be solved.

In the other paper, Ahmed and Daly [20] derive a very restricted form of the finite element method and demonstrate that the method produces very accurate solutions by applying their method to a waveguide half-filled with

dielectric. However, their work is unnecessarily restricted to special geometries by imposing a regular mesh spacing and is limited in computational efficiency by confining their polynomial approximation to first order. Two of the great advantages of the finite element method are the freedom to fit any polygonal shape by choosing arbitrary triangular element shapes and sizes, and the extremely accurate approximations provided by high order polynomials. Both of these advantages are retained in the finite element formulation presented in this thesis.

Chapter 2

MATHEMATICAL DEVELOPMENT OF THE FINITE ELEMENT METHOD

2.1 Variational Principles

Variational principles are extremely useful in the numerical solution methods used for boundary value problems for two reasons. They provide stationary expressions about a solution and consequently produce very good answers from approximations that would otherwise be unsatisfactory. In addition, they furnish simple criteria for establishing how the parameters in any function set should be chosen so that the answer will be the best possible.

For the following analysis, it will be convenient to define as the scalar product of two elements $\langle u|$ and $|v\rangle$ of a complex valued vector function space, a number that has the following four properties [21]

(1) $$\langle u \mid v \rangle = \langle v \mid u \rangle \qquad (2.1a)$$

(2) $$\langle u \mid a v \rangle = a \langle u \mid v \rangle \quad \text{for any constant } a \qquad (2.1b)$$

(3) $$\langle u_1 + u_2 \mid v \rangle = \langle u_1 \mid v \rangle + \langle u_2 \mid v \rangle \qquad (2.1c)$$

(4) $$\langle u \mid u \rangle \geq 0 \qquad (2.1d)$$

where equality holds if and only if $|u\rangle = 0$. Furthermore, define as positive definite an operator A if $\langle u|Au\rangle > 0$ when $|u\rangle \neq 0$ and as Hermitian (or self-adjoint for real operators) an operator if $\langle u|Av\rangle = \langle Au|v\rangle = \langle u|A|v\rangle$. Then the following theorem may be proved [21]

Theorem: Let $$A\,|u\rangle = |f\rangle \qquad (2.2)$$

where A is a linear positive definite, Hermitian operator and $|f\rangle$ is a given element. Then if (2.2) has a solution it is unique, and is the one and only element that maximizes the functional

$$F(v) = \langle f \mid v \rangle + \langle v \mid f \rangle - \langle v \mid A \mid v \rangle \qquad (2.3)$$

Proof: (i) To prove that only one element may satisfy (2.2) suppose that $|u_1\rangle$ and $|u_2\rangle$ are both solutions and let $|Z\rangle = |u_1 - u_2\rangle$. Then $A|Z\rangle = 0$, so that $\langle Z|A|Z\rangle = 0$. Since A is positive definite, this implies that $|Z\rangle = 0$ or that $|u_1\rangle = |u_2\rangle$.

(ii) To prove that the solution of (2.2) maximizes (2.3), let $|v\rangle = |u - h\rangle$. Then

$$F(u) - F(v) = \langle f|h\rangle + \langle h|f\rangle - \langle h|A|u\rangle - \langle u|A|h\rangle + \langle h|A|h\rangle$$
$$= \langle f - Au|h\rangle + \langle h|f - Au\rangle + \langle h|A|h\rangle$$
$$= \langle h|A|h\rangle$$

Consequently, if $|h\rangle > 0$ then $\langle h|A|h\rangle > 0$ and $F(u) > F(v)$.

(iii) The converse property of the theorem is that if $|u\rangle$ makes $F(v)$ a maximum, then $|u\rangle$ is a solution of (2.2). This means that for any real parameter ϵ and any fixed element $|\eta\rangle$, $F(v) = F(u + \epsilon\eta)$ is stationary:

$$0 = \left[\frac{d}{d\epsilon} F(u + \epsilon\eta)\right]_{\epsilon=0} \qquad (2.4)$$
$$= \langle f|\eta\rangle + \langle \eta|f\rangle - \langle \eta|A|u\rangle - \langle u|A|\eta\rangle$$
$$= \langle f - Au|\eta\rangle + \langle \eta|f - Au\rangle$$

The same procedure with the element $|j\eta\rangle$ yields

$$0 = \langle f - Au|\eta\rangle - \langle \eta|f - Au\rangle$$

Adding this to the previous equation, there results $0 = \langle f - Au|\eta\rangle$. Since $|\eta\rangle$ is arbitrary, it follows from the fourth property of the scalar product that

$$0 = |f - Au\rangle$$

Similar theorems may be derived for general operators [21], although it is not possible to guarantee that (2.2) has a unique solution. In particular, if the condition that A be positive definite is removed, the following theorem holds [21].

Theorem: If A is a Hermitian operator, the one and only element which makes the functional (2.3) stationary is the function $|v\rangle = |u\rangle$ where $|u\rangle$ is the solution of (2.2).

Proof: The proof that if $F(u)$ is stationary then $|u\rangle$ is the solution of (2.1) is identical to part (iii) of the previous theorem. The rest of the theorem follows from the definition of the functional (2.3) and the properties of the scalar product:

$$\left[\frac{d}{d\epsilon} F(u+\epsilon\eta)\right]_{\epsilon=0} = \langle f|\eta\rangle + \langle\eta|f\rangle - \langle u|A|\eta\rangle - \langle\eta|A|u\rangle$$

$$= \langle f - Au|\eta\rangle + \langle\eta|f - Au\rangle$$

$$= \langle 0|\eta\rangle + \langle\eta|0\rangle$$

$$= 0$$

In order to solve the variational expression (2.3) approximately for the solution of (2.2) the Rayleigh-Ritz procedure may be used. Consider trial functions of the form

$$|v\rangle = \sum_{i=1}^{n} c_i |v_i\rangle \tag{2.5}$$

where the $\{|v_i\rangle\}$ are any set of linearly independent elements of the function space and the $\{c_i\}$ are real numbers. Then (2.3) will be stationary if the expressions $\{\frac{\partial F}{\partial c_i}\}$ are set equal to zero. In terms of (2.5), the functional is

$$F(v) = \sum_i c_i \langle f|v_i\rangle + \sum_i c_i \langle v_i|f\rangle - \sum_{ij} c_i c_j \langle v_i|A|v_j\rangle$$

Therefore

$$\frac{\partial F}{\partial c_k} = \langle f|v_k\rangle + \langle v_k|f\rangle - \sum_i c_i \langle v_i|A|v_k\rangle - \sum_j c_j \langle v_k|A|v_j\rangle$$

$$= 2\,\mathrm{Re}\left\{\langle v_k|f\rangle - \sum_i c_i \langle v_k|A|v_i\rangle\right\} = 0$$

Consequently the Rayleigh-Ritz equations are obtained

$$\langle v_k|f\rangle = \sum_{i=1}^{n} c_i \langle v_k|A|v_i\rangle \qquad , k = 1, \ldots, n \tag{2.6}$$

If the $|v_i\rangle$ are chosen so that $\langle v_i|A|v_k\rangle = \delta_{ik}$, the Rayleigh-Ritz equations simplify to

$$\langle v_k | f\rangle = c_k \qquad , k = 1,\ldots, n \tag{2.7}$$

and the solution of (2.2) is

$$|u\rangle = \sum_{i=1}^{n} c_i |v_i\rangle = \sum_{i=1}^{n} |v_i\rangle\langle v_i | f\rangle \tag{2.8}$$

Since any function may be expanded as

$$|u\rangle = \sum_{i=1}^{\infty} |v_i\rangle\langle v_i|A|u\rangle = \sum_{i=1}^{\infty} |v_i\rangle\langle v_i | f\rangle \tag{2.9}$$

if the $\{|v_i\rangle\}$ form a complete set, it can be seen that the Rayleigh-Ritz equations merely approximate the first n terms of the exact solution.

In fact, it is not difficult to show that for a Hermitian operator, the Rayleigh-Ritz procedure produces the best approximation in a least squares sense, i.e., so that

$$\left\| \sum_{i=1}^{n} c_i A|v_i\rangle - |f\rangle \right\|^2 \tag{2.10}$$

is a minimum. This norm will be a minimum when the $\{c_i\}$ are determined by projecting $|f\rangle$ on to the subspace spanned by the vectors $\{A|v_i\rangle\}$. Let P be this projection operator. Then the projected element, $P|f\rangle$, may be expanded in the $\{A|v_i\rangle\}$:

$$P|f\rangle = \sum_{i=1}^{n} c_i A|v_i\rangle \tag{2.11}$$

Taking the scalar product of both sides with $\langle v_k|$, this becomes

$$\langle v_k|P|f\rangle = \langle v_k|f\rangle = \sum_{i=1}^{n} c_i \langle v_k|A|v_i\rangle \tag{2.12}$$

which are the Rayleigh-Ritz equations for (2.2). Therefore, minimizing the functional (2.3) by the Rayleigh-Ritz procedure yields the best possible solution to (2.2) using the vectors $\{|v_i\rangle\}$.

2.2 Boundary Conditions and Functional Formulation

In order to derive a variational expression for axially uniform, dielectric loaded waveguides, the behavior of their physically realizable electromagnetic fields must be examined. These fields are governed by Maxwell's equations and are assumed to vary sinusoidally in time and in the direction of propagation

$$\vec{H} = \vec{H}(x,y)\, e^{j\omega t + j\beta z} \tag{2.13a}$$

$$\vec{E} = \vec{E}(x,y)\, e^{j\omega t + j\beta z} \tag{2.13b}$$

With these simplifications, the source free field equations are [22, 23]

$$\frac{\partial E_z}{\partial y} = j\beta E_y - j\omega\mu H_x \tag{2.14a}$$

$$\frac{\partial E_z}{\partial x} = j\beta E_x + j\omega\mu H_y \tag{2.14b}$$

$$j\omega\mu H_z = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \tag{2.14c}$$

$$\frac{\partial H_z}{\partial y} = j\beta H_y + j\omega\epsilon E_x \tag{2.14d}$$

$$\frac{\partial H_z}{\partial x} = j\beta H_x - j\omega\epsilon E_y \tag{2.14e}$$

$$j\omega\epsilon E_z = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{2.14f}$$

From (2.14a) and (2.14e), it is evident that if E_z and H_z are completely known, then E_y and H_x can be deduced and from (2.14b) and (2.14d) that E_x and H_y are determined. Hence, the determination of the electromagnetic

field reduces essentially to finding the two quantities E_z and H_z. As a consequence of this, a variational formula containing all six field components, such as Berk's [24], requires three times the necessary calculation and the three component formula used by English [15], while much more efficient, still requires excessive computation.

If the transverse field components are eliminated from (2.14), it is found that in a homogeneous region the axial components satisfy the homogeneous Helmholtz equation

$$(\nabla^2 + k^2)\psi = 0 \tag{2.15}$$

Where

$$\psi = \begin{pmatrix} E_z \\ H_z \end{pmatrix} \tag{2.16}$$

$$k^2 = \omega^2 \epsilon \mu - \beta^2 \tag{2.17}$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{2.18}$$

In order to derive an equation that may be used at a dielectric interface, let τ and z be orthogonal directions tangent to the interface and let n be normal to them.

Then, from (2.14 a,b,d,e)

$$\frac{\mu}{k^2}\frac{\partial H_z}{\partial n} = -\frac{j\omega\epsilon\mu}{k^2} E_\tau + \frac{j\beta\mu}{k^2} H_n \tag{2.19a}$$

$$\frac{\beta}{\omega k^2}\frac{\partial E_z}{\partial \tau} = \frac{j\beta^2}{\omega k^2} E_\tau - \frac{j\beta\mu}{k^2} H_n \tag{2.19b}$$

Upon adding this becomes

$$\frac{\mu}{k^2}\frac{\partial H_z}{\partial n} + \frac{\beta}{\omega k^2}\frac{\partial E_z}{\partial \tau} = -\frac{j}{\omega} E_\tau \tag{2.20a}$$

Similarly

$$\frac{\epsilon}{k^2}\frac{\partial E_z}{\partial n} - \frac{\beta}{\omega k^2}\frac{\partial H_z}{\partial \jmath} = \frac{j}{\omega} H_\jmath \tag{2.20b}$$

These two equations may be put in the concise form [25]

$$\frac{1}{k^2} M \frac{\partial \psi}{\partial n} = \frac{\beta}{\omega k^2} J \frac{\partial \psi}{\partial \jmath} + \frac{j}{\omega} J \xi \tag{2.21}$$

Where

$$M = \begin{pmatrix} \epsilon & 0 \\ 0 & \mu \end{pmatrix} \tag{2.22}$$

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{2.23}$$

$$\xi = \begin{pmatrix} E_\jmath \\ H_\jmath \end{pmatrix} \tag{2.24}$$

The importance of this equation is that ξ and $\frac{\partial \psi}{\partial \jmath}$ are continuous across the boundary even if ϵ and μ change discontinuously there. (That the latter is continuous may be seen from (2.19b) if one recalls that $E_{\jmath 1} = E_{\jmath 2}$ and $\mu_1 H_{n1} = \mu_2 H_{n2}$). It may also be observed from this equation that on a metal boundary where $E_\jmath$ and E_z are zero, $\frac{\partial H_z}{\partial n}$ is zero.

In order to be able to use (2.21), define

$$D = M\left(1 + \frac{1}{k^2}\nabla^2\right) \tag{2.25}$$

and write the wave equations for a homogeneous region t as

$$D\,|\psi\rangle = 0 \tag{2.26}$$

The operator D is self-adjoint and the functional

$$F(\phi) = \langle\phi|D|\phi\rangle = \langle\phi|M|\phi\rangle + \frac{1}{k^2}\langle\phi|M\nabla^2|\phi\rangle \tag{2.27}$$

is stationary if and only if $|\phi\rangle = \begin{pmatrix}\phi_e \\ \phi_h\end{pmatrix} = |\psi\rangle$. Using a suitable integral

definition for the scalar product, the functional at the stationary value may be written as

$$F(\psi) = \int_t \psi^T M \psi \, dt + \frac{1}{k^2} \int_t \psi^T M \nabla^2 \psi \, dt \tag{2.28}$$

Green's first theorem states that [26]

$$\int_t \psi^T M \nabla^2 \psi \, dt = -\int_t \nabla \psi^T M \cdot \nabla \psi \, dt + \oint_\gamma \psi^T M \frac{\partial \psi}{\partial n} \, dt \tag{2.29}$$

and the functional for the union of many homogeneous regions will be the sum of the contributions from each

$$F_R(\psi) = \sum_t F_t \tag{2.30}$$

Therefore the functional for an dielectric loaded waveguide is

$$F_R(\psi) = \int_R \psi^T M \psi \, dR - \int_R \frac{1}{k_t^2} \nabla \psi^T M \cdot \nabla \psi \, dR + \oint_\Gamma \frac{1}{k_t^2} \psi^T M \frac{\partial \psi}{\partial n} \, d\Gamma \tag{2.31}$$

where the integration is over the whole waveguide cross-section R and around all of the boundaries Γ of each subregion. Using (2.21) on the boundary integral gives

$$\int_\Gamma \frac{1}{k_t^2} \psi^T M \frac{\partial \psi}{\partial n} \, d\Gamma = \frac{\beta}{\omega} \oint_\Gamma \psi^T J \frac{\partial \psi}{\partial \gamma} \, d\Gamma + \frac{\lambda}{\omega} \oint_\Gamma \psi^T J \xi \, d\Gamma \tag{2.32}$$

Consider the second integral on the right side of this equation. If external boundaries are either metal, where E_z and E_γ vanish, or lines of symmetry, where H_z and H_γ vanish, then this boundary integral is zero around the external walls. Furthermore, it can be seen from Figure 2.1 that all of the internal boundaries are to be travelled twice, in opposite directions. Since the values of ψ and ξ must be continuous, there is no contribution to the functional from this integral. The remaining boundary integral is

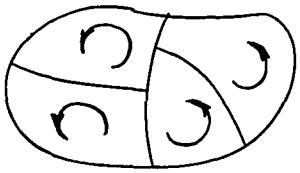


Figure 2.1

$$\frac{\beta}{\omega}\oint_{\Gamma}\frac{1}{k_t^2}\psi^T J\frac{\partial\psi}{\partial\tau}\,d\Gamma=\frac{\beta}{\omega}\sum_t\frac{1}{k_t^2}\oint_{\tau}\left(E_z\frac{\partial H_z}{\partial\tau}-H_z\frac{\partial E_z}{\partial\tau}\right)d\tau \tag{2.33}$$

This integral contains all of the boundary conditions for dielectric loaded waveguides. Notice that this integral is zero on external boundaries and on internal boundaries across which k_t does not change. At cutoff, i.e. when $\beta = 0$, the boundary contribution to the functional is again zero and when it is zero, the electric and magentic fields are uncoupled.

This line integral may be converted into a surface integral by using the vector identity. [26]

$$\oint_{\tau}(u\nabla v)\cdot d\vec{\tau}=\int_t(\nabla u\times\nabla v)\cdot d\vec{t} \tag{2.34}$$

Then, in the formal scalar product notation of (2.27), the functional expression for dielectric loaded waveguides may be written as

$$F_R(\phi)=\langle\phi|M|\phi\rangle+\langle\nabla\phi|\frac{\beta}{\omega k_t^2}JC_z-\frac{1}{k_t^2}M|\nabla\phi\rangle \tag{2.35}$$

where C_z is the operator producing the z component of the cross product. In integral notation, this equation reads

$$F_R\begin{pmatrix}\phi_e\\ \phi_h\end{pmatrix}=\sum_t\Big\{\epsilon_t\int_t\phi_e^2\,dt+\mu_t\int_t\phi_h^2\,dt-\frac{\epsilon_t}{k_t^2}\int_t\nabla\phi_e\cdot\nabla\phi_h\,dt \tag{2.36}$$

$$-\frac{\mu_t}{k_t^2}\int_t\nabla\phi_h\cdot\nabla\phi_h\,dt+2\frac{\beta}{\omega}\frac{1}{k_t^2}\int_t(\nabla\phi_e\times\nabla\phi_h)\cdot\vec{a}_z\,dt$$

This functional has also been independently derived by Ahmed [27] .

The natural boundary conditions of this functional can be obtained by using its stationary property about the solution to (2.25) and imposing the conditions contained in (2.21) [28, 29] . Although it may not be obvious, such a procedure reverses the steps used to derive the functional and, therefore,

will reproduce the boundary conditions that were built into it. In both cases, the natural boundary conditions that result are homogeneous Neumann for a field quantity, either E_z or H_z, not specified on the boundary.

2.3 The Rayleigh - Ritz Procedure

In order to solve (2.26) by determining the stationary condition of the functional (2.35), the Rayleigh-- Ritz method may be applied. As outlined in section 2.1, the procedure is to write

$$\phi^{(e)} = \sum_{i=1}^{n} \phi_i^{(e)} \alpha_i(x,y) \tag{2.37a}$$

$$\phi^{(h)} = \sum_{i=1}^{n} \phi_i^{(h)} \alpha_i(x,y) \tag{2.37b}$$

and set $\frac{\partial F}{\partial \phi_i^{(e)}}$ and $\frac{\partial F}{\partial \phi_i^{(h)}}$ equal to zero. The first and second terms of (2.36) yield

$$\frac{\partial}{\partial \phi_k^{(e)}} \int \sum_{i,j} \phi_i^{(e)} \phi_j^{(e)} \alpha_i \alpha_j \, dt = 2 \sum_i \phi_i^{(e)} \int \alpha_i \alpha_k \, dt \tag{2.38a}$$

$$\frac{\partial}{\partial \phi_k^{(h)}} \int \sum_{i,j} \phi_i^{(h)} \phi_j^{(h)} \alpha_i \alpha_j \, dt = 2 \sum_i \phi_i^{(h)} \int \alpha_i \alpha_j \, dt \tag{2.38b}$$

and the third and fourth give

$$\frac{\partial}{\partial \phi_k^{(e)}} \int \sum_{i,j} \phi_i^{(e)} \phi_j^{(e)} \nabla \alpha_i \cdot \nabla \alpha_j \, dt = 2 \sum_i \phi_i^{(e)} \int \nabla \alpha_i \cdot \nabla \alpha_k \, dt \tag{2.39a}$$

$$\frac{\partial}{\partial \phi_k^{(h)}} \int \sum_{i,j} \phi_i^{(h)} \phi_j^{(h)} \nabla \alpha_i \cdot \nabla \alpha_j \, dt = 2 \sum_i \phi_i^{(h)} \int \nabla \alpha_i \cdot \nabla \alpha_k \, dt \tag{2.39b}$$

The last term has both derivatives

$$\frac{\partial}{\partial \phi_k^{(e)}} \int \sum_{i,j} \phi_i^{(e)} \phi_j^{(h)} \left(\alpha_i \frac{\partial \alpha_j}{\partial y} - \alpha_j \frac{\partial \alpha_i}{\partial y} \right) dy = \sum_j \phi_j^{(h)} \int \left(\alpha_k \frac{\partial \alpha_j}{\partial y} - \alpha_j \frac{\partial \alpha_k}{\partial y} \right) dy \tag{2.40a}$$

$$\frac{\partial}{\partial \phi_k^{(h)}} \int \sum_{i,j} \phi_i^{(e)} \phi_j^{(h)} \left(\alpha_i \frac{\partial \alpha_j}{\partial y} - \alpha_j \frac{\partial \alpha_i}{\partial y} \right) dy = \sum_i \phi_i^{(e)} \int \left(\alpha_i \frac{\partial \alpha_k}{\partial y} - \alpha_k \frac{\partial \alpha_i}{\partial y} \right) dy \tag{2.40b}$$

Therefore, the Rayleigh - Ritz equations are

$$\frac{\partial F}{\partial \Phi_k^{(e)}} = 0 = \sum_t \left\{ 2\epsilon_t \sum_i T_{ki} \Phi_i^{(e)} - 2 \frac{\epsilon_t}{k_t^2} \sum_i S_{ki} \Phi_i^{(e)} + \frac{\beta}{\omega} \frac{1}{k_t^2} \sum_i U_{ki} \Phi_i^{(h)} \right\} \tag{2.41a}$$

$$\frac{\partial F}{\partial \Phi_k^{(h)}} = 0 = \sum_t \left\{ 2\mu_t \sum_i T_{ki} \Phi_i^{(h)} - 2 \frac{\mu_t}{k_t^2} \sum_i S_{ki} \Phi_i^{(h)} + \frac{\beta}{\omega} \frac{1}{k_t^2} \sum_i U_{ik} \Phi_i^{(e)} \right\} \tag{2.41b}$$

where

$$S_{ki} = \int_t \nabla \alpha_k \cdot \nabla \alpha_i \, dt \tag{2.42}$$

$$T_{ki} = \int_t \alpha_k \alpha_i \, dt \tag{2.43}$$

$$U_{ki} = \oint_\gamma \left(\alpha_k \frac{\partial \alpha_i}{\partial \gamma} - \alpha_i \frac{\partial \alpha_k}{\partial \gamma} \right) d\gamma \tag{2.44}$$

These equations can be cast into matrix form if the following vector and symmetric matrices are defined

$$\mathcal{X} = \begin{pmatrix} \Phi_1^{(e)} \\ \vdots \\ \Phi_n^{(e)} \\ \hline \Phi_1^{(h)} \\ \vdots \\ \Phi_n^{(h)} \end{pmatrix} \tag{2.45}$$

$$\mathcal{U} = \sum_t \frac{k_0^2}{k_t^2} \left(\begin{array}{ccc|ccc} \epsilon_t S_{11} & \cdots & \epsilon_t S_{1n} & -\frac{\delta}{2} U_{11} & \cdots & -\frac{\delta}{2} U_{1n} \\ \vdots & & \vdots & \vdots & & \\ \epsilon S_{n1} & \cdots & \epsilon_t S_{nn} & -\frac{\delta}{2} U_{n1} & \cdots & -\frac{\delta}{2} \\ \hline -\frac{\delta}{2} U_{11} & \cdots & -\frac{\delta}{2} U_{n1} & \mu_t S_{11} & \cdots & \mu_t S_{1n} \\ \vdots & & \vdots & \vdots & & \vdots \\ -\frac{\delta}{2} U_{1n} & \cdots & -\frac{\delta}{2} U_{nn} & \mu_t S_{n1} & \cdots & \mu_t S_{nn} \end{array} \right) \tag{2.46}$$

$$\mathcal{T} = \sum \left(\begin{array}{c|c} \begin{matrix} \epsilon_t T_{11} & \cdots & \epsilon_t T_{1n} \\ \vdots & & \vdots \\ \epsilon_t T_{n1} & \cdots & \epsilon_t T_{nn} \end{matrix} & 0 \\ \hline 0 & \begin{matrix} \mu_t T_{11} & \cdots & \mu_t T_{1n} \\ \vdots & & \vdots \\ \mu_t T_{n1} & \cdots & \mu_t T_{nn} \end{matrix} \end{array} \right) \tag{2.47}$$

Here $k_o^2 = \omega^2 \epsilon_o \mu_o$, $\delta = \frac{\beta}{\omega}$ and $\frac{k_o^2}{k_t^2} = \frac{1}{\epsilon_t \mu_t - \delta^2}$. With these definitions, (2.41) becomes

$$\mathcal{U}\, \mathcal{x} = k_o^2\, \mathcal{T}\, \mathcal{x} \tag{2.48}$$

For any value of phase velocity this is a matrix eigenvalue equation that may be solved for the frequency of propagation and the field distribution in the waveguide. In a homogeneous waveguide, or at cutoff, the off-diagonal matrix sections of $\mathcal{U}$ are identically zero and (2.48) reduces to the equations used in the finite element formulation of homogeneous waveguides [30,31].

$$S\phi = k_o^2 T \phi \qquad , \phi = \begin{pmatrix} \phi_1^{(e)} \\ \vdots \\ \phi_n^{(e)} \end{pmatrix} \text{ or } \phi = \begin{pmatrix} \phi_1^{(h)} \\ \vdots \\ \phi_n^{(h)} \end{pmatrix} \tag{2.49}$$

Therefore, with the Rayleigh - Ritz procedure, the physical coupling of the electric and magnetic fields on the dielectric interface has been converted to the coupling between their corresponding matrices.

2.4 Triangular Finite Elements

The Rayleigh - Ritz expansion that led to the development of (2.48) has been performed with an arbitrary set of trial functions $\{\alpha_i(x,y)\}$, subject only to the conditions that they be linearly independent and that the operations performed on them be defined. However, for each set of trial functions that is chosen, the integrals in (2.42) - (2.44) must be evaluated and, as a practical matter, it is wise to choose trial functions that minimize such calculations.

The basic idea of the finite element method is to split the region of integration into a number of simple elements. The integration over each element of some particular sets of trial functions may then be reduced to the evaluation of a few parameters and the calculation of the total integral may be performed by a simple combination of these parameters.

These fundamental elements should be chosen to possess the following properties:

(1) The number of parameters due to the element shape should be as small as possible.

(2) It should be possible to divide any geometrical region into these elements, at least approximately.

A triangular element of arbitrary shape and size best satisfies these two criteria. Therefore, the trial functions used in finite element analysis are written in the following form

$$\phi^{(e)} = \sum_{t=1}^{N} \sum_{i=1}^{n} \phi_i^{(e)} \alpha_i^t(x,y) \qquad (2.50)$$

where $\alpha_i^t(x,y)$ is defined to be zero if (x,y) is not in triangle t.

In order that trial functions of the above form be meaningful when

applied to a functional, it is necessary that ~~they have continuous~~ the derivatives of order one less than required in the functional they are applied to [32, 33]. ~~be continuous~~ The functional derived for dielectric loaded waveguides requires one differentiation. Hence, the trial functions in (2.50) must be continuous but ~~may~~ possess discontinuous derivatives. In physical terms, discontinuous derivatives that are not infinite along a line do not affect the value of the functional because it is integrated over an area.

In order to facilitate calculations over triangular elements, it is convenient to write (2.42) in terms of homogeneous triangular or trilinear coordinates. They are defined as the ratio of the distance from each side of a triangle to the corresponding altitude [34, 35, 36]. Each may be written in terms of the Cartesian coordinates of the three triangle vertices as

$$\zeta_i = \frac{1}{2A}(a_i + b_i x + c_i y) \tag{2.51}$$

where

$$a_i = x_j y_k - y_j x_k \tag{2.52a}$$

$$b_i = y_j - y_k \tag{2.52b}$$

$$c_i = x_k - x_j \tag{2.52c}$$

$$A = \text{triangle area} = 1/2 \begin{vmatrix} Xi & Yi & 1 \\ Xj & Yj & 1 \\ X_K & Y_K & 1 \end{vmatrix} \tag{2.53}$$

and are related by the relationship

$$\zeta_1 + \zeta_2 + \zeta_3 = 1 \tag{2.54}$$

By changing the variables of differentiation to triangular coordinates, (2.42) becomes [37].

$$S_{pq} = \frac{1}{2A^2} \sum_{i,j=1}^{3} (b_i b_j + c_i c_j) \int_t \frac{\partial \alpha_p}{\partial \zeta_i} \frac{\partial \alpha_q}{\partial \zeta_j} \, dt \tag{2.55}$$

From the properties of a triangle, it is found that

$$b_i b_j + c_i c_j = -2A \cot \theta_k \quad , i \neq j \tag{2.56a}$$

$$b_i^2 + c_i^2 = 2A(\cot \theta_j + \cot \theta_k) \tag{2.56b}$$

where θ_i is the included angle at vertex i. Using these relationships, (2.55) may be written as

$$S_{pq} = \frac{1}{2} \sum_{i=1}^{3} \cot \theta_i \, Q_{pq}^{i} \tag{2.57}$$

where

$$Q_{pq}^{i} = \frac{1}{A} \int \left(\frac{\partial \alpha_p}{\partial \zeta_j} - \frac{\partial \alpha_p}{\partial \zeta_k} \right) \left(\frac{\partial \alpha_q}{\partial \zeta_j} - \frac{\partial \alpha_q}{\partial \zeta_k} \right) dt \tag{2.58}$$

The quantities Q_{pq}^{i}, $\frac{1}{A} T_{pq}$ and U_{pq} are independent of any variable that depends on triangle shape or size. Consequently, the integrations indicated in (2.43), (2.44) and (2.57) need to be performed only once for any set of triangular trial functions. Once they are evaluated, the matrices $\mathcal{U}$ and $\mathcal{S}$ may be assembled by performing simple arithmetic operations for each element and, according to (2.50), summing all of the element values.

Chapter 3

SOLUTION BY INTERPOLATION POLYNOMIALS

3.1 The Conventional Finite Element Method

The family of polynomials is one of the simplest class of functions to employ as approximating functions [38] and were the first to be applied to the finite element method [39, 37] . It has been found that a set of complete polynomials may be obtained from interpolation polynomials and, with such a formulation, many of the finite element calculations simplify.

Polynomial expressions for the potential in an element are required to be complete in x and y if their form is to be rotationally invariant [40, 37] . A complete polynomial of order N contains $n = 1/2\ (N + 1)\ (N + 2)$ terms. As a result, this must be the number of parameters for each element in (2.50). It has been stated that the trial functions should be continuous across element boundaries. In order to ensure this condition, the two Nth order polynomials on either side of an element edge must be made equal on the boundary. This implies that (N + 1) parameters must be specified on each side of a triangular element with the remaining $1/2(N - 1)\ (N - 2)$ parameters specified in the interior. Silvester [37] found that the polynomials

$$\alpha_{ijk}(\zeta_1, \zeta_2, \zeta_3) = P_i(\zeta_1) P_j(\zeta_2) P_k(\zeta_3) \tag{3.1}$$

where

$$P_m(z) = \prod_{i=1}^{m} \left(\frac{Nz - i + 1}{i} \right) \qquad , m \geq 1 \tag{3.2}$$

$$P_0(z) = 1$$

satisfy these requirements. In addition, these polynomials have the very desirable property

$$\alpha_{ijk}\left(\frac{i}{N}, \frac{j}{N}, \frac{k}{N}\right) = 1 \tag{3.3}$$

if i,j,k are integers satisfying

$$i + j + k = N \ , \quad 0 \leq i,j,k \leq N \tag{3.4}$$

and

$$\alpha_{ijk}\left(\frac{l}{N}, \frac{m}{N}, \frac{n}{N}\right) = 0 \tag{3.5}$$

if l,m,n are integers that satisfy (3.4) but have

$$l \neq i \quad \text{or} \quad m \neq j \quad \text{or} \quad n \neq k \tag{3.6}$$

The regularly spaced points defined by the triangular coordinates in (3.3) thus form an interpolation point set. The finite element trial functions (2.5) are completely specified for the polynomials (3.1) by the potential values $\phi^{(e)}_{\ldots ijk}$ on this interpolation point set. As a consequence of this property, the coefficients in the Rayleigh - Ritz expansion which form the vector $\mathfrak{A}$ have the useful physical interpretation of representing the potential values of regularly spaced points on each element. Hence, each trial function is completely specified by a point value, and the terms trial function number and point number may be used interchangeably.

Very nice idea.

Furthermore, a very useful simplification can be made. Ordinarily the use of (2.50) in (2.48) results in a double summation over the parameters in each element. However, with interpolating polynomials only one parameter is non - zero in each trial function and only a single summation remains for each element.

The matrix elements S_{ki}, T_{ki} and U_{ki} may now be evaluated for polynomials approximation by using (3.1) and performing the indicated operations. As all of the terms are polynomials, the calculations require the evaluation of the integrals of monomial expressions over a general triangular area. Expressed in triangular coordinates, these integrals are the Beta functions [41, 42]

$$\frac{B(m+1,\ n+2)}{(n+1)} = \int_0^1 \int_0^{1-\zeta_1} \zeta_1^m \zeta_2^n \, d\zeta_2 \, d\zeta_1 = \frac{m!\ n!}{(m+n+2)!} \tag{3.7}$$

and tabulated values are available [43] .

Moreover, the matrix elements S_{ki} and T_{ki} have been used to solve the homogeneous Helmholtz equation and are therefore known and tabulated [37] for polynomial approximations up to fourth order. In order to evaluate the remaining element, U_{ki} , an expression must be found for the tangential derivatives in (2.44). This may be obtained by finding the derivative of the polynomial (3.2)

$$\frac{\partial P_m}{\partial z}(z) = \sum_{i=1}^{m} \frac{P_m(z)}{\frac{N - i + 1}{N}} \quad , \ m \geq 1 \tag{3.8}$$

$$= 0 \quad , \ m = 0$$

and using the relationship

$$\zeta_p + \zeta_q = 1 \tag{3.9}$$

valid on triangular element edges. After a little algebra, it follows that the derivatives on the edges may be expressed as the polynomials

$$\frac{\partial \alpha_{0;k}}{\partial \zeta_1} = \begin{cases} N\left[\sum_{i=1}^{k} \frac{P_k(\zeta_3) P_{N-k}(1-\zeta_3)}{N\zeta_3 - i + 1} - \sum_{i=1}^{N-k} \frac{P_k(\zeta_3) P_{N-k}(1-\zeta_3)}{N(1-\zeta_3) - i + 1}\right] & , k \neq 0, k \neq N \\ -N\left[\sum_{i=1}^{N} \frac{P_N(1-\zeta_3)}{N(1-\zeta_3) - i + 1}\right] & , k = 0 \\ N\left[\sum_{i=1}^{N} \frac{P_N(\zeta_3)}{N\zeta_3 - i + 1}\right] & , k = N \end{cases} \tag{3.10}$$

The U_{ki} may then be evaluated to any order of polynomial approximation and for any triangular shape by substituting this expression into (2.44) and integrating the result around the perimeter of a triangle, using triangular coordinates. Since the number of arithmetic steps required

increases as $3N^6$, this task appears at first to be quite formidable. Fortunately however, the elements of this matrix are antisymmetric and due to triangular symmetries, behaves as a group modulo three. It is also apparent that the diagonal elements and those between two interpolation points not on the same side must be zero. As a result, the number of different matrix elements for each order of polynomial approximation is reduced to

$$1/4\ (N^2 + 2N + 1) \qquad \text{if N is odd} \qquad (3.11a)$$

$$1/4\ (N^2 + 2N) \qquad \text{if N is even} \qquad (3.11b)$$

The U matrix values have been computed up to fourth order polynomial approximations and are presented in table 3.1 . These matrices have also been independently computed by Daly [44] . Once the matrix element values are computed, the assembly and solution of (2.48) can be easily coded and performed by an automatic digital computer.

3.2 Computer Program and Results

A general computer program has been written to analyze dielectrically loaded waveguides by the conventional finite element method and a listing is given in Appendix 1. The program requires as input only the desired values of phase velocity, the coordinates of the numbered triangle vertices and a list of the vertex numbers for each triangle. It is difficult to ascertain the number of triangles that may be fed in using 100 K bytes of immediate access memory because this number depends on the topology of the system. Generally, the limit is about fifty first order triangles or about five fourth order elements; these estimates may be doubled if only cutoff values are required. Of course, if for some reason more points were required, another partition of cone storage could be allocated by simply increasing

Table 3.1

N = 1, common denominator = 1

0	+1	-1
	0	+1
		0

N = 2, common denominator = 3

0	4	-4	-1	0	1
	0	0	4	0	0
		0	0	0	-4
			0	4	-1
				0	4
					0

N = 3, common denominator = 80

0	57	-57	-24	0	24	7	0	0	-7
	0	0	81	0	0	-24	0	0	0
		0	0	0	-81	0	0	0	24
			0	0	0	57	0	0	0
				0	0	0	0	0	0
					0	0	0	0	-57
						0	57	-24	7
							0	81	-24
								0	57
									0

N = 4, common denominator = 945

0	1472	-1472	-804	0	804	384	0	0	-384	-107	0	0	0	+107
	0	0	2112	0	0	-1024	0	0	0	384	0	0	0	0
		0	0	0	-2112	0	0	0	1024	0	0	0	0	-384
			0	0	0	2112	0	0	0	-804	0	0	0	0
				0	0	0	0	0	0	0	0	0	0	0
					0	0	0	0	-2112	0	0	0	0	804
						0	0	0	0	1472	0	0	0	0
							0	0	0	0	0	0	0	0
								0	0	0	0	0	0	0
									0	0	0	0	0	-1472
										0	1472	-804	384	-107
											0	2112	-1024	384
												0	2112	-804
													0	1472
														0

the dimensions on the first card.

In the subroutine READIN, the computer generates the additional points required for high order polynomial approximation and renumbers all of the points so that Dirichlet points are listed at the upper end of the vector $\mathcal{X}$. The remaining READIN function is the calculation of the four paremeters, the area and the three included angle cotangents, necessary to specify the triangular finite element geometry. Once evaluated, the remaining computer memory may be cleared to allow for the $\mathcal{U}$ and $\mathcal{S}$ matrices.

These matrices are assembled in the subroutine ASSEMB from element values contained in the block data subprogram. Since the U matrix sections are zero at cutoff, in this case provision is made to solve for the electric and magnetic fields separately with a considerable saving of computer time. The only complication in the assembly routine is the need to interchange point numbers on the boundary where the electric field satisfies Dirichlet conditions and the magnetic field is Neumann, or vice versa. This is, however, taken care of fairly simply with the numbering scheme obtained from the READIN routine.

The matrix eigenvalue problem is solved by a procedure that has become fairly standard and, in fact, the routines are more or less borrowed from the routines used to solve the homogeneous waveguide problem [31] . First, since the matrix $\mathcal{S}$ is symmetric and positive definite, it is possible to perform a Choleski decomposition on $\mathcal{S}$ [45,46,47]

$$\mathcal{S} = GG^T \tag{3.12}$$

where G is a nonsingular lower triangular matrix. Hence (2.48) may be written in standard form as

$$A Z = k_0^2 Z \tag{3.13}$$

where

$$A = G^{-1} \nu G^{T^{-1}} \tag{3.14}$$

$$= G^{T} \mathfrak{x} \tag{3.15}$$

The Choleski decomposition is performed by the subroutine CHOLOW and the inversion of G and multiplication $G^{-1} \nu G$ are done by the subroutines INVLOW and GAGT respectively.

The solution of the matrix eigenvalue problem (3.13) is performed by Householder tridiagonalization of the matrix A followed by the method of bisection to locate the eigenvalues [45,46]. These operations are performed by the subroutines TRIDIA and BISLCT. Although the solution algorithms involved are rather complicated, this procedure is the most efficient method presently available for a general symmetric matrix. One useful advantage of this procedure, due to the Sturm sequence property of the determinant of a tridiagonal matrix, is the ability to determine the value of any eigenvalue of the matrix. Since only the positive eigenvalues are desired in (2.48), the program determines the number of non-positive eigenvalues and proceeds, in most cases, to evaluate the next fifteen. If no eigenvalue is positive, a message is printed and the program advances to the next data set.

Next, the eigenvectors of the tridiagonal matrix are determined by Wielandt iteration [45] in the subroutine WIELND. As the eigenvalues are accurately known, a very strong dominance of the proper eigenvector over the others exists, and only two passes are required to achieve full accuracy. The subroutine REVERS then performs the reverse transformation to convert these eigenvectors into the eigenvectors of the full matrix A. Subsequently, TRIMUL computes the eigenvectors $\mathfrak{x}$ by using the matrix $G^{T^{-1}}$ still stored in memory.

The remaining subroutines given in Appendix 1 are needed for the

solution procedure developed in the next chapter and will be discussed there. Although they are incorporated in this program, they do not affect the solution. The entire package has been molded into a unit that requires comparatively little memory and performs with outstanding efficiency and accuracy.

Dispersion curves may thus be obtained by plotting the dominant eigenvalue at different values of phase velocity. Figures 3.1-3.2 contain dispersion curves obtained by finite element analysis for some waveguide configurations that have been solved by using other methods. The values are remarkably good throughout the curves and any difference is attributable to difficulty in reading the dispersion curves from the references.

As with any numerical method, the question "What is the solution accuracy?" must be answered. This is perhaps conceptually most easily accomplished by considering a rectangular waveguide divided into four fourth order triangles. Along the centerline the first electric mode (TM_{11}) is theoretically known to be half of a sine wave. In the finite element analysis, this wave is approximated by the best four fourth order polynomials possible. Clearly the difference between the two curves is extremely small and, due to the variational formulation, the error in the eigenvalue is even less. In fact, if the computations are performed on a computer with 24-bit maubissa, it is not possible to distinguish between the two, as all errors can be traced to round-off error [31] .

The field in a more complicated waveguide configuration is no longer a sine wave but as long as the region contains no re-entrant corners, it is still well-behaved. As an example, the first ten eigenvalues of the waveguide in Figure 3.2 were evaluated twice, once with half the number of triangles in completely different configurations, and their values only

differed only in the seventh significant figure. Therefore, the accuracy of the solutions obtained by finite element analysis of dielectric loaded waveguides is limited by the same process that limits all computer solutions: round-off error accumulation due to a finite word length.

One of the problems encountered in plotting dispersion curves is the selection of the correct eigenvalue. For values of δ less than $1/c$, all eigenvalues are real and the dominant eigenvalue is the first. An exception is the case of pure Neumann boundaries, where the dominant eigenvalue is the second, $\psi = 0$ being a trivial, non-physical solution of the Helmholtz equation in this case.

At $\delta = 1/c$, (2.48) is not defined; this corresponds to the case where $k = 0$ in the Helmholtz equation and the fields are governed by Laplace's equation. For values of δ close to $1/c$ only those regions with $\epsilon_t\mu_t = \epsilon_0\mu_0$ will contribute appreciably to ν but, so long as no overflows are encountered, (2.48) is still valid, and good results are obtained, as indicated by the dispersion curves.

For δ greater than $1/c$ but less than $\epsilon_1\mu_1$ the regions with $\epsilon_0\mu_0$ will produce negative contributions to ν and usually about half of the resulting eigenvalues are negative. The first real eigenvalue then corresponds to the dominant mode, except for some values of δ near $\epsilon_0\mu_0$ where some smaller real eigenvalues appear. These extraneous solutions are troublesome and their properties will be examined below.

At $\delta = \epsilon_1\mu_1$ the same problem occurs as at $\delta = \epsilon_0\mu_0$ and for δ greater than $\epsilon_1\mu_1$, all eigenvalues are negative.

This behavior of the eigenvalues can be explained theoretically by examining the nature of the functional (2.33).

The first term

$$\sum_t \int_t (\epsilon_t E_z^2 + \mu_t H_z^2)\, dt \qquad (3.16)$$

is positive definite and the last

$$\sum_t \frac{1}{k_t^2} \int_t (\epsilon_t |\nabla E_z|^2 + \mu_t |\nabla H_z|^2) \, dt \qquad (3.17)$$

is either positive semi-definite or negative semi-definite in each region, depending on the sign of k_t^2 . The remaining term is a boundary integral that, in physical terms, cannot add energy to the system. Between cutoff and the first singularity, k_t^2 is always positive. Hence, the solution is unique there, except for the case of pure Neumann boundaries where $\psi =$ constant is possible.

Slightly above the first singularity, some values of k_t^2 are still positive but others are negative and the total functional is indefinite. Consequently, in addition to the true solution, extraneous, non-physical solutions are possible. Although they cannot be eliminated mathematically, extraneous solutions may be detected by their non-physical behavior [12] . A plot of the field solution provides a good, if somewhat tedious test, since the dominant field in a physical waveguide has an easily recognizable convexity.

As the value of δ is increased, the magnitudes of the positive contributions increase and the magnitudes of the negative ones decrease. The functional never becomes definitely positive but there are, correspondingly, fewer extraneous solutions.

Beyond the final singularity, all values of k_t^2 are negative and the operator is negative semi-definite. All eigenvalues are negative and free propagation in the waveguide cannot exist.

An interesting mathematical property of the matrix may be ascertained by considering a waveguide in which the direction of propagation is reversed. Physically, the waveguide modes are not changed and so the eigenvalues and eigenvectors of the matrix must be the same. However, this corresponds to

changing the sign of $\delta = \beta/\omega$ and, consequently, of the off-diagonal matrix sections. Hence, the antisymmetric elements U_{ki} may be assembled with the incorrect sign without affecting the properties of the matrix.

Finally, Figure 3.3 contains an example of the solution of a problem that has important practical applications [9,10] . The configuration is a microstrip conductor on a dielectric surface and surrounded by a zero potential surface.

Figure 3.1: Normalized phase-change coefficient as a function of angular frequency for a half-filled rectangular waveguide.
$\epsilon_r = 2.45$
— Harrington [23] o finite element

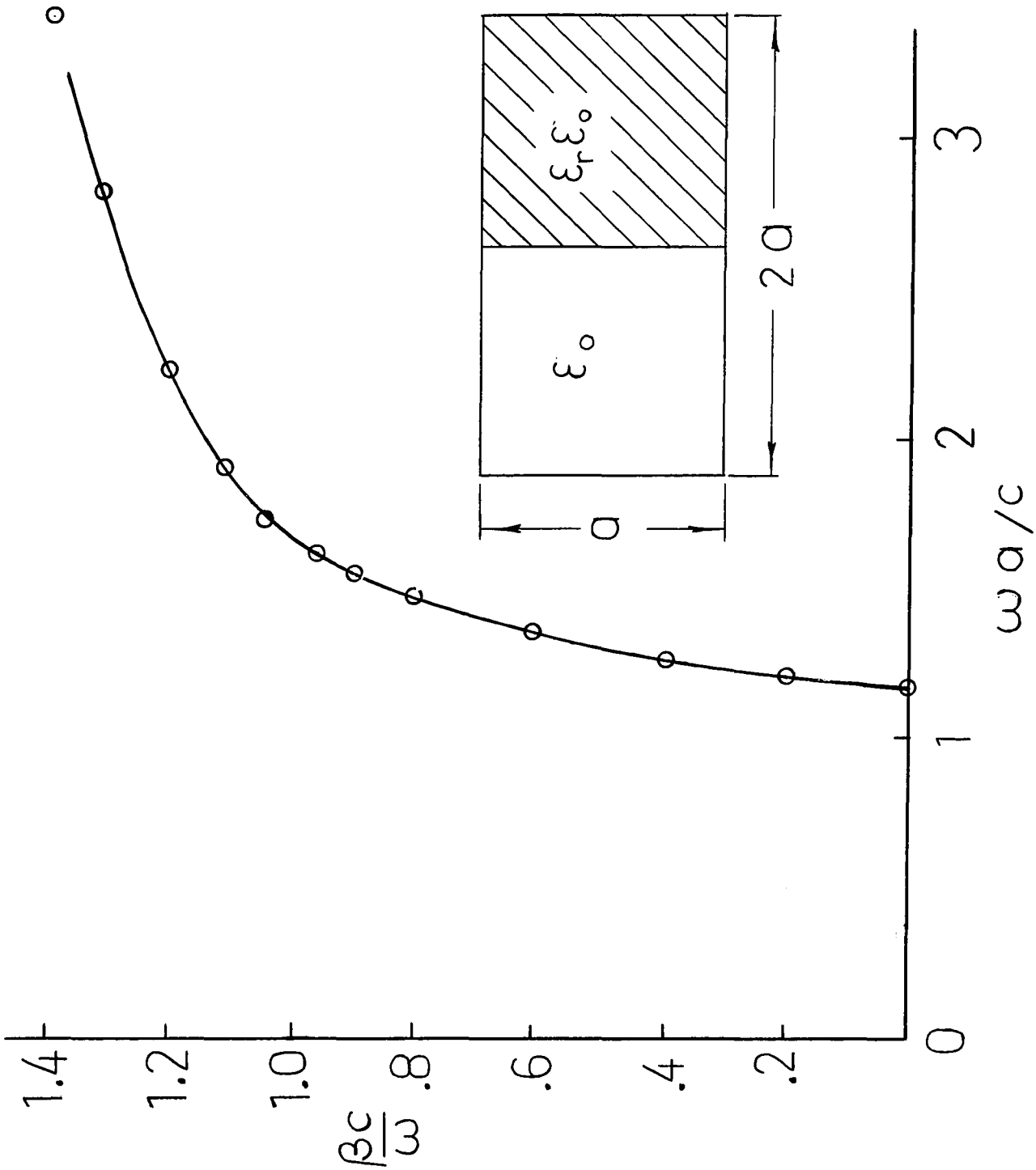


Figure 3.2: Dispersion curve for a waveguide partially loaded with dielectric. $\epsilon_r = 3.00$

— Lavik and Unger [17] o finite element

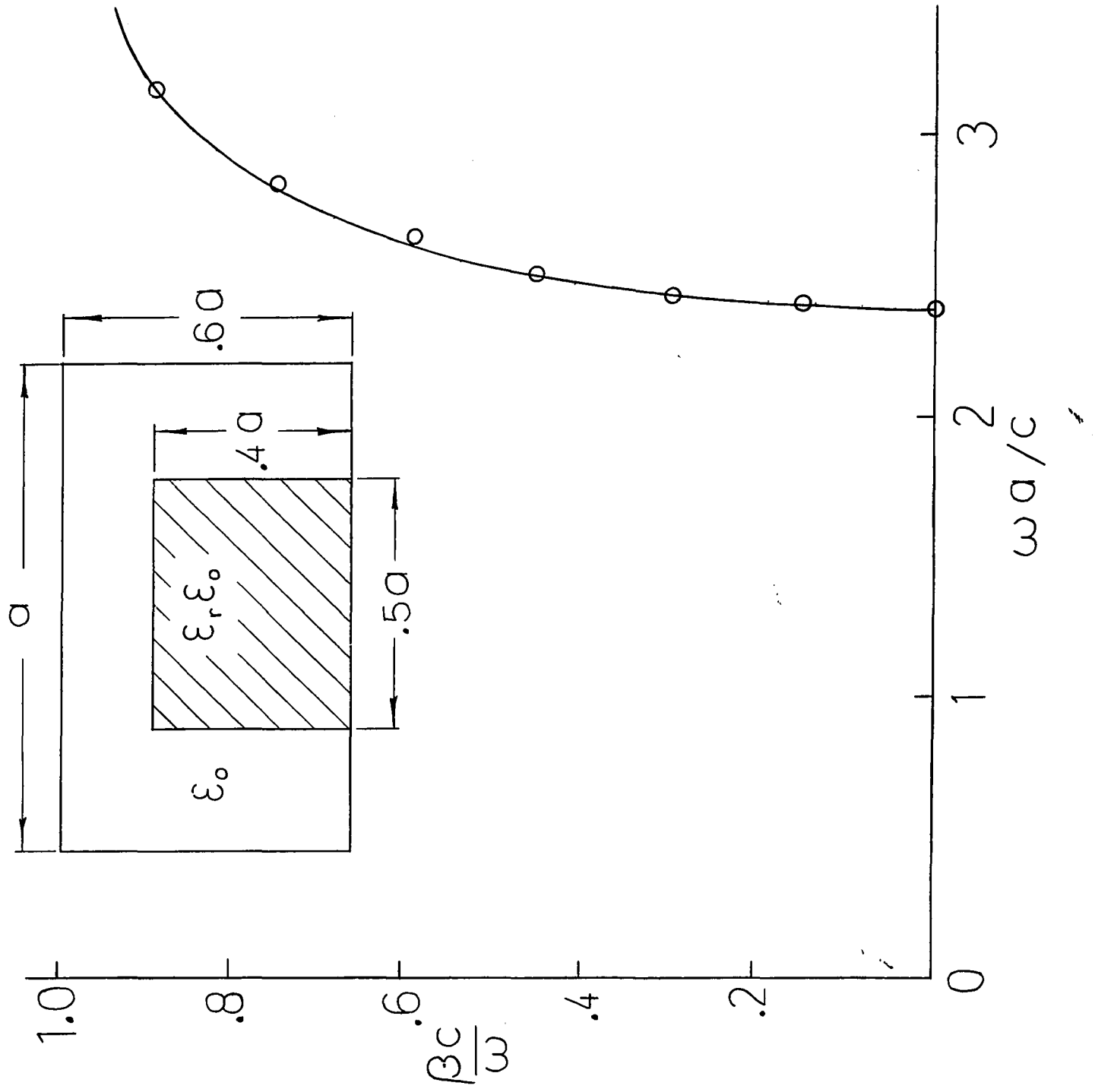
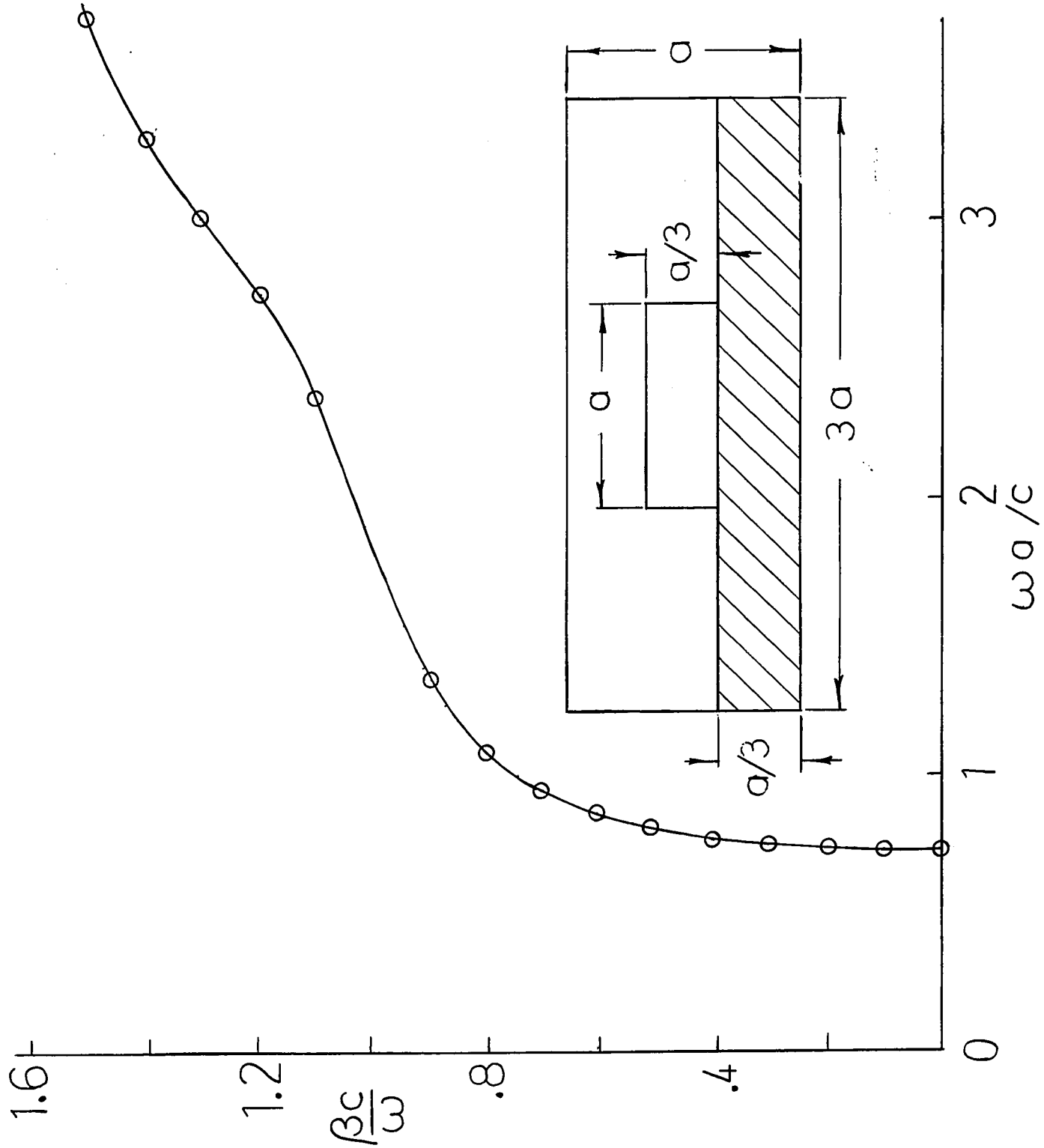


Figure 3.3: Dispersion curve for a microstrip conductor on a dielectric with $\epsilon_r = 3.58$ and surrounded by a zero potential rectangular box.



Chapter 4

SOLUTION BY ORTHOGONAL FUNCTIONS

4.1 Waveguide Mode Expansion

Although the conventional finite element method as developed in Chapter 3 represents the first practical solution method for general dielectric loaded waveguides, it has a serious deficiency: to obtain a dispersion curve, the same geometric problem is solved many times with no attempt to utilize previously computed results. In this chapter, a method is developed for expanding the fields above cutoff in terms of waveguide modes. This method will be shown to be much more efficient for treating the inhomogeneous waveguide problem than the conventional finite element method. A general inhomogeneous waveguide program will also be described and results for some interesting waveguide configurations will be given.

In order to present a conceptual picture of the disadvantages of the trial functions used in conventional finite element analysis, imagine a representative first order finite element trial function. One group of triangles rises from the bounded zero potential surface to form a pyramidal surface; the remainder is flat. This trial function is very easy to work with, but unfortunately, is not very closely related to the field distributions usually found in waveguides. Consequently, many such trial functions must be used to produce good approximations and a large matrix eigenvalue problem must be assembled and solved. Furthermore, the finite element trial functions are not orthogonal (they could be made orthogonal by lowering the potential reference level) and computational effort must be spent to assemble and decompose the S matrix.

In the search to find trial functions that are more naturally suited to the problem, the waveguide cutoff modes appear to be the ideal candidates. Each mode is a continuous function that exists exactly over the waveguide cross-section and satisfies all of the external boundary conditions. For rectangular waveguides, they are products of the sine and cosine functions, in circular waveguides the modes are composed of Bessel functions, and for other geometric shapes the modes are composed of functions as yet unnamed by mathematicians. Each of these waveguide modes forms an orthogonal set of functions. It is possible, therefore, to expand any function in a series of waveguide cutoff modes, provided the function satisfies the same boundary conditions. In particular, it is possible to expand the electric and magnetic fields in a waveguide at any value of phase velocity in the cutoff modes and, because of their close physical relationship, relatively few cutoff modes need to be used.

Their most important property, however, is their availability in the form of finite element approximations. Since the electric and magnetic fields are uncoupled only at cutoff, the analysis of cutoff modes may be performed in about one-quarter of the time and with one-quarter of the fast access memory that is required at any other value of phase velocity.

In such a waveguide mode expansion, it is useful to consider their orthogonality properties [1,2,48] . A simple proof of some of these follows directly from the matrix properties of (2.48) if a complete set of trial functions is chosen. When (2.48) is solved, the eigenvectors $\mathfrak{E}^{(i)}$, if properly normalized, form a basis in which $\mathcal{S}$ is the identity matrix and the diagonal elements of $\mathcal{U}$ are the eigenvalues k_o^2.

Consequently, the following orthogonality properties must hold for dielectric loaded waveguides of arbitrary shape:

$$\int_R \epsilon_t E_z^{(i)} E_z^{(j)} \, dR = \delta_{ij} \tag{4.1a}$$

$$\int_R \mu_t H_z^{(i)} H_z^{(j)} \, dR = \delta_{ij} \tag{4.1b}$$

$$\int_R \frac{\epsilon_t}{k_t^2} \nabla E_z^{(i)} \cdot \nabla E_z^{(j)} = \delta_{ij} \tag{4.1c}$$

$$\int_R \frac{\mu_t}{k_t^2} \nabla H_z^{(i)} \cdot \nabla H_z^{(j)} = \delta_{ij} \tag{4.1d}$$

$$\int_R \frac{1}{k_t^2} (\nabla E_z^{(i)} \times \nabla H_z^{(j)}) \cdot \vec{a}_z \, dR = 0 \tag{4.1e}$$

Then, taking the trial functions to be waveguide cutoff modes, (2.48) reduces to a matrix eigenvalue equation in standard form

$$\sum_t \sum_i \sum_j \frac{1}{\epsilon_t \mu_t - \delta^2} \begin{vmatrix} \epsilon_t \phi_i^{(e)} \phi_j^{(d)} S_{ij} & -\frac{\delta}{2} \phi_i^{(e)} \phi_j^{(h)} U_{ij} \\ -\frac{\delta}{2} \phi_i^{(e)} \phi_j^{(h)} U_{ij} & \mu_t \phi_i^{(h)} \phi_j^{(g)} S_{ij} \end{vmatrix} \begin{pmatrix} e \\ h \end{pmatrix} = k_0^2 \begin{pmatrix} e \\ h \end{pmatrix} \tag{4.2}$$

where $\phi_i^{(e)}$ is now the potential value of the (e)th mode obtained for point i in triangle t by conventional finite element analysis. The summations in this equation cannot be simplified and generally no element of this form of the $\mathcal{V}$ matrix is zero.

A theoretical comparison with the conventional finite element method is possible. If it is assumed that the order of the polynomial approximation is the same in both cases, then both sets of trial functions will span the same function space. Since each method will produce the best approximation possible in that function space, it follows that the eigenvalues of the two procedures must be identical.

Indeed, it is easy to show that this change of trial functions represents a change of basis in the N dimensional Hilbert space defined by the interpolation polynomials. A general modal matrix element M_{pq} is given by

$$M_{pq} = \sum_t \sum_i \sum_j \phi_i^{(p)} M_{ij} \phi_j^{(q)} = \left(\phi^{(p)}\right)^T \left(\mathcal{M}\right)\left(\phi^{(q)}\right) \qquad (4.3)$$

where M_{ij} is a matrix element for the order of the approximation considered, $(\mathcal{M})$ is the conventional finite element assembled matrix for N points and the (ϕ) are N-vectors. If the (ϕ) are normalized to $1/N$ then (4.3) may be written as

$$N\ M_{pq}\left(\phi^{(p)}\right) = \left(\mathcal{M}\right)\left(\phi^{(q)}\right) \qquad (4.4)$$

Next, choose the (ϕ) to be eigenvectors of (M). Then

$$N \sum_p M_{pq}\left(\phi^{(p)}\right) = N\,\nu\left(\phi^{(q)}\right) \qquad (4.5)$$

and

$$\sum_p \left(\mathcal{M}\right)\left(\phi^{(q)}\right) = N\left(\mathcal{M}\right)\left(\phi^{(q)}\right) = N\nu\left(\phi^{(q)}\right) \qquad (4.6)$$

Thus, if (ϕ) is an eigenvector of (M) with eigenvalue ν then (ϕ) is also an eigenvector of $(\mathcal{M})$ with eigenvalue ν.

The beauty of the mode expansion technique is now apparent. By expanding the fields in terms of their cutoff values, instead of some physically unrelated polynomials, the matrix eigenvalue equation requires much less transformation to obtain the solution. Typically, if one works with a one hundred point set, the conventional finite element method requires the assembly and solution this large eigenvalue problem. By comparison, in mode expansion, the highest ninety modes make negligible contribution to

to the dominant eigenvalue and, consequently, the assembly and solution of only one matrix of order twenty is required.

4.2 Computer Results

The parts of the computer program listed in Appendix 1 and not explained in Chapter 3 are designed to analyze dielectric loaded waveguides by the mode expansion technique. First the program solves for the waveguide cutoff electric and magnetic fields separately, using the conventional finite element method, as outlined in Chapter 3. These fields are normalized in the subroutine TNORM according to (4.1a&b) and stored in the spaces vacated by the bulky conventional finite element matrices. The computer then assembles the matrix in equation (4.2) for the requested values of δ in the subroutine VEE using the S matrix elements evaluated in SMAT. The eigenvalues are again evaluated by tridiagonalization and bisection and the eigenvectors computed by Wielandt iteration. On an IBM 360/75 the analysis for ten values on a dispersion curve of an 80 point set requires about 90 seconds of computing time. This is at most 1/20 of the time required for the conventional finite element solution on the same point set.

For the purpose of checking the orthogonality of the mode solution obtained by finite element analysis, the modal T matrix was assembled for several loaded waveguides. With the diagonal terms normalized to unity, the off-diagonal terms had values of the order of 10^{-5}. This value is considered satisfactory, since the arithmetic was performed using a 24 bit mantissa. In fact, the eigenvalues of the full eigenvalue problem and the eigenvalues computed assuming T to be the unit matrix were identical to six significant figures.

The other assumption made in the preceding analysis, that of using only about the first ten cutoff modes in the analysis, also passed computational tests. It is found that the components of modes ten or higher in the dominant

eigenvector are less than 1% of its major components. Since a variational method is used, this indicates that very good values for the dominant eigenvalue can be obtained with this method.

For comparative purposes, both conventional finite element solutions and cutoff mode approximation solutions were obtained for some particular geometries. It is found that for most of the dispersion curve, the values from the two methods are identical to three of four significant figures, but with much shorter computation times needed with the modal solution method.

In order to check the program results with analytically known values [3], rectangular waveguides with side and center slabs parallel to the E-field and with a side slab perpendicular to the E-field were solved. These results are shown in Figures 4.1 - 4.3. In addition, a rectangular waveguide with a dielectric ridge was solved with different values of permittivity; the results are compared with those of Lavik and Unger (17) in Figure 4.4. Note that all values are in excellent agreement.

In addition, several waveguide configurations have been investigated which have not been solved previously. Figure 4.5 shows the dispersion characteristies of a rectangular waveguide with a diamond shaped dielectric insert and in Figure 4.6 are the dispersion characteristies of a "maple leaf" waveguide with three dielectric inserts of different permittivities. It can be seen that by varying the values of permittivity in the three regions, dispersion curves with different characteristies are obtained. Consequently, the method can be used in practical applications not only to obtain the fields in waveguides already built, but also to design waveguides in which the dispersion curve has some desired form.

The remaining figures contain contour plots showing the field variation with different values of phase velocity.

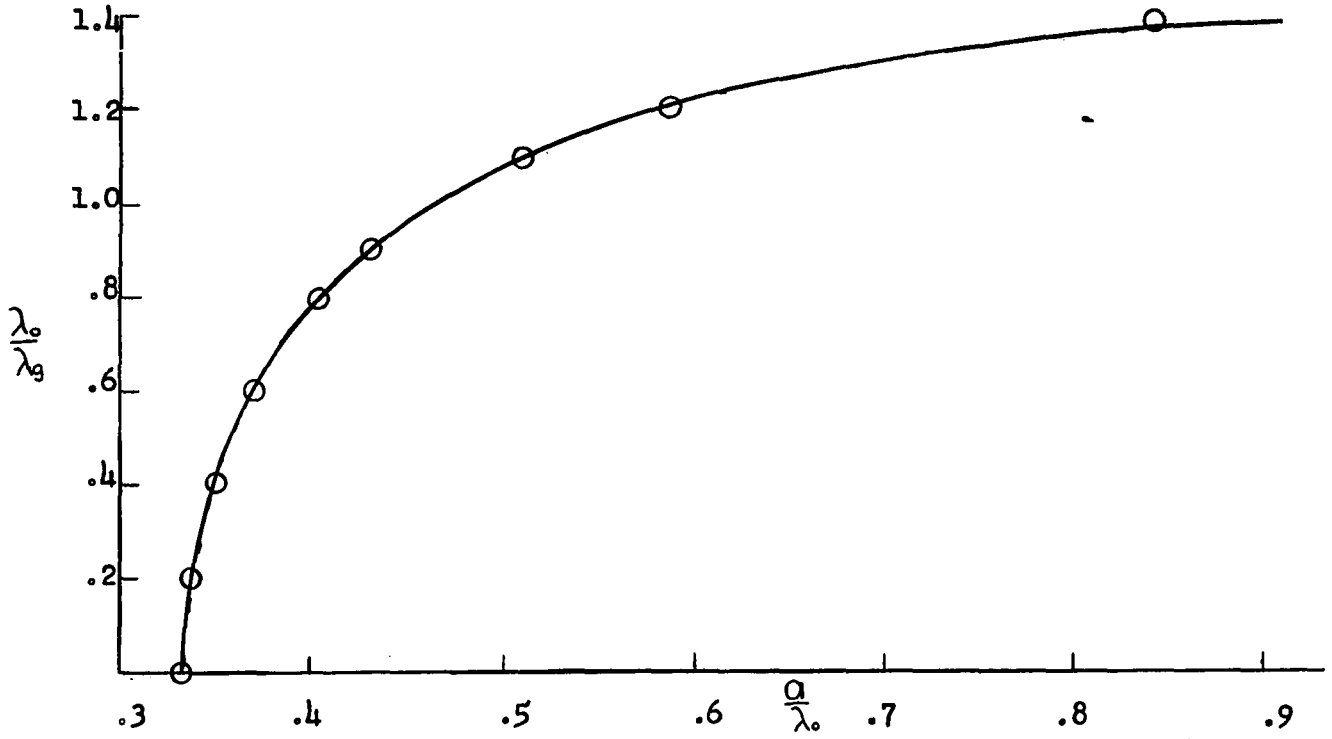


Figure 4.1: Propagation characteristic of a rectangular waveguide half-filled with a central dielectric slab parallel to E. $\epsilon_r = 2.45$

—— Marcuvitz [3] o modal approximation

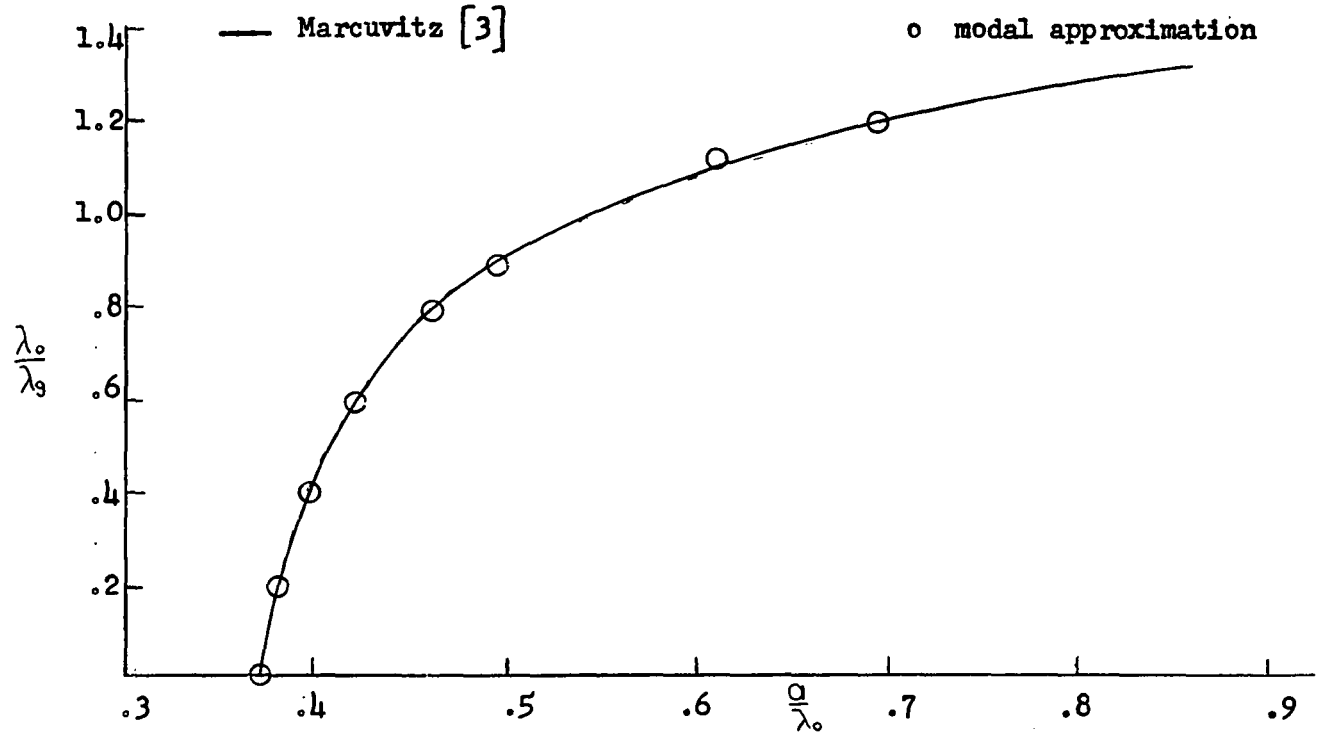


Figure 4.2: Propagation characteristic of a rectangular waveguide half-filled with a dielectric slab parallel to E and at one side. $\epsilon_r = 2.45$

—— Marcuvitz [3] o modal approximation

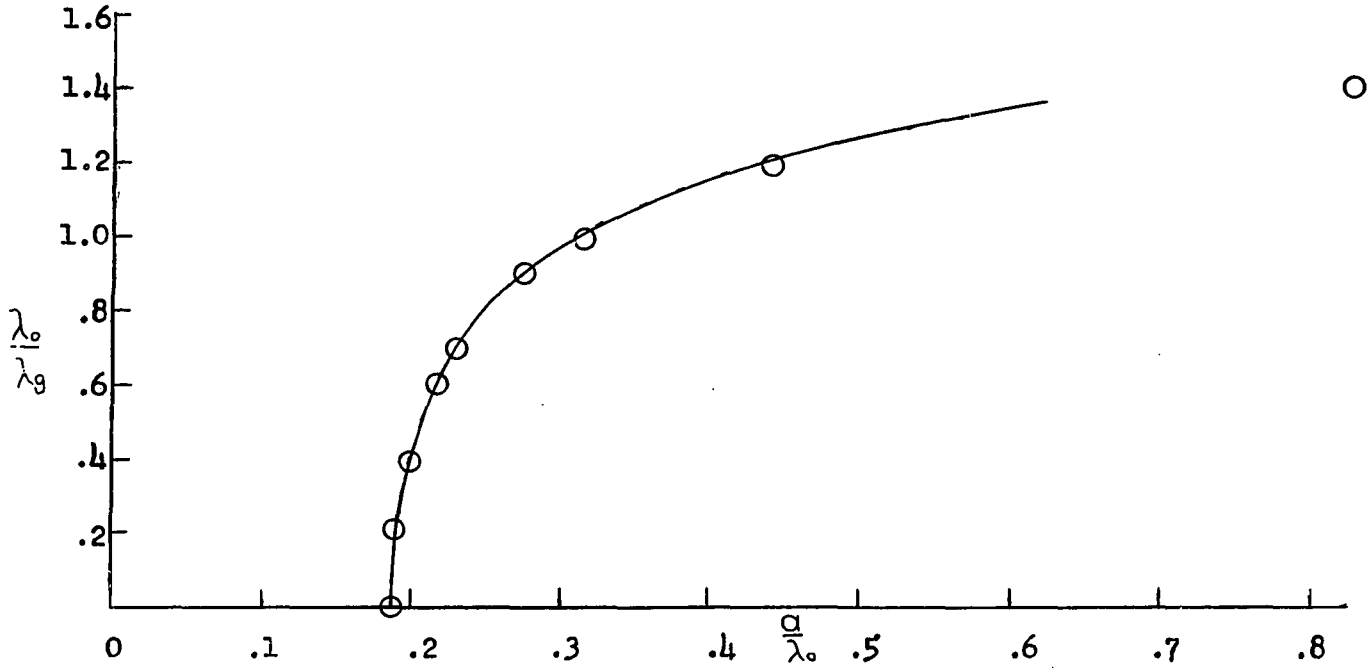


Figure 4.3: Propagation characteristic of a rectangular waveguide half-filled with a dielectric slab perpendicular to E and at one side $\epsilon_r = 2.45$.

—— Marcuvitz [3] o modal approximation

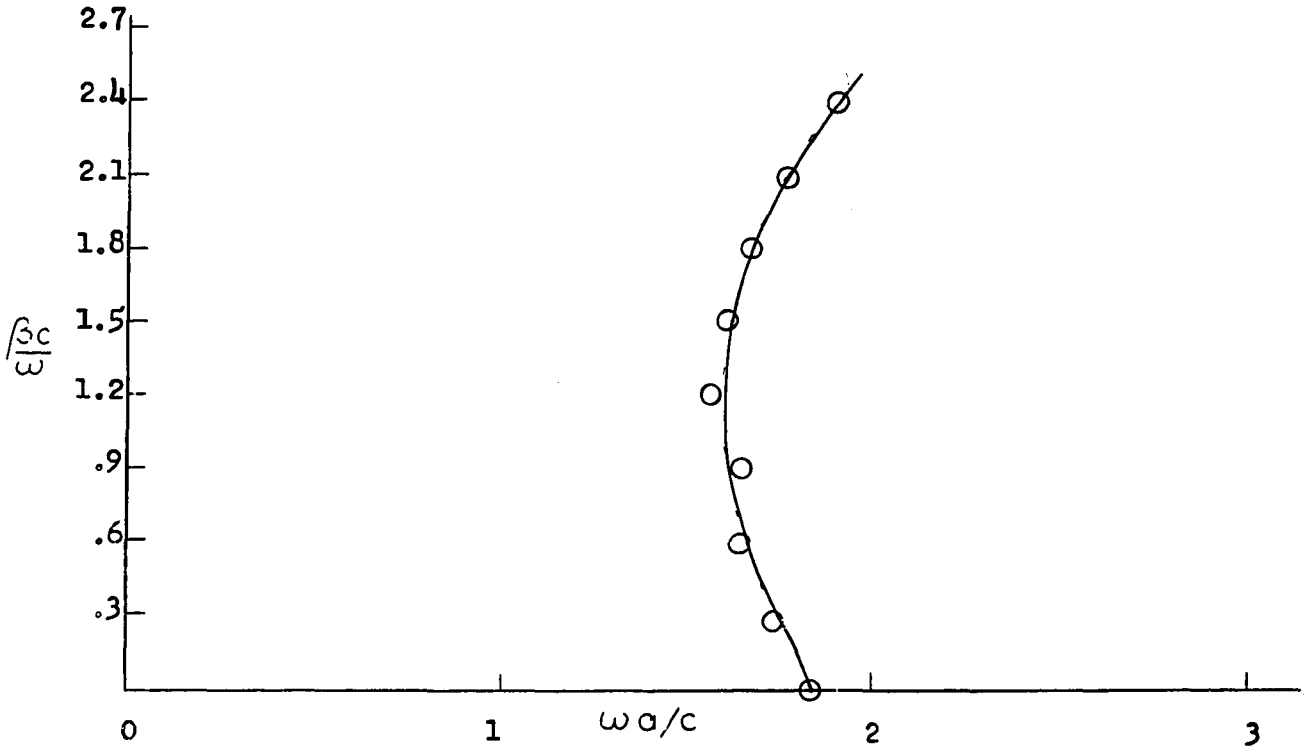


Figure 4.4: Dispersion curve of the waveguide in Figure 3.2 with $\epsilon_r = 15.0$.

—— Lavik and Unger [17] o modal approximation

Figure 4.5: Dispersion characteristic of a rectangular waveguide with a diamond shaped dielectric insert. $\epsilon_r = 10.0$

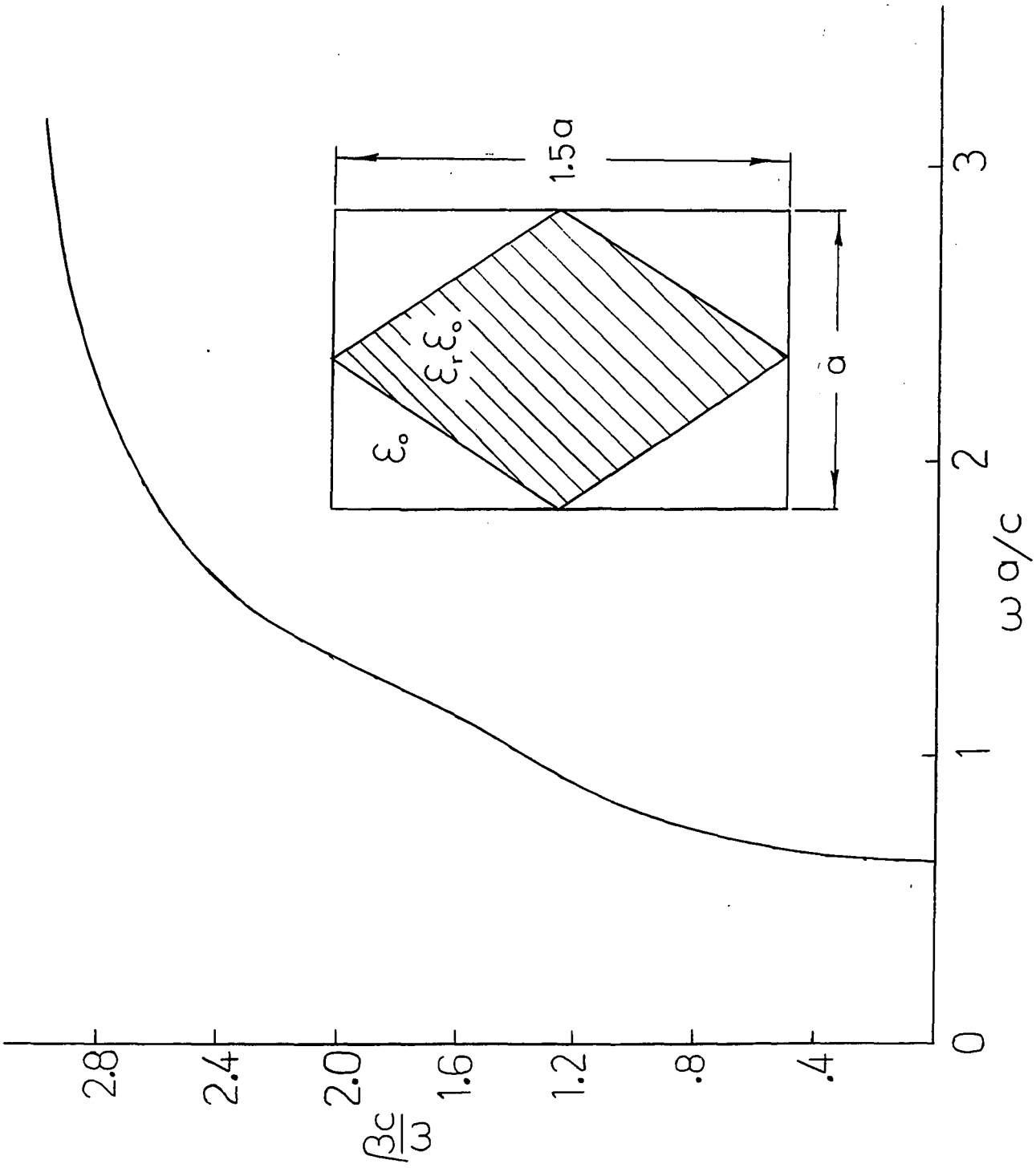


Figure 4.6: Dispersion characteristics of two inhomogeneous waveguides with a "maple leaf" cross-section.

I. $\epsilon_1 = 3.0$, $\epsilon_2 = 6.0$ and $\epsilon_3 = 1.5$
II. $\epsilon_1 = 6.0$, $\epsilon_2 = 1.5$ and $\epsilon_3 = 3.0$

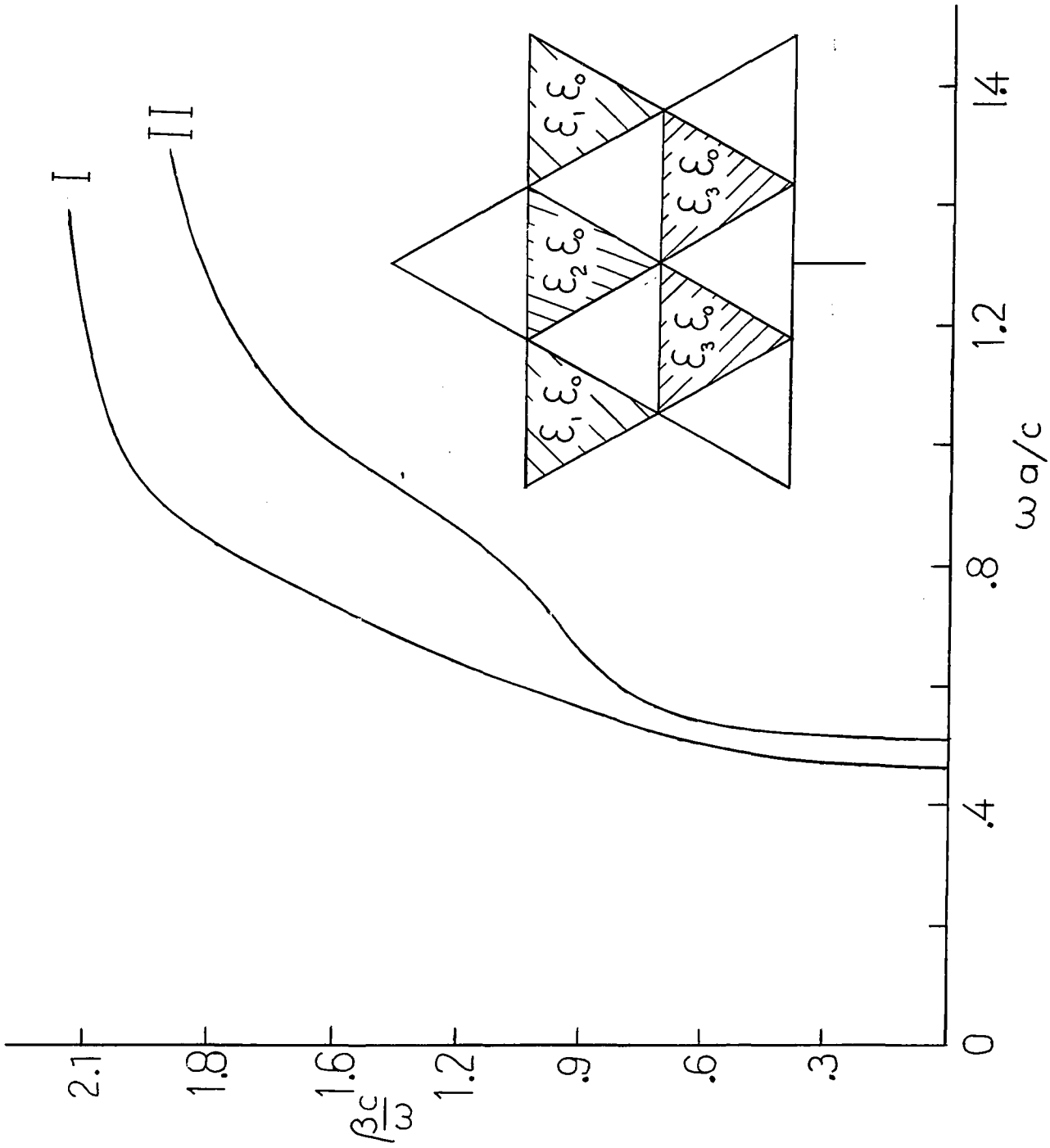


Figure 4.7 : Field plot of the first electric mode at cutoff for the waveguide in figure 3.2 .

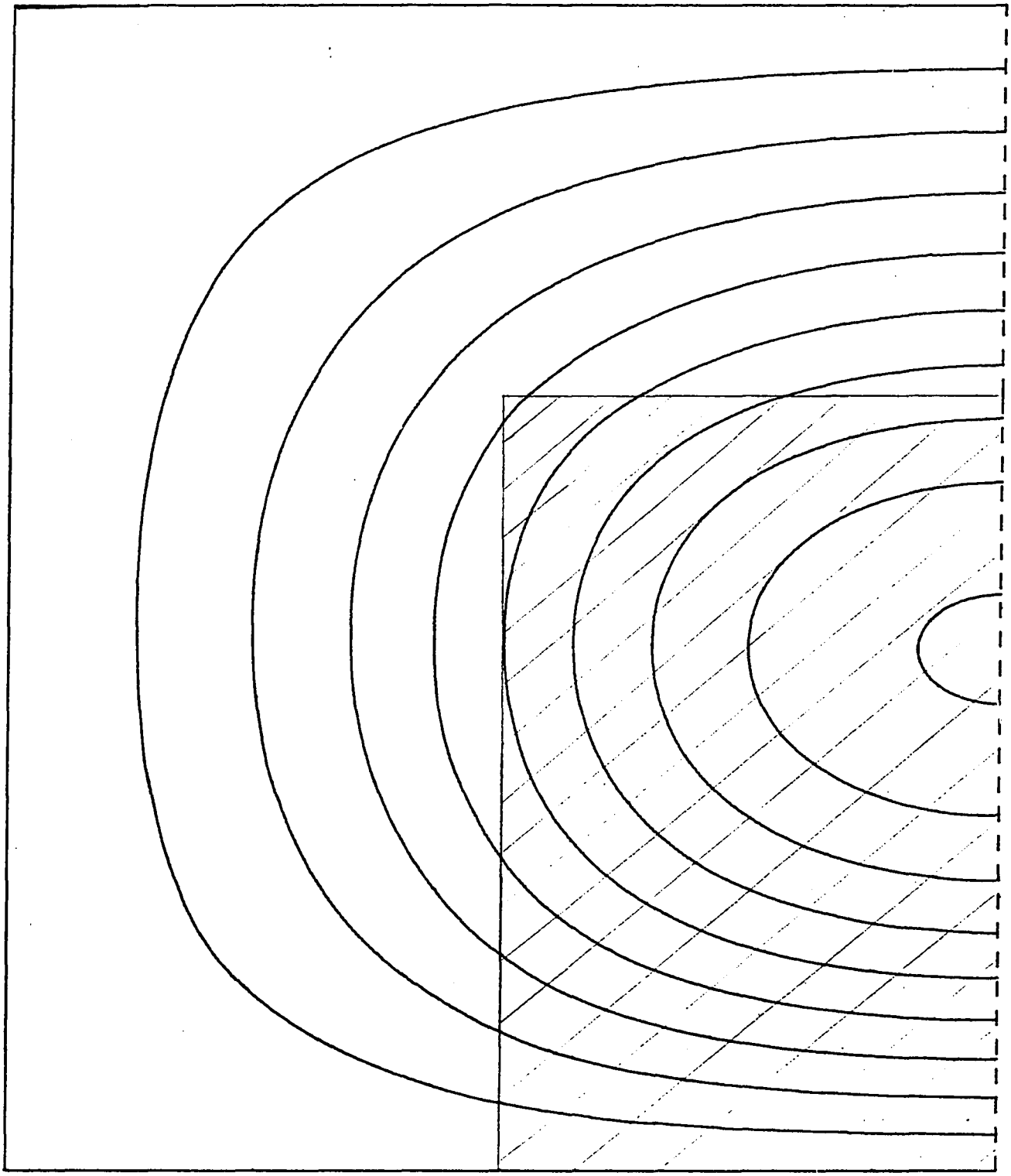
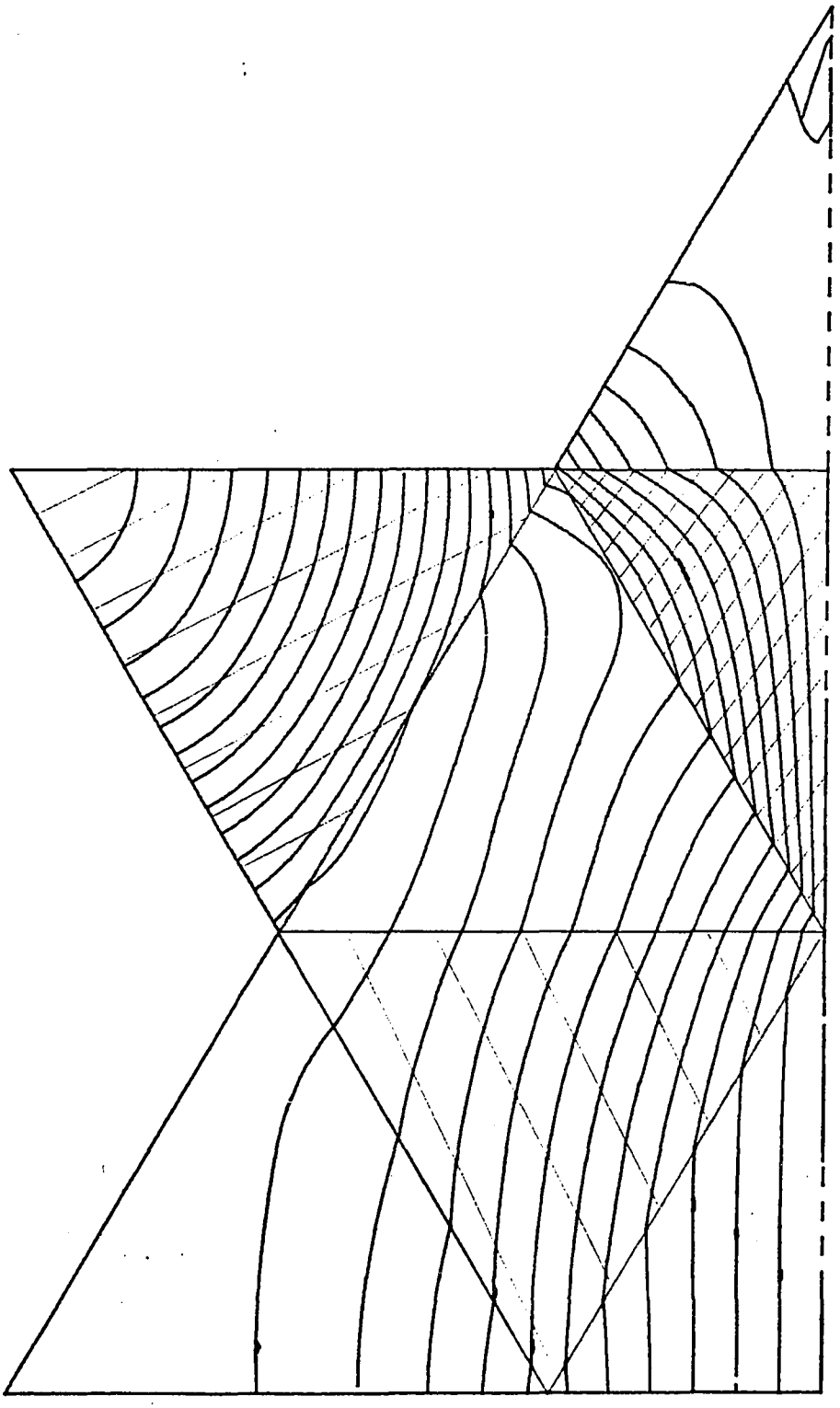


Figure 4.8 : Field plot of the first magnetic mode at cutoff for the "maple leaf" waveguide in figure 4.6 l .



CHAPTER 5

CONCLUSIONS

The two finite element methods described in this thesis can be used to determine the electromagnetic fields in dielectric loaded waveguides. Both methods rely on the same variational formulation and both produce the best possible approximation in their respective function spaces. However, the mode expansion technique is more naturally suited to the problem and, as a result, produces field solutions with greater computational efficiency.

There are two apparent drawbacks in the methods developed in this thesis. The first is the appearance of extraneous solutions for values of phase velocity where the functional is indefinite. This difficulty is implicit in the variational formulation of multi-dielectric problems and, while such solutions may be detected, they cannot be eliminated from the procedure. The other is the impossibility of exactly modeling waveguide problems with curved boundaries by triangular finite elements. For such problems, good approximate solutions are obtained by representing a curved side by a many sided polygon; however, this detracts from the inherent accuracy and simplicity of the method.

These difficulties notwithstanding, the general finite element computer program in Appendix 1 provides an accurate, reliable, economical and easy to use method to solve dielectric loaded waveguides. It is hoped that its use will facilitate their design and that some of the procedures developed in this thesis will be extended to solve other problems in electromagnetic field theory.

REFERENCES

(1) R.E. Collin, Field Theory of Guides Waves, Chapter 5, New York: McGraw-Hill, 1960.

(2) S.K. Chatterjee and R. Chatterjee, "Dielectric loaded waveguides - a review of theoretical solutions", The Radio and Electronic Engineer, Vol. 30, Nos. 3,4, 5 and 6, pp. 145-160, 195-205, 259-288 and 353-364.

(3) N. Marcuvitz, Waveguide Handbook, New York: McGraw-Hill, 1951.

(4) R. Seckelmann, "Propagation of TE modes in dielectric loaded waveguides", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-14, No. 11, pp. 518-527, November 1966.

(5) F.E. Gardiol, "Higher-Order modes in dielectrically loaded rectangular waveguides", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-16, No. 11, pp. 919-924, November 1968.

(6) N. Eberhardt, "Propagation in the off center E-plane dielectrically loaded waveguide", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-15, No. 5, pp. 282-289, May 1967.

(7) J.H. Collins and P. Daly, "Calculations for guided electromagnetic waves using finite difference methods", J. ELECTRONICS CONTROL., Vol. 14, No. 4, pp. 361-380, 1963.

(8) C.D. Hannaford, The Finite Difference/Variational Method for Inhomogeneous Isotropic or Anisotropic Waveguides, Ph.D. thesis, University of Leeds, 1967.

(9) D. Corr, J.B. Davies and C.A. Muilwyk, "Finite-difference solution of arbitrarily-shaped dielectric loaded waveguides, including microstrip and coaxial structures", U.R.S.I. Symposium on Electromagnetic Waves, Stresa, Italy, June 1968.

(10) J.S. Hornsby and A. Gopinath, "Numerical analysis of a dielectric-loaded waveguide with a microstrip line - finite difference methods", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-17, No. 9, pp. 684-690, September 1969.

(11) L. Cairo and T. Kahan, Variational Techniques in Electromagnetism, New York: Gordon and Breach, 1965.

(12) R.F. Harrington, Field Computation by Moment Methods, New York: MacMillan, 1968.

(13) D.T. Thomas, "Functional approximations for solving boundary value problems by computer", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-17, pp. 447-454, 1969.

(14) D.T. Thomas, "Choice of trial functions in matrix methods", PROC. IEEE, Vol. 55, pp. 1106-1108, June 1967.

(15) W.J. English, Variational Solutions of the Maxwell Equations with Applications to Wave Propagation in Inhimogeneously Filled Waveguides, Ph.D. thesis, Carnegie Institute of Technology, 1969.

(16) A. Lavik and H.G. Unger, "Der rechteckhohlleiter mit rechteckigem Stoffeinsatz", A.E.U., Vol. 18, pp. 25-34, 1964.

(17) A. Lavik and H.G. Unger, "Ruchwartswellen in homogenen Wellenleitern", A.E.U., Vol. 18, pp. 35-42, 1964.

(18) W. Schlesser and H.G. Unger, Advances in Microwaves, Vol. 1, pp. 340-415, 1966.

(19) P.L. Arlett, A.K. Bahrani and O.C. Zienkiewicz, "Application of finite elements to the solution of Helmholtz's equation", PROC. IEE, Vol. 115, No. 12, pp. 1762-1766, December 1968.

(20) S. Ahmed and P. Daly, "Finite-element methods for inhomogeneous waveguides", PROC. IEE, Vol. 116, No.10, pp. 1661-1664, October 1969.

(21) I. Stakgold, Boundary Value Problems of Mathematical Physics, New York: MacMillan, 1968.

(22) D.S. Jones, The Theory of Electromagnetism, New York: MacMillan, 1964.

(23) R.F. Harrington, Time Harmonic Electromagnetic Fields, New York: McGraw-Hill, 1961.

(24) A.D. Berk, "Variational Principles for electromagnetic resonators and waveguides", I.R.E. TRANS. ANTENNAS AND PROPAGATION, Vol. AP-4, No. 2, pp. 104-111, 1956.

(25) P. Silvester and Z. Csendes, "Finite element solution of dielectric loaded waveguides", 1970 International G-MTT Symposium (to be published).

(26) J.W. Gibbs and E.B. Wilson, Vector Analysis, New York: Dover , 1960.

(27) S. Ahmed, "Finite-element method for waveguide problems", ELECTRONICS LETTERS, Vol. 4, No. 18, pp. 387-9, September 1968.

(28) F.B. Hildebrand, Methods of Applied Mathematics, Englewoods Cliffs, N.J.: Prentice-Hall, 1965.

(29) S.G. Mikhlin and K.L. Smolitskiy, Approximate Methods for Solution of Differential and Integral Equations, New York: Elsevier, 1967.

(30) P. Silvester, "Finite element solution of homogeneous waveguide problems", ALTA FREQUENZA (SPECIAL ISSUE ON URSI STRESA SYMPOSIUM, 1968) Vol. 38, pp. 313-317, May 1969.

(31) P. Silvester, "A general high-order finite-element waveguide analysis program", IEEE TRANS. MICROWAVE THEORY AND TECHNIQUES, Vol. MTT-17, No. 4, pp. 204-210, April 1969.

(32) E.R. de Arantes, "Theoretical foundations of the finite element method", INTERNATIONAL JOURNAL OF SOLIDS AND STRUCTURES, Vol. 4, pp. 929-952, 1968.

(33) T.H.H. Pian and Pin Tong, "Basis of finite element methods for solid continua", INTERNATIONAL JOURNAL. NUMERICAL METHODS IN ENGINEERING, Vol. 1, pp. 3-28, 1969.

(34) G. Salmon, Conic Sections, New York: Chelsea, 1954.

(35) E.A. Maxwell, General Homogeneous Coordinates, Cambridge: University Press, 1960.

(36) F.S. Woods, Higher Geometry, New York: Dover, 1961.

(37) P. Silvester, "High-order polynomial triangular finite elements for potential problems", INT. J. ENGNG. SCI., Vol. 7, pp. 849-861, 1969.

(38) E. Isaacson and H.B. Keller, Analysis of Numerical Methods, New York: John Wiley, 1966.

(39) O.C. Zienciewicz with Y.K. Cheung, The Finite Element Method in Structural and Continuum Mechanics, London: MacGraw-Hill, 1967.

(40) P.C. Dunne, JOUR. ROY. AERO. SOC., Vol. 72, pp. 245, 1968.

(41) P.M. Morse and H. Fesbach, Methods of Theoretical Physics, New York: McGraw-Hill, 1953.

(42) G.C. Best, "Helpful formulas for integrating polynomials in three dimensions".

(43) P. Silvester, "Symmetric Quadrature formulae for simplexes", MATHEMATICS OF COMPUTATION, 1969.

(44) P. Daly, "Finite-element coupling matrices", ELECTRONICS LETTERS, Vol. 5, No. 24, pp. 613-615, November 1969.

(45) J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford: Clarendon, 1965.

(46) L. Fox, Introduction to Numerical Linear Algebra, Oxford: Clarendon, 1964.

(47) P. Silvester, Lecture notes.

(48) A.D. Bresler, G.H. Joshi and N. Marcuvitz, "Orthogonality properties in passive and active uniform wave guides", J. APPLIED PHYSICS, Vol. 29, No. 5, pp. 794-799, 1958.

```
C
C
C                                  APPENDIX 1
C
C
C
C      GENERAL FINITE ELEMENT DIELECTRIC LOADED WAVEGUIDE
C                          ANALYSIS PROGRAM
C
C
C
C      THIS COMPUTER PROGRAM ANALYSES DIELECTRIC LOADED WAVEGUIDES
C      BY FINITE ELEMENT METHODS. THE WAVEGUIDE CROSS-SECTION MUST
C      BE DIVIDED INTO TRIANGULAR ELEMENTS, THE VERTICES OF WHICH
C      ARE READ IN AS DATA. THE USER MUST ALSO SPECIFY THE VALUES OF
C      DELTA = BETA/OMEGA TO BE USED AND DECIDE TO USE EITHER THE
C      CONVENTIONAL FINITE ELEMENT METHOD OR THE MODAL APPROXIMATION
C      METHOD. THE LATTER IS TO BE MUCH PREFERRED SINCE IT IS ABOUT
C      20 TIMES FASTER AND WILL ACCOMODATE TWICE AS MANY POINTS.
C
C
C      EACH SET OF DATA MUST BE ARRANGED IN THE FOLLOWING MANNER.
C      FIRST CARD:
C      COLUMN 1      *   THIS SYMBOL MUST APPEAR AT THE BEGINNING
C                    OF EACH DATA SET.
C             2-41   TITLE
C            45-54   SCALE FACTOR FOR X - IF BLANK, 1. IS ASSUMED
C            55-64   SCALE FACTOR FOR Y - IF BLANK, 1. IS ASSUMED
C            65      ORDER OF POLNOMIAL DESIRED - 1 THROUGH 4 ALLOWED
C            69-70   NUMBER OF MODES TO BE USED - USUALLY ABOUT 10
C            79-80   NUMBER OF EIGENVECTORS TO BE CALCULATED
C      THE NEXT GROUP OF CARDS MUST CONTAIN THE VERTEX NUMBERS, ONE
C      AT A TIME AND IN INCREASING ORDER, AND THEIR CORRESPONDING
C      COORDINATES.
C      COLUMN 2-5    VERTEX NUMBER
C             6-15   X COORDINATE
C            16-25   Y COORDINATE
C      THIS GROUP OF CARDS IS FOLLOWED BY A BLANK CARD AND THEN
C      CARDS  WITH THE TRIANGULAR INFORMATION
C      COLUMNS 2-5,6-10,11-15  THREE VERTEX NUMBERS FOR EACH TRIANGLE-
C                    THESE MUST BE COUNTERCLOCKWISE
C            30      THE CONSTRAINT NUMBER FOR SIDE 1-2
C            31      THE CONSTRAINT NUMBER FOR SIDE 2-3
C            32      THE CONSTRAINT NUMBER FOR SIDE 1-3
C                    THESE CONSTRAINT NUMBERS ARE:
C                         0 FOR NO CONSTRAINT
C                         1 FOR METAL BOUNDARIES
C                         2 FOR EXTERNAL LINES OF SYMMETRY
C      FIN ALLY, THE VALUES OF DELTA ARE READ IN TEN AT A TIME
C      COLUMN 2,9,16,...,72          1 FOR CONVENTIONAL FINITE ELEMENT AN
C                                    2 FOR MODAL APPROXIMATION
C             4-7,11-14,...,74-77    DELTA
C      MORE VALUES OF DELTA MAY BE  READ IN IF THE NEXT CARD HAS A
C      + IN COLUMN 1 AND THE FOLLOWING CARD HAS MORE VALUES OF DELTA.
C
```

```
      C
)001        DIMENSION A(2628),V(2628),T(2628),DIAG(72),SUBD(72)
)002        DIMENSION AREA(100),COTNG(3,100),NVTX(300),WT(2,100)
)003        DIMENSION WSQ(15),IDEL(10),DEL(10),SMAT(1),E(15,1),H(15,1),W(1)
)004        EQUIVALENCE (A(466),SMAT(1)),(V(466),H(1,1)),(T(466),E(1,1)),
           1          (V(1),W(1))
)005        COMMON T,A,V
)006       1 CALL READIN(AREA,COTNG,WT,NVTX,NFIT,NPTS,NFREE,NEL,
           1          NMAG,NTOTPT,NELMT,IDEL,DEL,NMODES,NVECT)
)007        NMODES = MINO(NMODES,15)
)008        NBIG = NPTS*(NPTS + 1)/2
)009        REWIND 1
)010        CALL SMATRX(SMAT,COTNG,NFIT,NPTS,NBIG,NELMT)
)011        DO 30 II = 1,10
)012        IF(IDEL(II).EQ.0) GO TO 1
)013        IF(IDEL(II).EQ.2) GO TO 7
)014        IVECT = NMODES
)015        IF(DEL(II).EQ.0.) GO TO 2
)016        N = 2*NFREE + NEL + NMAG
)017        GO TO 3
)018       2 N = NFREE + NMAG
)019        IEH = 1
)020       3 NERROR = N*(N + 1)/2
)021        IF(NERROR.GT.2628) GO TO 50
)022        DO 4 I = 1,NERROR
)023        V(I) = 0.
)024       4 T(I) = 0.
)025        CALL ASSEMB(V,T,AREA,SMAT,WT,NVTX,NFREE,NEL,NMAG,
           1          NFIT,NPTS,NBIG,NELMT,IEH,DEL(II))
)026        DO 5 I = 1,NERROR
)027       5 A(I) = 0.
)028        CALL CHOLOW(T,T,N,NERROR)
)029        IF(NERROR.EQ.0)GO TO 6
)030        GO TO 51
)031       6 CALL INVLOW(T,N)
)032        CALL GAGT(A,T,V,N)
)033        GO TO 8
)034       7 N = 2*NMODES
)035        IVECT = NVECT
)036        CALL VEE(A,E,H,SMAT,WT,NVTX,NFIT,NPTS,NBIG,NELMT,NMODES,DEL(II))
)037        WRITE(6,100) DEL(II)
)038
)039       8 CALL TRIDIA(A,DIAG,SUBD,N)
)040        NEIGV = 15
)041        CALL BISLCT(DIAG,SUBD,N,LEAST,NEIGV,WSQ)
)042        IF(LEAST.GT.N) GO TO  52
)043        DO 9 J = 1,NEIGV
)044       9 W(J) = SQRT(WSQ(J))
)045        LEAST = LEAST - 1
)046        WRITE(6,101) DEL(II),LEAST
)047        WRITE(6,102) (J,W(J),J=1,NEIGV)
)048        IVECT = MINO(IVECT,NEIGV)
)049        DO 21 I = 1,IVECT
)050        CALL WIELND(DIAG,SUBD,N,WSQ(I),W)
```

```
0051                CALL REVERS(A,SUBD,W,N)
0052                IF(IDEL(II).EQ.2) GO TO 20
0053                CALL TRIMUL(W,T,W,N)
0054                N1 = N + 1
0055                IF(DEL(II).NE.0.) GO TO 16
0056                DO 10 J = N1,NTOTPT
0057             10 W(J) = 0.
0058                IF(IEH.EQ.2) GO TO 14
0059                JX = NFREE
0060                LEAST = NEL + NMAG
0061                DO 11 J = 1,LEAST
0062                JX = JX + 1
0063             11 V(J + 300) = W(JX)
0064                JX = NFREE
0065                DO 12 J = 1,NEL
0066                JX = JX + 1
0067             12 W(JX) = V(J + NMAG + 300)
0068                DO 13 J = 1,NMAG
0069                JX = JX + 1
0070             13 W(JX) = V(J + 300)
0071                WRITE (1) (W(J),J=1,NTOTPT)
0072                GO TO 21
0073             14 DO 15 J = 1,NTOTPT
0074             15 H(I,J) = W(J)
0075                GO TO 21
0076             16 LEAST = 2*NTOTPT
0077                DO 17 J = N1,LEAST
0078             17 W(J) = 0.
0079                LEAST = NFREE + NEL + 1
0080                DO 18 J = 1,NTOTPT
0081             18 H(I,J) = W(J + LEAST)
0082                LEAST = NFREE + 1
0083                DO 19 J = LEAST,NTOTPT
0084             19 W(J) = 0.
0085                WRITE (1) (W(J),J=1,NTOTPT)
0086                GO TO 21
0087             20 WRITE(6,103) I
0088                WRITE(6,104) (J,W(J),J=1,N)
0089             21 CONTINUE
0090                IF(IDEL(II).EQ.2) GO TO 30
0091                IF(DEL(II).NE.0. .OR. IEH .NE. 1) GO TO 24
0092                N = NFREE + NEL
0093                IEH = 2
0094                CALL SMATRX(SMAT,COTNG,NFIT,NPTS,NBIG,NELMT)
0095                GO TO 3
0096             24 REWIND 1
0097                DO 22 I = 1,NMODES
0098             22 READ (1) (E(I,J),J=1,NTOTPT)
0099                CALL TNORM(E,H,AREA,WT,NVTX,NFIT,NPTS,NBIG,NTOTPT,NELMT,NMODES)
0100                DO 23 I = 1,NMODES
0101                WRITE(6,103) I
0102                WRITE(6,104) (J,E(I,J),J=1,NTOTPT)
0103                WRITE(6,105)
0104             23 WRITE(6,104) (J,H(I,J),J=1,NTOTPT)
```

```
0105              REWIND 1
0106              CALL SMATRX(SMAT,COTNG,NFIT,NPTS,NBIG,NELMT)
0107           30 CONTINUE
0108           50 WRITE(6,106) N,NERROR
0109              GO TO 1
0110           51 WRITE(6,107)NERROR
0111              GO TO 1
0112           52 WRITE(6,108)DEL(II)
0113              GO TO 1
0114          100 FORMAT(1H1,40X,'V MATRIX FOR DELTA =',F10.5)
0115          101 FORMAT(1H1,30X,'ANGULAR FREQUENCY OMEGA FOR DELTA =',F10.5//
                 1           33X,'(THERE ARE',I3,' IMAGINARY FREQUENCIES)'//)
0116          102 FORMAT(5(I7,E13.6))
0117          103 FORMAT(1H-,32X,'EIGENVECTOR NUMBER',I3//)
0118          104 FORMAT(5(I6,F10.6))
0119          105 FORMAT(1H-)
0120          106 FORMAT('-THE MATRIX SIZE IS TOO LARGE. ITS RANK IS'I5,' AND NEEDS
                1A DIMENSION OF',I5)
0121          107 FORMAT('-CHOLOW HAS ENCOUNTERED A NONPOSITIVE PIVIT - NUMBER',I7)
0122          108 FORMAT('-ALL EIGENVALUES ARE NEGATIVE - DELTA =',F10.5)
0123              END
```

TOTAL MEMORY REQUIREMENTS 002160 BYTES

```
0001                SUBROUTINE READIN(AREA,COTNG,WTS,NVTX,NFIT,NEW,NFREE,NEL,
                   1      NMAG,NPTS,NELMT,IDEL,DEL,NMODES,NVECT)
0002                DIMENSION IDEL(1),DEL(1)
0003                DIMENSION AREA(1),COTNG(3,1),WTS(2,1),NVTX(1)
0004                DIMENSION TITLE(10),INLIST(3),KONSTR(3)
0005                DIMENSION XX(25), YY(25), NEWVTX(25), NEWDIR(25)
0006                DIMENSION ND(800),X(800),Y(800),WORKX(800),WORKY(800),NFR(4)
0007                DATA IBLANK,ISTAR,IPLUS/1H ,1H*,1H+/
0008                COMMON ND,X,Y,WORKX,WORKY,XX,YY,NEWDIR,NEWVTX,TITLE
           C
0009              1 READ(5,100,END=49,ERR=50) MIND,TITLE,SCALX,SCALY,N,NN,M
0010                IF(MIND.EQ.IPLUS) GO TO 31
0011                NFIT = N
0012                NMODES = NN
0013                NVECT = M
0014                IF(MIND.NE.ISTAR) GO TO 1
0015                IF(SCALX.EQ.0.) SCALX = 1.
0016                IF(SCALY.EQ.0.) SCALY = 1.
0017                WRITE(6,101) TITLE
0018                NN = 1
0019              2 READ(5,102,END=49,ERR=50)MIND,N,EX,WHY
0020                IF(MIND.NE.IBLANK) GO TO 50
0021                IF(N.LE.0) GO TO 3
0022                IF(N.NE.NN) GO TO 50
0023                NN = NN + 1
0024                X(N) = EX * SCALX
0025                Y(N) = WHY * SCALY
0026                WRITE(6,103) N,X(N),Y(N)
0027                GO TO 2
           C
0028              3 NPTS = NN - 1
0029                REFX = 0.
0030                REFY = 0.
0031                DO 7 I = 1,NPTS
0032                REFX = AMAX1(REFX, ABS(X(I)))
0033                REFY = AMAX1(REFY, ABS(Y(I)))
0034              7 CONTINUE
0035                REFX = REFX * 1.E-5
0036                REFY = REFY * 1.E-5
0037                DO 8 I = 1,NPTS
0038              8 ND(I) = 1
0039                NV = 0
0040                NN = 0
0041                WRITE(6,104)
0042              4 NN = NN + 1
0043                READ(5,105,END=49,ERR=50)M,INLIST,KONSTR,EX,WHY
0044                IF(M.NE.IBLANK) GO TO 50
0045                IF(INLIST(1).EQ.0)GO TO 14
0046                IF(EX.EQ.0.)EX=1.
0047                IF(WHY.EQ.0.)WHY=1.
0048                WRITE(6,106)NN,INLIST,KONSTR,EX,WHY
0049                DO 5 I=1,3
0050              5 KONSTR(I) = KONSTR(I) + 1
0051                WTS(1,NN) = EX
```

```
052                WTS(2,NN) = WHY
                C
053             61 CALL TRIPTS(X, Y, INLIST, KONSTR, XX, YY, NEWDIR,
                  1           NEWVTX, NFIT, NEW)
                C
054                DO 10 I = 1,NPTS
055                DO 9 N = 1,NEW
056                IF(ABS(X(I) - XX(N)).GT.REFX)GO TO 9
057                IF(ABS(Y(I) - YY(N)).GT.REFY) GO TO 9
058                ND(I) = ND(I)*NEWDIR(N)
059                NEWVTX(N) = I
060                GO TO 10
061              9 CONTINUE
062             10 CONTINUE
                C
063                DO 12 I = 1,NEW
064                IF(NEWVTX(I).NE.0) GO TO 11
065                NPTS = NPTS + 1
066                X(NPTS) = XX(I)
067                Y(NPTS) = YY(I)
068                NEWVTX(I) = NPTS
069                ND(NPTS) = NEWDIR(I)
070             11 NV = NV + 1
071                NVTX(NV) = NEWVTX(I)
072             12 CONTINUE
073                GO TO 4
                C
074             14 IF(NN.EQ.1) GO TO 50
075                NELMT = NN - 1
076                DO 13 I=1,NPTS
077                IF(ND(I).EQ.1)GO TO 13
078                IF(MOD(ND(I),6).EQ.0)GO TO 6
079                IF(MOD(ND(I),2).EQ.0) ND(I) = 2
080                IF(MOD(ND(I),3).EQ.0) ND(I) = 3
081                GO TO 13
082              6 ND(I) = 4
083             13 CONTINUE
084                DO 15 I=1,4
085             15 NFR(I) = 0
086                DO 16 I=1,NPTS
087             16 NFR(ND(I)) = NFR(ND(I)) + 1
088                NN = 0
089                NFREE = NFR(1)
090                NEL = NFR(1) + NFR(2)
091                NMAG = NEL + NFR(3)
092                DO 22 I=1,NPTS
093                N = ND(I)
094                GO TO (18,19,20,21),N
095             18 NN = NN + 1
096                ND(I) = NN
097                GO TO 22
098             19 NFREE = NFREE + 1
099                ND(I) = NFREE
100                GO TO 22
```

```
L01             20 NEL = NEL + 1
L02                ND(I) = NEL
L03                GO TO 22
L                  NMAG = NMAG + 1
L05                ND(I) = NMAG
L06             22 CONTINUE
L07                DO 23 I=1,NPTS
L08                WORKX(I) = X(I)
L09             23 WORKY(I) = Y(I)
L10                DO 24 I =1,NPTS
.11                X(ND(I)) = WORKX(I)
L12             24 Y(ND(I)) = WORKY(I)
.13                DO 25 I=1,NV
.14             25 NVTX(I) = ND(NVTX(I))
.15                WRITE(6,107) (I,X(I),Y(I),I=1,NPTS)
.16                WRITE(6,108)
.17                NN = -NEW
.18                 NF = NEW - NFIT
.19                DO 30 N  =1,NELMT
.20                NN = NN + NEW
.21                NA = NVTX(1 + NN)
.22                NB = NVTX(NF + NN)
.23                NC = NVTX(NEW + NN)
.24                SIZE = ABS((Y(NB) - Y(NA))*(X(NC) - X(NA))
                  1          - (Y(NC) - Y(NA))*(X(NB) - X(NA)))
.25                DO 26 I = 1,3
.26                COTNG(I,N) = ((X(NC) - X(NA))*(X(NB) - X(NA))
                  1             + (Y(NC) - Y(NA))*(Y(NB) - Y(NA)))/SIZE
.27                J = NA
.28                NA = NB
.29                NB = NC
.30             26 NC = J
.31                AREA(N) = SIZE/2
.32             30 WRITE(6,109) N, (NVTX(NN + I),I=1,NEW)
.33                NFREE = NFR(1)
.34                NEL = NFR(2)
.35                NMAG = NFR(3)
.36             31 READ(5,113) MIND, (IDEL(I),DEL(I),I=1,10)
37                 IF(MIND.NE.IBLANK) GO TO 50
38                 IF(SCALX.NE.0.) IDEL(1) = 1
39                 RETURN
40              49 WRITE(6,112)
41                 STOP
          C
42              50 WRITE(6,110)
43                 GO TO 1
          C
44             100 FORMAT(A1,10A4,3X,2F10.0,I1,3X,I2,8X,I2)
45             101 FORMAT('1'//10X, 10A4//  23X, 'INPUT POINT LIST'//  4X,
                  1   16X, 'NO.', 6X, 'X', 13X, 'Y'// )
46             102 FORMAT(A1, I4, 2F10.0)
47             103 FORMAT(20X, I3, 2G14.6)
48             104 FORMAT('0'//  27X, 'INPUT ELEMENT LIST' // 4X, 'NO.',
                  1       17X,'VERTICIES',9X,'CONSTRAINTS',
```

```
                2      3X, 'CONST-1', 4X, 'CONST-2'//)
149          105 FORMAT(A1,I4,2I5,14X,3I1,28X,2F10.0)
150          106 FORMAT(3X,I3,17X,3I3,13X,3I2,2G13.5)
             107 FORMAT('1'//20X, 'ASSEMBLED POINT LIST'// (2X, I3, 2G13.5,
                1     I16, 2G13.5, I16, 2G13.5))
152          108 FORMAT('0'// 28X, 'ASSEMBLED ELEMENT LIST'//  3X, 'NO.',
                1        25X,'VERTEX NUMBERS'//)
153          109 FORMAT(3X,I3,13X,25I4)
154          110 FORMAT('ODATA ERROR'//)
155          111 FORMAT('OELEMENT ', A4, 'NOT IN LIBRARY'//)
156          112 FORMAT(1H1)
157          113 FORMAT(A1,I1,1X,F4.0,9(1X,I1,1X,F4.0))
158              END
```

TOTAL MEMORY REQUIREMENTS 001222 BYTES

```
0001                SUBROUTINE TRIPTS(X, Y, INLIST, KONSTR, XX, YY, NEWDIR,
                   1           NEWVTX, NFIT, NEW)
0002                DIMENSION X(1), Y(1), INLIST(1), KONSTR(1)
0003                DIMENSION XX(1), YY(1)
0004                DIMENSION NEWDIR(1), NEWVTX(1)
0005                X1 = X(INLIST(1))
0006                Y1 = Y(INLIST(1))
0007                X2 = X(INLIST(2))
0008                Y2 = Y(INLIST(2))
0009                X3 = X(INLIST(3))
0010                Y3 = Y(INLIST(3))
              C
0011                NF1 = NFIT + 1
0012                I = 0
0013                DO 10 IPP = 1,NF1
0014                IP = NF1 - IPP
0015                NIP = IP + 1
0016                DO 10 IQQ = NIP,NF1
0017                IQ = NF1 - IQQ
0018                IR = NF1 - IQ - IP - 1
0019                I = I + 1
0020                XX(I) = (IP * X1 + IQ * X2 + IR * X3) / NFIT
0021                YY(I) = (IP * Y1 + IQ * Y2 + IR * Y3) / NFIT
0022                NEWVTX(I) = 0
0023                NEWDIR(I) = 1
0024                IF(IR.EQ.0)NEWDIR(I) = NEWDIR(I)*KONSTR(1)
0025                IF(IP.EQ.0) NEWDIR(I) = NEWDIR(I)*KONSTR(2)
0026                IF(IQ.EQ.0) NEWDIR(I) = NEWDIR(I)*KONSTR(3)
0027             10 CONTINUE
0028                NEW = (NFIT + 1)*(NFIT + 2)/2
0029                RETURN
0030                END
```

TOTAL MEMORY REQUIREMENTS 0004EE BYTES

```
      SUBROUTINE ASSEMB(V,T,AREA,SMAT,WT,NVTX,NFREE,NEL,NMAG,
     1     NFIT,NPTS,NBIG,NELMT,IEH,DELTA)
      DIMENSION AREA(1),SMAT(1),WT(2,1),NVTX(1),V(1),T(1)
      DIMENSION NPA(4),NMA(4),NP(34),TT(202),U(202),Q(202)
      COMMON /TUQ/ NPA,NMA,NP,TT,U,Q
      IF(DELTA.NE.0) GO TO 2
      NA = NFREE + NEL
      NB = NA
      IF(IEH.EQ.1) NA = NFREE + NMAG
      IHE = MOD(IEH,2) + 1
      DO 1 N = 1,NELMT
      NC = (N - 1)*NPTS
      ND = (N - 1)*NBIG
      DO 1 I = 1,NPTS
      IN = NVTX(I + NC)
      IF(IEH.EQ.2) GO TO 5
      IF(IN.GT.NFREE .AND. IN.LE.NB) GO TO 1
      IF(IN.GT.NB) IN = IN - NEL
    5 DO 1 J = 1,I
      JN = NVTX(J + NC)
      IF(IEH.EQ.2) GO TO 6
      IF(JN.GT.NFREE .AND.JN.LE.NB) GO TO 1
      IF(JN.GT.NB) JN = JN - NEL
    6 IF(IN.GT.NA .OR. JN.GT.NA) GO TO 1
      L = LOCATE(IN,JN)
      L1 = I*(I - 1)/2 + J
      V(L) = V(L) + SMAT(L1 + ND)/WT(IHE,N)
      T(L) = T(L) + TT(L1 + NMA(NFIT))*AREA(N)*WT(IEH,N)
    1 CONTINUE
      RETURN
    2 NA = NFREE + NEL
      NB = NFREE + NEL + NMAG
      NC = 2*NFREE + NEL + NMAG
      NC = NC*(NC + 1)/2
      ND = NA*(NA + 1)/2
      DO 4 N = 1,NELMT
      IEH = (N - 1)*NPTS
      IHE = (N - 1)*NBIG
      COEF = WT(1,N)*WT(2,N) - DELTA**2
      COEF2 = DELTA/(2.*COEF)
      L1 = 0
      DO 4 I = 1,NPTS
      IN = NVTX(I + IEH)
      DO 4 J = 1,I
      L1 = L1 + 1
      JN = NVTX(J + IEH)
      IM = MAX0(IN,JN)
      JM = IN + JN - IM
      IF(IM.GT.NB) GO TO 4
      IN = IM
      JN = JM
      IF(IN.GT.NA) IM = IN - NEL
      IF(JN.GT.NA) JM = JN - NEL
      L = IN*(IN - 1) /2 + JN
```

```
)54                LD = ND + NA*IM + IM*(IM -1)/2 + JM
)55                LUO = ND + NA*(IM - 1) + IM*(IM - 1)/2 + JM
)56                LUI = ND + NA*(JM - 1) + JM*(JM - 1)/2 + IM
)                  L2 = L1 + IHE
)58                L3 = L1 + NMA(NFIT)
)59                IF(L.GT.ND) GO TO 3
)60                V(L) = V(L) + SMAT(L2)*WT(1,N)/COEF
)61                T(L) = T(L) + TT(L3)*AREA(N)*WT(1,N)
)62                IF(LD.GT.NC) GO TO 4
)63                IF(IN.GT.NFREE .AND. IN.LE.NA)GO TO 4
)64                IF(JN.GT.NFREE .AND. JN.LE.NA)GO TO 4
)65              3 V(LD) = V(LD) + SMAT(L2)*WT(2,N)/COEF
)66                T(LD) = T(LD) + TT(L3)*AREA(N)*WT(2,N)
)67                IF(L.GT.ND) GO TO 4
)68                V(LUO) = V(LUO) + U(L3)*COEF2
)69                V(LUI) = V(LUI) - U(L3)*COEF2
)70              4 CONTINUE
)71                RETURN
)72                END
```

TOTAL MEMORY REQUIREMENTS 000A6A BYTES

```
0001                SUBROUTINE SMATRX(SMAT,COTNG,NFIT,NPTS,NBIG,NELMT)
0002                DIMENSION SMAT(1),COTNG(3,1)
0003                DIMENSION NPA(4),NMA(4),NP(34),TT(202),U(202),Q(202)
0004                COMMON /TUQ/ NPA,NMA,NP,TT,U,Q
0005                DO 1 N = 1,NELMT
0006                K = (N - 1)*NBIG
0007                L1 = 0
0008                DO 1 I = 1,NPTS
0009                NPI = NP(I + NPA(NFIT))
0010                DO 1 J = 1,I
0011                NPJ = NP(J + NPA(NFIT))
0012                L1 = L1 + 1
0013                L2 = LOCATE(NPI,NPJ) + NMA(NFIT)
0014                L3 = LOCATE(NP(NPI + NPA(NFIT)),NP(NPJ + NPA(NFIT))) + NMA(NFIT)
0015              1 SMAT(L1 + K) = Q(L1 + NMA(NFIT))*COTNG(1,N) + Q(L2)*COTNG(2,N)
                   1          + Q(L3)*COTNG(3,N)
0016                RETURN
0017                END
```

TOTAL MEMORY REQUIREMENTS 00036C BYTES

```
0001                  SUBROUTINE TNORM(E,H,AREA,WT,NVTX,NFIT,NPTS,NBIG,
                     1         NTOTPT,NELMT,NMODES)
0002                  DIMENSION E(15,1),H(15,1),AREA(1),WT(2,1),NVTX(1)
0003                  DIMENSION SMAT(1),COTNG(3,1)
0004                  DIMENSION NPA(4),NMA(4),NP(34),T(202),U(202),Q(202)
0005                  COMMON /TUQ/ NPA,NMA,NP,T,U,Q
0006                  DOUBLE PRECISION ENORM,HNORM,EDIV,HDIV
0007                  DO 4 M = 1,NMODES
0008                  EDIV = 0.
0009                  HDIV = 0.
0010                  K = -NPTS
0011                  DO 3 N=1,NELMT
0012                  ENORM = 0.
0013                  HNORM = 0.
0014                  K = K + NPTS
0015                  L = NMA(NFIT)
0016                  DO 2 I = 1,NPTS
0017                  N1 = NVTX(I + K)
0018                  DO 2 J = 1,I
0019                  N2 = NVTX(J + K)
0020                  L = L + 1
0021                  ENORM = ENORM + E(M,N1)*E(M,N2)*T(L)
0022                 2 HNORM = HNORM + H(M,N1)*H(M,N2)*T(L)
0023                  L = NMA(NFIT)
0024                  DO 5 J = 2,NPTS
0025                  N2 = NVTX(J + K)
0026                  L = L + 1
0027                  J1 = J - 1
0028                  DO 5 I = 1,J1
0029                  N1 = NVTX(I + K)
0030                  L = L + 1
0031                  ENORM = ENORM + E(M,N1)*E(M,N2)*T(L)
0032                 5 HNORM = HNORM + H(M,N1)*H(M,N2)*T(L)
0033                  EDIV = EDIV + ENORM*AREA(N)*WT(1,N)
0034                 3 HDIV = HDIV + HNORM*AREA(N)*WT(2,N)
0035                  EDIV = DSQRT(DABS(1./EDIV))
0036                  HDIV = DSQRT(DABS(1./HDIV))
0037                  DO 4 I = 1,NTOTPT
0038                  E(M,I) = EDIV*E(M,I)
0039                 4 H(M,I) = HDIV*H(M,I)
0040                  RETURN
0041                  END
```

TOTAL MEMORY REQUIREMENTS 00067E BYTES

```
0001                SUBROUTINE VEE(V,E,H,SMAT,WT,NVTX,NFIT,NPTS,NBIG,
                   1       NELMT,NMODES,DELTA)
0002                DIMENSION E(15,1),H(15,1),WT(2,1),NVTX(1),V(1),SMAT(1)
0003                DIMENSION COEF(150),COEF2(150)
0004                DIMENSION NPA(4),NMA(4),NP(34),TT(202),U(202),Q(202)
0005                COMMON COEF, COEF2
0006                COMMON /TUQ/ NPA,NMA,NP,TT,U,Q
0007                DOUBLE PRECISION VE,VM,VUO,VUI,TE,TM,TUO,TUI
0008                DO 5 N = 1,NELMT
0009                COEF(N) = WT(1,N)*WT(2,N) - DELTA**2
0010              5 COEF2(N) = DELTA/(2.*COEF(N))
0011                ND = NMODES*(NMODES + 1)/2
0012                DO 4 M1 = 1,NMODES
0013                DO 4 M2 = 1,M1
0014                LP = LOCATE(M1,M2)
0015                LD = ND + M1*NMODES + LP
0016                LUO = ND + (M1 -1)*NMODES + M1*(M1 -1)/2 +M2
0017                LUI = ND + (M2 -1) *NMODES + M2*(M2 - 1)/2 + M1
0018                TE = 0.
0019                TM = 0.
0020                TUO = 0.
0021                TUI = 0.
0022                K = -NPTS
0023                KT = -NBIG
0024                DO 3 N = 1,NELMT
0025                VE = 0.
0026                VM = 0.
0027                VUO = 0.
0028                VUI = 0.
0029                K = K + NPTS
0030                KT = KT + NBIG
0031                LU = NMA(NFIT)
0032                LT = KT
0033                DO 1 I = 1,NPTS
0034                N1 = NVTX(I + K)
0035                DO 1 J = 1,I
0036                N2 = NVTX(J + K)
0037                LU = LU + 1
0038                LT = LT + 1
0039                VE = VE + E(M1,N1)*E(M2,N2)*SMAT(LT)
0040                VM = VM + H(M1,N1)*H(M2,N2)*SMAT(LT)
0041                VUO = VUO + H(M1,N1)*E(M2,N2)*U(LU)
0042              1 VUI = VUI + H(M2,N1)*E(M1,N2)*U(LU)
0043                LU = NMA(NFIT)
0044                LT = KT
0045                DO 2 J = 2,NPTS
0046                N2 = NVTX(J + K)
0047                LU = LU + 1
0048                LT = LT + 1
0049                J1 = J - 1
0050                DO 2 I = 1,J1
0051                N1 = NVTX(I + K)
0052                LU = LU + 1
0053                LT = LT + 1
```

```
0054                VE = VE + E(M1,N1)*E(M2,N2)*SMAT(LT)
0055                VM = VM + H(N1,N1)*H(M2,N2)*SMAT(LT)
0056                VUO = VUO - H(M1,N1)*E(M2,N2)*U(LU)
00                2 VUI = VUI - H(M2,N1)*E(M1,N2)*U(LU)
0058                TE = TE + WT(1,N)*VE/COEF(N)
0059                TM = TM + WT(2,N)*VM/COEF(N)
0060                TUO = TUO + COEF2(N)*VUO
0061              3 TUI = TUI + COEF2(N)*VUI
0062                V(LP) = TE
0063                V(LD) = TM
0064                V(LUO) = TUO
0065              4 V(LUI) = TUI
0066                RETURN
0067                END
```

TOTAL MEMORY REQUIREMENTS 0009B0 BYTES

```
001               BLOCK DATA
002               DIMENSION N(8),NN(34),A(101),B(101),C(101),D(101),E(101),F(101)
003               COMMON /TUQ/N,NN,A,B,C,D,E,F
004               DATA N/0,3,9,19,0,6,27,82/
005               DATA NN/3, 1, 2, 6, 3, 5, 1, 2, 4, 10, 6, 9, 3, 5, 8, 1, 2, 4, 7
                 1     15, 10, 14, 6, 9, 13, 3, 5, 8, 12, 1, 2, 4, 7, 11/
006               DATA    A/ Z402AAAAA, Z40155555, Z402AAAAA, Z40155555, Z40155555
                 * Z402AAAAA, Z3F888888, Z00000000, Z402D82D8, Z00000000, Z4016C16
                 * Z402D82D8, ZBF16C16C, Z00000000, ZBF5B05B0, Z3F888888, ZBF5B05B
                 * Z4016C16C, Z4016C16C, Z00000000, Z402D82D8, ZBF16C16C, ZBF5B05B
                 * Z00000000, ZBF16C16C, Z00000000, Z3F888888, Z3F2E52E5, Z3EAF8AF
                 * Z40149249, Z3EAF8AF8, Z3FA49249, Z40149249, Z00000000, ZBF73333
                 * ZBF524924, Z40149249, Z3F15F15F, Z3F62BE2B, Z3F62BE2B, Z3F62BE2
                 * Z404A0EA0, Z00000000, ZBF524924, ZBF733333, ZBF20EA0E, Z3F62BE2
                 * Z40149249, Z3E6B46B4, Z00000000, Z3F107507, Z3EAF8AF8, Z3F15F15
                 * Z3F107507, Z3F2E52E5, Z3F107507, ZBF524924, ZBF20EA0E, Z3FA4924
                 * Z3F62BE2B, ZBF524924, Z3EAF8AF8, Z40149249, Z3F107507, ZBF20EA0
                 * ZBF524924, ZBF524924, Z3F62BE2B, Z3FA49249, Z00000000, ZBF73333
                 * Z40149249, Z3E6B46B4, Z3F107507, Z00000000, Z3F107507, Z3F15F15
                 * Z3EAF8AF8, Z3E6B46B4, Z00000000, Z3EAF8AF8, Z3F2E52E5, Z3F14F31
                 * Z3EB8EF1D, Z3FB8EF1D, Z3EB8EF1D, Z3F5C778E, Z3FB8EF1D, ZBE5C778
                 * ZBF5C778E, ZBF4559AA, Z3FE4DB1A, Z3EB8EF1D, Z3F5C778E, Z3F5C778
                 * Z3F1BBD77, Z40308B91, ZBE5C778E, ZBF4559AA, ZBF5C778E, Z3E377AE
007               DATA    B/ Z3F1BBD77, Z3FE4DB1A, Z00000000, Z3F377AEF, Z3F24FC9F
                 * ZBF5C778E, Z3F127E4F, Z3E49F93E, Z3FB8EF1D, ZBEB8EF1D, Z3F127E4
                 * ZBF127E4F, Z3F1BBD77, ZBF6EF5DE, ZBF377AEF, Z3F5C778E, Z40308B9
                 * ZBEB8EF1D, ZBF127E4F, Z3F127E4F, ZBF377AEF, ZBF6EF5DE, Z3F1BBD7
                 * ZBF127E4F, ZBF6EF5DE, Z40308B91, Z00000000, Z3F24FC9F, Z3F377AE
                 * Z3E49F93E, Z3F127E4F, ZBF5C778E, Z3F127E4F, ZBF127E4F, Z3F5C778
                 * Z3FB8EF1D, ZBE1F3526, Z00000000, ZBE81742E, ZBE5C778E, ZBEB8EF1
                 * ZBDDDEBBC, Z3EB8EF1D, Z3EB8EF1D, ZBEB8EF1D, ZBE81742E, Z3F14F31
                 * ZBE81742E, Z3F24FC9F, Z3F127E4F, ZBF4559AA, ZBF127E4F, Z3E49F93
                 * Z3F5C778E, Z3F5C778E, Z3F127E4F, Z3F24FC9F, Z3EB8EF1D, Z3FB8EF1
                 * ZBDDDEBBC, Z3E49F93E, Z3E49F93E, Z3E377AEF, ZBF377AEF, Z3E377AE
                 * ZBF4559AA, Z3F1BBD77, Z3F1BBD77, ZBF4559AA, ZBE5C778E, ZBF5C778
                 * Z3FE4DB1A, ZBE81742E, Z3F127E4F, Z3F24FC9F, Z3E49F93E, ZBF127E4
                 * ZBF4559AA, Z3F24FC9F, Z3F127E4F, Z3F5C778E, Z3F5C778E, Z0000000
                 * Z3F377AEF, ZBF5C778E, Z3FB8EF1D, ZBE1F3526, ZBE81742E, Z0000000
                 * ZBDDDEBBC, ZBEB8EF1D, ZBE5C778E, ZBE81742E, ZBEB8EF1D, Z3EB8EF1
                 * Z3EB8EF1D, ZBE1F3526, Z00000000, ZBE5C778E, Z3EB8EF1D, Z3F14F31
008               DATA    C/ Z00000000, Z41100000, Z00000000, ZC1100000, Z41100000
                 * Z00000000, Z00000000, Z41155555, Z00000000, ZC1155555, Z0000000
                 * Z00000000, ZC0555555, Z41155555, Z00000000, Z00000000, Z0000000
                 * Z00000000, Z00000000, Z41155555, Z00000000, Z40555555, Z0000000
                 * ZC1155555, ZC0555555, Z41155555, Z00000000, Z00000000, Z4116CCC
                 * Z00000000, ZC116CCCC, Z00000000, Z00000000, ZC0999999, Z4120666
                 * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z0000000
                 * Z00000000, Z40999999, Z00000000, ZC1206666, Z00000000, Z0000000
                 * Z00000000, Z402CCCCC, ZC0999999, Z00000000, Z4116CCCC, Z0000000
                 * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z0000000
                 * Z00000000, Z00000000, Z4116CCCC, Z00000000, Z00000000, Z0000000
                 * Z00000000, Z00000000, Z00000000, Z00000000, ZC0999999, Z4120666
                 * Z00000000, ZC02CCCCC, Z00000000, Z40999999, Z00000000, Z0000000
                 * ZC116CCCC, Z402CCCCC, ZC0999999, Z4116CCCC, Z00000000, Z0000000
```

```
     * Z4118EC39, Z00000000, ZC118EC39, Z00000000, Z00000000, ZC0D9CD9C
     * Z4123C23C, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z40D9CD9C, Z00000000, ZC123C23C, Z00000000
D    DATA    D/ Z00000000, Z00000000, Z40680680, ZC111566A, Z00000000,
     * Z4123C23C, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, ZC0680680, Z00000000, Z4111566A
     * Z00000000, Z00000000, ZC123C23C, Z00000000, Z00000000, Z00000000
     * Z00000000, ZC01CFC7A, Z40680680, Z00000000, ZC0D9CD9C, Z00000000
     * Z00000000, Z4118EC39, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z4118EC39, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, ZC0D9CD9C, Z4123C23C
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z00000000, Z40680680
     * ZC111566A, Z4123C23C, Z00000000, Z401CFC7A, Z00000000, ZC0680680
     * Z00000000, Z00000000, Z40D9CD9C, Z00000000, Z00000000, Z00000000
     * ZC118EC39, ZC01CFC7A, Z40680680, ZC0D9CD9C, Z4118EC39, Z00000000
)10  DATA    E/ Z00000000, Z00000000, Z40800000, Z00000000, ZC0800000,
     * Z40800000, Z00000000, Z00000000, Z41155555, Z00000000, ZC1155555
     * Z41155555, Z00000000, Z00000000, Z00000000, Z40800000, Z00000000
     * Z00000000, Z00000000, ZC0AAAAAA, Z41155555, Z00000000, Z00000000
     * Z00000000, Z402AAAAA, ZC0AAAAAA, Z40800000, Z00000000, Z00000000
     * Z411B0000, Z00000000, ZC11B0000, Z411B0000, Z00000000, ZC0566666
     * Z40566666, Z411B0000, Z00000000, Z00000000, Z00000000, ZC1206666
     * Z4140CCCC, Z00000000, Z40566666, ZC0566666, Z40566666, ZC1206666
     * Z411B0000, Z00000000, Z3F999999, ZBF999999, Z3F999999, Z00000000
     * ZBF999999, Z406CCCCC, Z00000000, Z00000000, Z00000000, Z00000000
     * Z00000000, Z00000000, ZC0ACCCCC, Z411B0000, Z00000000, Z00000000
     * Z00000000, Z00000000, Z00000000, Z00000000, Z40566666, ZC1159999
     * Z411B0000, Z00000000, ZBF999999, Z3F999999, ZBF999999, Z00000000
     * Z3F999999, ZC0166666, Z40566666, ZC0ACCCCC, Z406CCCCC, Z00000000
     * Z00000000, Z4121976E, Z00000000, ZC121976E, Z4121976E, Z00000000
     * ZC0C30C30, Z40C30C30, Z41273673, Z00000000, Z00000000, Z00000000
     * ZC12D82D8, Z415B05B0, Z00000000, Z40C30C30, ZC0C30C30, Z4064C64C
)11  DATA    F/ ZC12D82D8, Z41273673, Z00000000, Z4056B015, ZC056B015,
     * ZC0A90A90, Z40D00D00, ZC0270270, Z411D41D4, Z00000000, Z00000000
     * Z00000000, Z40680680, ZC0D00D00, Z40680680, ZC1270270, Z415B05B0
     * Z00000000, Z00000000, Z00000000, Z40680680, ZC0D00D00, Z40680680
     * Z40D00D00, ZC1410410, Z415B05B0, Z00000000, ZC056B015, Z4056B015
     * ZC0270270, Z40D00D00, ZC0A90A90, ZC0340340, Z40D00D00, ZC1270270
     * Z411D41D4, Z00000000, ZBFAD602B, Z3FAD602B, Z3FAD602B, ZC015AC05
     * Z3FAD602B, Z40208208, ZC015AC05, ZC015AC05, Z3FAD602B, Z405F7DF7
     * Z00000000, Z00000000, Z00000000, ZC011566A, Z4022ACD5, ZC011566A
     * ZC022ACD5, Z4022ACD5, Z4022ACD5, ZC022ACD5, ZC0A6DFC3, Z411D41D4
     * Z00000000, Z00000000, Z00000000, Z3FD00D00, ZC01A01A0, Z3FD00D00
     * Z401A01A0, ZC01A01A0, ZC01A01A0, Z401A01A0, Z4077BCD1, ZC11F2747
     * Z412F56F5, Z00000000, Z00000000, Z00000000, ZC011566A, Z4022ACD5
     * ZC011566A, ZC022ACD5, Z4022ACD5, Z4022ACD5, ZC022ACD5, ZC03ED942
     * Z41104104, ZC11F2747, Z411D41D4, Z00000000, Z3FAD602B, ZBFAD602B
     * Z3FAD602B, ZC015AC05, Z3FAD602B, Z3FAD602B, ZC015AC05, ZC015AC05
     * Z40208208, Z3FE7E3D3, ZC03ED942, Z4077BCD1, ZC0A6DFC3, Z405F7DF7
     END
```

```
C                                                                       WAVE6
      SUBROUTINE CHOLOW(A, B, N, NONPOS)                                WAVE6
C                                                                       WAVE6
C     THIS SUBROUTINE PERFORMS A CHOLESKI DECOMPOSITION ON THE MATRIX A WAVE6
C     AND STORES THE RESULTING LOWER TRIANGULAR FACTOR IN B.  BOTH MAT- WAVE6
C     RICES ARE OF ORDER N, AND SYMMETRIC, ONLY THE LOWER TRIANGLE ELE- WAVE6
C     MENTS BEING STORED, IN COMPRESSED FORM BY ROWS.  IF A IS NOT POSI-WAVE6
C     TIVE DEFINITE, AN IMMEDIATE RETURN OCCURS, WITH THE ERROR INDICA- WAVE6
C     TOR 'NONPOS' SET TO THE NUMBER OF THE NEGATIVE PIVOTAL ELEMENT.   WAVE6
C                                                                       WAVE6
      DIMENSION A(1), B(1)                                              WAVE6
      NONPOS = 0                                                        WAVE6
C                                                                       WAVE6
C          FORM THE FIRST THREE ELEMENTS                                WAVE6
      IF(A(1).LT.0.) GO TO 24                                           WAVE6
      B(1) = SQRT(A(1))                                                 WAVE6
      IF(N.EQ.1) GO TO 12                                               WAVE6
      B(2) = A(2) / B(1)                                                WAVE6
      SUM = A(3) - B(2)**2                                              WAVE6
      IF(SUM.LT.0.) GO TO 24                                            WAVE6
      B(3) = SQRT(SUM)                                                  WAVE6
      IF(N.EQ.2) GO TO 12                                               WAVE6
      IJ = 3                                                            WAVE6
C                                                                       WAVE6
C          REMAINING ROWS FOLLOW                                        WAVE6
      DO 10 I = 3,N                                                     WAVE6
       II = LOCATE(I,I)                                                 WAVE6
      ISTART = LOCATE(I,1)                                              WAVE6
C                                                                       WAVE6
C               FIRST ELEMENT IN ROW                                    WAVE6
      IJ = IJ + 1                                                       WAVE6
      B(IJ) = A(IJ) / B(1)                                              WAVE6
C                                                                       WAVE6
C               ROW ELEMENTS UP TO DIAGONAL                             WAVE6
      LAST = I - 1                                                      WAVE6
      DO 5 J = 2,LAST                                                   WAVE6
      IJ = IJ + 1                                                       WAVE6
      JJ = LOCATE(J,J)                                                  WAVE6
      JSTART = JJ - J + 1                                               WAVE6
      JEND = JJ - 1                                                     WAVE7
      IK = ISTART                                                       WAVE7
      SUM = 0.                                                          WAVE7
C                                                                       WAVE7
C               FORM ROW PRODUCT SUM                                    WAVE7
      DO 3 K = JSTART,JEND                                              WAVE7
      SUM = SUM + B(K) * B(IK)                                          WAVE7
    3 IK = IK + 1                                                       WAVE7
C                                                                       WAVE7
C               SET VALUE OF B(IJ)                                      WAVE7
      B(IJ) = (A(IJ) - SUM) / B(JJ)                                     WAVE7
    5 CONTINUE                                                          WAVE7
C                                                                       WAVE7
C               FORM DIAGONAL ELEMENT                                   WAVE7
      LAST = II - 1                                                     WAVE7
```

```
      IJ = IJ + 1                                                    WAVE71
      SUM = 0.                                                       WAVE71
      DO 7 K = ISTART,LAST                                           WAVE71
    7 SUM = SUM + B(K)**2                                            WAVE71
      SUM = A(IJ) - SUM                                              WAVE71
      IF(SUM.LT.0.) GO TO 24                                         WAVE72
      B(IJ) = SQRT(SUM)                                              WAVE72
   10 CONTINUE                                                       WAVE72
   12 RETURN                                                         WAVE72
   24 NONPOS = IJ                                                    WAVE72
      GO TO 12                                                       WAVE72
      END                                                            WAVE72
```

MORY REQUIREMENTS 0004AC BYTES

```
C                                                                       WAVE72
      SUBROUTINE INVLOW(A,N)                                            WAVE72
C     THIS SUBROUTINE REPLACES THE LOWER TRIANGULAR MATRIX A OF ORDER N WAVE73
C     BY ITS INVERSE.  A IS STORED BY ROWS UP TO THE DIAGONAL ELEMENT.  WAVE73
C                                                                       WAVE73
      DIMENSION A(1)                                                    WAVE73
      A(1) = 1. / A(1)                                                  WAVE73
      DO 3 M = 2,N                                                      WAVE73
      M1 = M - 1                                                        WAVE73
      MO = LOCATE(M,0)                                                  WAVE73
      MM = MO + M                                                       WAVE73
      DO 2 K = 1,M1                                                     WAVE73
      SUM = 0.                                                          WAVE74
      IK = LOCATE(K,K)                                                  WAVE74
      DO 1 I = K,M1                                                     WAVE74
      MI = MO + I                                                       WAVE74
      SUM = SUM + A(MI) * A(IK)                                         WAVE74
      AMI = A(MI)                                                       WAVE74
      AIK = A(IK)                                                       WAVE74
    1 IK = IK + I                                                       WAVE74
      MK = MO + K                                                       WAVE74
    2 A(MK) = - SUM / A(MM)                                             WAVE74
      MK = MK + 1                                                       WAVE75
    3 A(MK) = 1. / A(MK)                                                WAVE75
      RETURN                                                            WAVE75
      END                                                               WAVE75
```

ORY REQUIREMENTS 0002DA BYTES

```
C                                                                       WAVE75
      SUBROUTINE GAGT(Z, G, A, N)                                       WAVE75
C                                                                       WAVE75
C     THIS SUBROUTINE PERFORMS THE MATRIX MULTIPLICATION Z = G * A * G',WAVE75
C     WHERE Z AND A ARE SYMMETRIC MATRICES OF ORDER N, AND G IS A LOWER WAVE75
C     TRIANGULAR MATRIX.  ALL THREE ARE STORED IN COMPRESSED FASHION BY WAVE76
C     ROWS OF THE LOWER TRIANGLE.                                       WAVE76
C                                                                       WAVE76
      DIMENSION Z(1), A(1), G(1)                                        WAVE76
      DIMENSION RKJ(115)                                                WAVE76
C                                                                       WAVE76
      DO 7 J = 1,N                                                      WAVE76
      JO = LOCATE(J, 0)                                                 WAVE76
C                                                                       WAVE76
      DO 6 K = 1,N                                                      WAVE76
      KO = LOCATE(K, 0)                                                 WAVE77
C                                                                       WAVE77
      TERM = 0.0                                                        WAVE77
      M = MINO(J,K)                                                     WAVE77
      DO 1 L = 1,M                                                      WAVE77
      KL = KO + L                                                       WAVE77
      JL = JO + L                                                       WAVE77
    1 TERM = TERM + A(KL) * G(JL)                                       WAVE77
      IF(M.EQ.J) GO TO 5                                                WAVE77
      K1 = K + 1                                                        WAVE77
      KL = KL + K                                                       WAVE78
      DO 4 L = K1,J                                                     WAVE78
      JL = JO + L                                                       WAVE78
      TERM = TERM + A(KL) * G(JL)                                       WAVE78
    4 KL = KL + L                                                       WAVE78
    5 CONTINUE                                                          WAVE78
    6 RKJ(K) = TERM                                                     WAVE78
C                                                                       WAVE78
      DO 3 I = J,N                                                      WAVE78
      IO = LOCATE(I, 0)                                                 WAVE78
      TERM = 0.0                                                        WAVE79
      DO 2 K = 1,I                                                      WAVE79
      IK = IO + K                                                       WAVE79
    2 TERM = TERM + G(IK) * RKJ(K)                                      WAVE79
      IJ = IO + J                                                       WAVE79
    3 Z(IJ) = TERM                                                      WAVE79
    7 CONTINUE                                                          WAVE79
C                                                                       WAVE79
      RETURN                                                            WAVE79
      END                                                               WAVE79
```

ORY REQUIREMENTS 0005BE BYTES

```
      SUBROUTINE TRIDIA(A, DIAG, SUBD, N)                                   WAVE80
C                                                                           WAVE80
C     THE SYMMETRIC MATRIX A, IN COMPRESSED STORAGE BY ROWS OF ITS LOW-     WAVE80
C     ER TRIANGLE, IS TRIDIAGONALISED USING THE HOUSEHOLDER METHOD.  THE    WAVE80
C     DIAGONAL ELEMENTS ARE  RETURNED IN ARRAY 'DIAG', THE SUBDIAGONAL      WAVE80
C     ELEMENTS IN THE FIRST N-1 LOCATIONS OF THE ARRAY 'SUBD'.  THE AR-     WAVE80
C     RAY A IS REPLACED  BY THE HOUSEHOLDER TRANSFORMATION VECTOR COM-      WAVE80
C     PONENTS ON RETURNING.                                                 WAVE80
C                                                                           WAVE80
      DIMENSION A(1), DIAG(1), SUBD(1)                                      WAVE81
C                                                                           WAVE81
C          TRIDIAGONALIZE IN REVERSED SEQUENCE                              WAVE81
      DO 7 MK = 3,N                                                         WAVE81
      K = N + 3 - MK                                                        WAVE81
      K1 = K - 1                                                            WAVE81
C                                                                           WAVE81
C          FIND ROW SUM OF SQUARES 'SIGMA'                                  WAVE81
      SIGMA = 0.0                                                           WAVE81
      LOW = LOCATE(K,0)                                                     WAVE81
      LIMT = LOCATE(K,K1)                                                   WAVE82
      LOWR = LOW + 1                                                        WAVE82
      DO 1 J = LOWR,LIMT                                                    WAVE82
    1 SIGMA = SIGMA + A(J)**2                                               WAVE82
C                                                                           WAVE82
C          FIND SUBDIAGONAL ELEMENT                                         WAVE82
      AK = A(LIMT)                                                          WAVE82
      SK = SIGN(SQRT(SIGMA), AK)                                            WAVE82
      SUBD(K1) = - SK                                                       WAVE82
C                                                                           WAVE82
C          FIND H AND CHECK FOR ZERO SUM                                    WAVE83
      IF(SIGMA.EQ.0.0) GO TO 7                                              WAVE83
      H = SIGMA + SK * AK                                                   WAVE83
C                                                                           WAVE83
C          FORM VECTOR U IN ROW K LOCATIONS OF A                            WAVE83
      A(LIMT) = SK + AK                                                     WAVE83
C                                                                           WAVE83
C          FORM VECTOR P IN LOCATIONS OF ARRAY DIAG                         WAVE83
C              ONLY THE NECESSARY K-1 LEADING COMPO-                        WAVE83
C              NENTS ARE CALCULATED.                                        WAVE83
      DO 3 I = 1,K                                                          WAVE84
      SUM = 0.0                                                             WAVE84
      IO = LOCATE(I,0)                                                      WAVE84
C                                                                           WAVE84
C               SUM ALONG ROW, UP TO DIAGONAL ELEMENT                       WAVE84
      DO 2 J = 1,I                                                          WAVE84
      IJ = IO + J                                                           WAVE84
      KJ = LOW + J                                                          
    2 SUM = SUM + A(IJ) * A(KJ)                                             WAVE84
      IF(I.GE.K1) GO TO 3                                                   WAVE84
C                                                                           WAVE85
C          SUM DOWN I'TH COLUMN                                             WAVE85
      I1 = I + 1                                                            WAVE85
      JI = LOCATE(I,I) + I                                                  WAVE85
      DO 9 J = I1,K1                                                        WAVE85
```

```
      KJ = LOW + J                                                      WAVE85
      SUM = SUM + A(JI) * A(KJ)                                         WAVE85
    9 JI = JI + J                                                       WAVE85
    3 DIAG(I) = SUM / H                                                 WAVE85
C                                                                       WAVE85
C           FIND BIGK                                                   WAVE86
      BIGK = 0.0                                                        WAVE86
      DO 4 J = 1,K1                                                     WAVE86
      KJ = LOW + J                                                      WAVE86
    4 BIGK = BIGK + A(KJ) * DIAG(J)                                     WAVE86
      BIGK = BIGK / (2.0 * H)                                           WAVE86
C                                                                       WAVE86
C           FORM VECTOR Q IN LOCATIONS OF ARRAY DIAG                    WAVE86
      DO 5 J = 1,K1                                                     WAVE86
      KJ = LOW + J                                                      WAVE86
    5 DIAG(J) = DIAG(J) - BIGK * A(KJ)                                  WAVE87
C                                                                       WAVE87
C           CALCULATE THE REDUCED MATRIX A                              WAVE87
      DO 6 I = 1,K1                                                     WAVE87
      IO = LOCATE(I, 0)                                                 WAVE87
      KI = LOW + I                                                      WAVE87
      DO 6 J = 1,I                                                      WAVE87
      IJ = IO + J                                                       WAVE87
      KJ = LOW + J                                                      WAVE87
    6 A(IJ) = A(IJ) - A(KI) * DIAG(J) - A(KJ) * DIAG(I)                 WAVE87
    7 CONTINUE                                                          WAVE88
C                                                                       WAVE88
C           COMPUTE OUTPUT ARRAYS                                       WAVE88
      DO 8 I = 1,N                                                      WAVE88
      J = LOCATE(I,I)                                                   WAVE88
    8 DIAG(I) = A(J)                                                    WAVE88
      SUBD(1) = A(2)                                                    WAVE88
      SUBD(N) = 0.0                                                     WAVE88
C                                                                       WAVE88
      RETURN                                                            WAVE89
      END                                                               WAVE89
```

MORY REQUIREMENTS 0006A0 BYTES

```
      SUBROUTINE BISLCT(DIAG, SUBD, N, NLEAST, NEIGV, EVALU)                WAVE89
C                                                                           WAVE89
C     RETURNS 'NEIGV' EIGENVALUES, STARTING FROM 'NLEAST', OF THE           WAVE89
C     SYMMETRIC TRIDIAGONAL MATRIX OF ORDER N, WHOSE PRINCIPAL AND          WAVE89
C     SUBDIAGONAL ARE STORED IN ARRAYS 'DIAG' AND 'SUBD' RESPECTIVELY.      WAVE89
C     THE EIGENVALUES ARE RETURNED IN ARRAY 'EVALU'.  THEY ARE ACCURATE     WAVE89
C     TO ABOUT EIGHT SIGNIFICANT FIGURES.                                   WAVE89
C                                                                           WAVE89
      DIMENSION DIAG(1), SUBD(1), EVALU(1)                                  WAVE90
C                                                                           WAVE90
C          FIND ROW NORM OF MATRIX                                          WAVE90
      RONORM = ABS(DIAG(1)) + ABS(SUBD(1))                                  WAVE90
      DO 1 K = 2,N                                                          WAVE90
    1 RONORM = AMAX1( RONORM, ABS(DIAG(K)) + ABS(SUBD(K)) + ABS(SUBD(K-1WAVE90
     1  ))                                                                 WAVE90
C                                                                           WAVE90
C          SQUARE SUBDIAGONAL ELEMENTS TO SAVE ARITHMETIC                   WAVE90
      DO 10 K = 1,N                                                         WAVE90
      SUBD(K) = SIGN(SUBD(K)**2, SUBD(K))                                   WAVE91
   10 IF(SUBD(K).EQ.0.0) SUBD(K) = 1.E-14 * RONORM**2                       WAVE91
C                                                                           WAVE91
C          FIND 'NEIGV' EIGENVALUES, STARTING WITH ROW NORM BOUNDS          WAVE91
      CALL STURM(DIAG,SUBD,N,0.0,NSIGN)
      NLEAST = NSIGN + 1
      IF(NLEAST.GT.N) GO TO 5
      MOST = MINO(N,NLEAST + NEIGV - 1)
      NEIGV = MOST - NLEAST + 1
      K = 0                                                                 WAVE91
      DO 4 I = NLEAST, MOST                                                 WAVE91
      K = K + 1                                                             WAVE91
      UPPERX = RONORM                                                       WAVE91
      SMALLX = - RONORM                                                     WAVE91
C                                                                           WAVE92
C          FIND I'TH EIGENVALUE                                             WAVE92
C                                                                           WAVE92
C          30 BISECTIONS FOLLOW TO REDUCE ERROR BY 2**30                    WAVE92
      DO 3 J = 1,30                                                         WAVE92
      TRIALX = (UPPERX + SMALLX) / 2.0                                      WAVE92
      CALL STURM(DIAG, SUBD, N, TRIALX, NSIGN)                              WAVE92
      IF(NSIGN.GE.I) GO TO 2                                                WAVE92
      SMALLX = TRIALX                                                       WAVE92
      GO TO 3                                                               WAVE92
    2 UPPERX = TRIALX                                                       WAVE93
    3 CONTINUE                                                              WAVE93
C                                                                           WAVE93
C          STORE EIGENVALUE                                                 WAVE93
    4 EVALU(K) = (UPPERX + SMALLX) / 2.0                                    WAVE93
C                                                                           WAVE93
C          RESTORE SUBDIAGONAL ELEMENTS                                     WAVE93
    5 CONTINUE
      DO 11 K = 1,N                                                         WAVE93
   11 SUBD(K) = SIGN(SQRT(ABS(SUBD(K))), SUBD(K))                           WAVE93
      RETURN                                                                WAVE93
      END                                                                   WAVE94
```

```
      SUBROUTINE WIELND(DIAG, SUBD, N, EVALU, X)                        WAVE94
C                                                                       WAVE94
C     THIS SUBROUTINE DETERMINES THE EIGENVECTOR 'X' CORRESPONDING TO   WAVE94
C     THE EIGENVALUE 'EVALU' OF THE SYMMETRIC TRIDIAGONAL MATRIX OF OR- WAVE94
C     DER N WHOSE PRINCIPAL AND SUBSIDIARY DIAGONALS APPEAR IN 'DIAG'   WAVE94
C     AND 'SUBD' RESPECTIVELY.  TWO PASSES OF WIELANDT ITERATION, USING WAVE94
C     GAUSSIAN DECOMPOSITION WITH INTERCHANGES, RETURN EIGENVECTORS OF  WAVE94
C     GOOD ACCURACY, BUT NOT GENERALLY ORTHOGONAL IF MULTIPLE EIGENVA-  WAVE94
C     LUES ARE ENCOUNTERED.                                             WAVE95
C                                                                       WAVE95
      DIMENSION DIAG(1), SUBD(1), X(1)                                  WAVE95
      DIMENSION DIA1(110), DIA2(110), DIA3(110), FACTR(110), INTER(110) WAVE95
      LOGICAL FINISH, INTER                                             WAVE95
C                                                                       WAVE95
      ABSMAX(A, B) = SIGN(AMAX1(ABS(A), B), A)                          WAVE95
C                                                                       WAVE95
C          SET UP STARTING VALUES                                       WAVE95
      RONORM = ABS(DIAG(1)) + ABS(SUBD(1))                              WAVE95
      DO 1 I = 2,N                                                      WAVE96
    1 RONORM = AMAX1(RONORM, ABS(SUBD(I-1)) + ABS(DIAG(I)) +            WAVE96
     1           ABS(SUBD(I)))                                          WAVE96
      TINY = ABS(RONORM) * 1.E-8                                        WAVE96
      U = DIAG(1) - EVALU                                               WAVE96
      V = ABSMAX(SUBD(1), TINY)                                         WAVE96
C                                                                       WAVE96
C          TRIANGULAR DECOMPOSITION NOW PRODUCES 2 SUPERDIAGONALS       WAVE96
      N1 = N - 1                                                        WAVE96
      DO 3 I = 1,N1                                                     WAVE96
      I1 = I + 1                                                        WAVE97
C                                                                       WAVE97
C          DECIDE WHETEHER INTERCHANGE IS REQUIRED                      WAVE97
      SUB = ABSMAX(SUBD(I), TINY)                                       WAVE97
      SUBNEX = ABSMAX(SUBD(I1), TINY)                                   WAVE97
      INTER(I) = ABS(SUB).GT.ABS(U)                                     WAVE97
      IF(INTER(I)) GO TO 2                                              WAVE97
C                                                                       WAVE97
C          DECOMPOSITION WITHOUT INTERCHANGE                            WAVE97
      FACTR(I1) = SUB / U                                               WAVE97
      DIA1(I) = U                                                       WAVE98
      DIA2(I) = V                                                       WAVE98
      DIA3(I) = 0.                                                      WAVE98
      U = DIAG(I1) - EVALU - V * FACTR(I1)                              WAVE98
      V = SUBNEX                                                        WAVE98
      GO TO 3                                                           WAVE98
C                                                                       WAVE98
C          DECOMPOSITION WITH INTERCHANGE REQUIRED                      WAVE98
    2 FACTR(I1) = U / SUB                                               WAVE98
      DIA1(I) = SUB                                                     WAVE98
      DIA2(I) = DIAG(I1) - EVALU                                        WAVE99
      DIA3(I) = SUBNEX                                                  WAVE99
      U = V - FACTR(I1) * DIA2(I)                                       WAVE99
      V = - FACTR(I1) * SUBNEX                                          WAVE99
    3 CONTINUE                                                          WAVE99
      FINISH = .FALSE.                                                  WAVE99
```

```
      DIA1(N)=U                                                          WAVE99
C           SET UP INITIAL ITERATE                                       WAVE99
      DO 4 I = 1,N                                                       WAVE99
    4 X(I) = 1.                                                          WAVE99
C                                                                        WAV100
C           PERFORM BACK SUBSTITUTION                                    WAV100
    9 X(N) = X(N) / ABSMAX(DIA1(N), TINY)                                WAV100
      X(N-1) = (X(N-1) - DIA2(N-1) * X(N)) / ABSMAX(DIA1(N-1), TINY)     WAV100
      H = X(N)**2 + X(N-1)**2                                            WAV100
      DO 5 KI = 2,N1                                                     WAV100
      I = N - KI                                                         WAV100
      X(I) = (X(I) - DIA2(I)*X(I+1) - DIA3(I)*X(I+2)) / ABSMAX(DIA1(I),  WAV100
     1       TINY)                                                       WAV100
    5 H = H + X(I)**2                                                    WAV100
      H = SQRT(FLOAT(N) / H)                                             WAV101
C                                                                        WAV101
C           SCALE EIGENVECTOR                                            WAV101
      DO 6 I = 1,N                                                       WAV101
    6 X(I) = H * X(I)                                                    WAV101
      IF(FINISH) RETURN                                                  WAV101
C                                                                        WAV101
C           FORWARD SUBSTITUTION                                         WAV101
      DO 8 I = 2,N                                                       WAV101
      I1 = I - 1                                                         WAV101
      IF(INTER(I)) GO TO 7                                               WAV102
      X(I) = X(I) - FACTR(I) * X(I1)                                     WAV102
      GO TO 8                                                            WAV102
    7 U = X(I1)                                                          WAV102
      X(I1) = X(I)                                                       WAV102
      X(I) = U - FACTR(I) * X(I1)                                        WAV102
    8 CONTINUE                                                           WAV102
      FINISH = .TRUE.                                                    WAV102
      GO TO 9                                                            WAV102
      END                                                                WAV102
```

IORY REQUIREMENTS 00104E BYTES

```
C                                                                       WAV110
      SUBROUTINE STURM( DIAG, SUBSQ, N, TRIALX, NEIGN)                  WAV110
C                                                                       WAV110
C     THIS SUBROUTINE DETERMINES THE NUMBER NEIGN OF EIGENVALUES SMAL-  WAV110
C     LER THAN 'TRIALX' OF THE TRIDIAGONAL MATRIX OF ORDER N WHOSE DIA- WAV110
C     GONAL TERMS AND SQUARES OF SUBDIAGONAL TERMS APPEAR IN 'DIAG' AND WAV110
C     'SUBSQ' RESPECTIVELY.  THE MODIFIED STURM SEQUENCE METHOD IS      WAV111
C     USED.                                                             WAV111
C                                                                       WAV111
      DIMENSION DIAG(1), SUBSQ(1)                                       WAV111
C                                                                       WAV111
      EPS = 1.E-8                                                       WAV111
      NEIGN = 0                                                         WAV111
      Q = DIAG(1) - TRIALX                                              WAV111
      IF(Q.LT.0.) NEIGN = 1                                             WAV111
      IF(Q.EQ.0.) Q = ABS(DIAG(1) * EPS)                                WAV111
C                                                                       WAV112
      DO 3 I = 2,N                                                      WAV112
      Q = DIAG(I) - TRIALX - ABS(SUBSQ(I-1))/Q                          WAV112
      IF(Q) 2, 1, 3                                                     WAV112
    1 Q = ABS(DIAG(I-1) * EPS)                                          WAV112
      GO TO 3                                                           WAV112
    2 NEIGN = NEIGN + 1                                                 WAV112
    3 CONTINUE                                                          WAV112
C                                                                       WAV112
      RETURN                                                            WAV112
      END                                                               WAV113
```

IORY REQUIREMENTS 000282 BYTES

```
      SUBROUTINE REVERS(A, SUBD, X, N)                                  WAV103
C                                                                       WAV103
C     THIS SUBROUTINE PERFORMS THE INVERSE HOUSEHOLDER TRANSFORMATION   WAV103
C     ON AN EIGENVECTOR 'X' SO AS TO OBTAIN THE CORRESPONDING EIGEN-    WAV103
C     VECTOR OF THE ORIGINAL SYMMETRIC MATRIX OF ORDER N.               WAV103
C                                                                       WAV103
      DIMENSION A(1) , X(1), SUBD(1)                                    WAV103
C                                                                       WAV103
      DO 3 K = 3,N                                                      WAV103
      K1 = K - 1                                                        WAV104
C                                                                       WAV104
C          FIND THE PRODUCT U'*X                                        WAV104
      SUM = 0.                                                          WAV104
      KO = LOCATE(K,0)                                                  WAV104
      DO 1 J = 1,K1                                                     WAV104
      KJ = KO + J                                                       WAV104
      IF(A(KJ).EQ.0.)A(KJ)=1.E-8
    1 SUM = SUM + X(J) * A(KJ)                                          WAV104
      SUM = - SUM / (SUBD(K1) * A(KJ))                                  WAV104
C                                                                       WAV104
C          DETERMINE TRANSFORMED X                                      WAV105
      DO 2 J = 1,K1                                                     WAV105
      KJ = KO + J                                                       WAV105
    2 X(J) = X(J) - SUM * A(KJ)                                         WAV105
    3 CONTINUE                                                          WAV105
C                                                                       WAV105
C          SCALE VECTOR TO LENGTH N                                     WAV105
      SUM = 0.                                                          WAV105
      DO 4 J = 1,N                                                      WAV105
    4 SUM = SUM + X(J)**2                                               WAV105
      SUM = SQRT(FLOAT(N) / SUM)                                        WAV106
      DO 5 J = 1,N                                                      WAV106
    5 X(J) = X(J) * SUM                                                 WAV106
      RETURN                                                            WAV106
      END                                                               WAV106
```

1ORY REQUIREMENTS 00038E BYTES

```
C                                                                       WAV1066
      SUBROUTINE TRIMUL(Y, A, Z, N)                                     WAV1067
C                                                                       WAV1068
C     THIS SUBROUTINE PERFORMS THE  MULTIPLICATION Y = A' * Z, WHERE Y  WAV1069
C     AND Z ARE COLUMN MATRICES OF ORDER N, AND A IS LOWER TRIANGULAR INWAV1070
C     COMPACTED STORAGE.    Y MAY OVERWRITE Z.                          WAV1071
C                                                                       WAV1072
      DIMENSION Y(1), Z(1), A(1)                                        WAV1073
C                                                                       WAV1074
      DO 2 I = 1,N                                                      WAV1075
      KI = LOCATE(I,I)                                                  WAV1076
      SUM = 0.                                                          WAV1077
      DO 1 K = I,N                                                      WAV1078
      SUM = SUM + A(KI)    * Z(K)                                       WAV1079
    1 KI = KI + K                                                       WAV1080
    2 Y(I) = SUM                                                        WAV1081
      RETURN                                                            WAV1082
      END                                                               WAV1083
```

IORY REQUIREMENTS 000220 BYTES

```
      FUNCTION LOCATE(I, J)                                                  WAV1132
C                                                                            WAV1133
C     THIS FUNCTION LOCATES THE POSITION IN LINEAR STORAGE OF THE (I,J)      WAV1134
C     ELEMENT OF A SYMMETRIC MATRIX, STORED BY ROWS OF THE LOWER OR CO-      WAV1135
C     LUMNS OF ITS UPPER TRIANGLE.                                           WAV1136
C                                                                            WAV1137
      MJ = MAXO(I, J)                                                        WAV1138
      MI = MINO(I, J)                                                        WAV1139
      LOCATE = (MJ * (MJ - 1)) / 2 + MI                                      WAV1140
      RETURN                                                                 WAV1141
      END                                                                    WAV1142
```

MORY REQUIREMENTS 00019E BYTES