John L. Volakis
Rad. Lab., EECS Dept.
University of Michigan
Ann Arbor, MI 48109-2122
(734) 647-1797
(734) 647-2106 (Fax)
volakis@umich.edu (email)

David B. Davidson
Dept. E&E Engineering
University of Stellencosch
Stellenbosch 7600, South Africa
(+27) 21 808 4458
(+27) 21 808 4981 (Fax)
davidson@firga.sun.ac.za (e-mail)

This month's contribution comes from one of the co-Editors of the column, David Davidson. It represents a summary of a number of issues that impact on the implementation of an FEM code using vector elements, and it is based on experience accumulated during the development of such a code. (The paper also includes some results computed with this code). As well as addressing some very practical issues in pre- and post-processing, the paper also includes some useful insights on vector element theory.

Implementation Issues for Three-Dimensional Vector FEM Programs

David B. Davidson

Department of Electrical and Electronic Engineering
University of Stellenbosch, Stellenbosch 7600, South Africa
Tel: +27 21 808 4458; Fax: +27 21 808 4981; E-mail: davidson@ing.sun.ac.za

Abstract

Theoretical and practical issues that impact on the development of (especially) three-dimensional vector finite-element (FEM) programs are discussed. The theory of vector elements is briefly reviewed, and some troublesome aspects are highlighted: in particular, the interpretation of the degrees of freedom. The choice of element shape is briefly addressed, as is the evaluation of the elemental matrices. Several useful references are provided in this context. Data structures appropriate for a three-dimensional code are outlined, as is the issue of establishing mesh interconnectivity. The topics of mesh generation and linear algebra are very briefly considered. The paper concludes with some comments on post-processing, in particular, the computation of fields within the FE mesh.

Keywords: Finite element methods; programming; vector finite elements; edge elements

1. Introduction

The Finite-Element Method (FEM) has become a popular tool for high-frequency electromagnetic-field simulation, in particular, for antennas, microwave circuits, and scattering applications. However, compared with its main differential-equation-based contender, the Finite-Difference Time-Domain Method, the FEM is rather more challenging to implement. The aim of this paper is to "smooth the path" for those intending to implement their own FEM codes by sharing some of the experiences accumulated whilst developing FEM codes: the author and his students are in the process of developing a three-dimensional vector FEM code, incorporating higher-order elements, in *FORTRAN 90*. The focus of this paper is on practical coding issues, but some subtle theoretical issues that impact practically on a code will also be discussed. On occasion, some points that, by and large, are only hinted at in the literature will be highlighted.

Although the FEM is also widely used for low-frequency quasi-static applications (machines, etc.), this paper is focused on the high-frequency (i.e., radio-frequency) applications, which are of primary interest to the *Magazine*'s readership.

2. Vector (Edge-Based) Finite Elements

2.1 Background and Literature

Electromagnetic finite-element analysis has been revolutionized by the development of vector finite elements over the last two decades. Originally known as "edge elements"–because the unknowns were associated with edges, rather than nodes–for the lowest-order elements, at least, the term "vector elements" is now the more-widespread current nomenclature. (This is due to the introduction of higher-order elements, where the unknowns are no longer assigned solely along edges). There are a number of excellent treatments of the theory of vector finite elements; all the modern textbooks and monographs on the FEM for high-frequency CEM address this issue [1, 2, 3, 4, 5]. The original paper by Nedelec remains formidable reading [6], and one should be aware that not all the higher-order vector elements published satisfy the criteria in Nedelec's paper, although such elements may, nonetheless, work satisfactorily.

For further reading, an excellent annotated collection of papers is available [7], and a comprehensive bibliography has been published [8]. The former contains a number of classic papers, but is starting to show its age in some aspects, due to continuing progress in the field since its publication. In particular, there has been significant progress on higher-order vector elements, exemplified by [9].

For the purpose of this paper, we will consider only vector FEs; the consensus in the HF CEM community is that these are the best elements to use for general-purpose three-dimensional electromagnetic codes. Webb's paper [10] remains a classic on vector elements, explaining both the fundamentals of the elements, as well as the theoretical reasons for their superior performance in terms of the improved approximation of the null-space of the vector wave equation. (It also addressed some practical implications, such as improved modeling of corners.) This interpretation is now generally accepted in the high-frequency FEM community. Peterson and Wilton's contribution to the volume edited by Itoh et al. [11] is an excellent treatment of vector elements. They consider firstly polynomial-complete vector elements, showing that one can formulate and use such elements. They then discuss the crucial contribution made by Nedelec in removing certain "wasted" degrees of freedom, which result in the mixed-order (incomplete) vector elements in general usage in the FE community.

Note that in two dimensions–for instance, for waveguiding problems–homogeneously filled waveguides (supporting only TE_z or TM_z modes) can be worked using scalar elements, involving only longitudinal (z-directed) electric or magnetic field, or alternately analyzed with two-dimensional vector elements. Inhomogeneously filled waveguides can support more complex modes; a suitable approach here is to use vector elements for the transverse fields, and conventional scalar Lagrangian elements for the longitudinal field. More details specifically on FE analysis for waveguide analysis may be found in [12].

2.2 Vector Elements and Nedelec's Work

Degrees of freedom are central to a FE program. However, almost all the papers in the CEM literature on vector elements concentrate on the *basis functions*, with the degrees of freedom discussed only briefly, if it all. Especially in this context, one struggles to reconcile modern CEM papers with Nedelec's seminal work [6]. This paper is very difficult reading indeed, and unless one has a strong background in functional analysis, one would be advised to read it in conjunction with the comprehensive treatment of the material by Salazar-Palma and her colleagues [5]. Their monograph provides the mathematical background necessary to fully appreciate Nedelec's work.

Some comments here are appropriate. Firstly, Nedelec's paper derives the *mathematical requirements* to be satisfied by mixed-order elements. However, nowhere in this paper (nor in his subsequent work [13]) is the actual element-interpolation function explicitly proposed–and certainly not using simplex coordinates, as almost universally done by FEM researchers. Nedelec's work instead concentrates on the appropriate degrees of freedom, and the dimension of the accompanying mixed-order polynomial spaces. What has become widely know as the Whitney element for simplex elements (triangles or tetrahedrons)–the $\left(\lambda_i \nabla \lambda_j - \lambda_j \nabla \lambda_i\right)$ form–is indeed a Nedelec-compliant element, but this is far from clear after an initial reading of his work!

The degrees of freedom as laid down by Nedelec are not unique, even for the lowest-order (Whitney) element above. This is rather cryptically implied in Definition 4 [6]: for "*k*th" order elements, the $6k$ edge-based degrees of freedom for three-dimensional elements ($3k$ in two dimensions) are given by

$$\int_a \vec{u} \cdot \hat{t}\, ds\,,\ \forall q \in \mathrm{P}_{k-1}\,,$$

where $\vec{u}$ is a basis function and $\hat{t}$ is the unit vector along edge a. P_k is the linear space of polynomials of degree $\leq k$. For the Whitney element, with $k = 1$, we see that q may only be a constant. In the case of the $\left(\lambda_i \nabla \lambda_j - \lambda_j \nabla \lambda_i\right)$ form, this constant is often implicitly unity, and the associated Nedelec degree of freedom (which may be viewed as located at the middle of the relevant edge, although this is not essential) is the tangential field on this edge, multiplied by the edge length. (It may be shown that the integral of the Whitney element along an edge is constant). *Note that in an FE code, the unknowns which are solved for–usually also called degrees of freedom–are simply the constants associated with each basis function.* Depending on the choice of q above, these may be precisely Nedelec's "degrees of freedom," or related by a constant.

Many authors pre-normalize the basis functions by the appropriate edge lengths: e.g., $w_{ij} = \ell_{ij}\left(\lambda_i \nabla \lambda_j - \lambda_j \nabla \lambda_i\right)$. Other choices of constant are also permissible. Either is quite acceptable, but one *must* work consistently: if the edge-lengths are included in the basis functions, then when post-processing is done, the lengths must be correctly included again in the basis functions. We return to this subsequently.

An exception to the simplex-based approach is the work of Salazar-Palma and her colleagues [5]. Their approach starts directly with Nedelec's definitions of the various degrees of freedom, using a right-angled "parent" triangle element in Cartesian coordinates. Interpolatory vector functions are then proposed, with unknown coefficients that are then derived by explicitly enforcing the standard Lagrangian interpolatory requirement that the function be non-zero only at its "own" node. (This requires the solution of a system of linear equations, but this need only be done once for each element type, and tabulated results are available [5]). This approach is elegant theoretically. It does, however, limit one to interpolatory elements. The work of Yioultsis and Tsiboukis also follows this approach [14].

Higher-order Nedelec-complaint elements are certainly not unique, and the differences are more than just in terms of constants as above. There are two classes of such elements, viz., interpolatory and hierarchical. The former have the advantage that the degree of freedom can be associated directly with a field component (possibly within a constant, however); the latter have the advantage that additional degrees of freedom may be added directly to the lower-order contributions. (In terms of mesh adaptation, this is also known as "enriching the element.") The hierarchical elements for simplex elements used by many researchers are generally derived by "inspection," relying on the natural separation of the field into normal and tangential field components when the basis function is expressed in terms of simplex coordinates and the gradients thereof. The result has been the publication of a large number of different higher-order elements. Some of the most recent are due to Andersen and Volakis [15, 16]. These elements appear attractive, and were favorably reviewed by other researchers at the recent "5th Finite Elements Workshop for Microwave Engineering" (Boston, June, 2000).

2.3 Coding Implications of Vector Elements

The above discussion may appear overly theoretical. However, as soon as one includes higher-order elements in an FE code, one appreciates that an understanding of the nature of the degrees of freedom is important. In the following, we use constant tangential/linear normal (CT/LN) for the Whitney elements (also known as H_0 (curl) elements), linear tangential/quadratic normal (LT/QN) for the H_1 (curl) elements, etc. For CT/LN elements, the degrees of freedom have already been discussed above, and all that is necessary is to ensure that adjoining edges correctly share the degree of freedom. In a typical code, this is taken into account when assembling the system matrix from the elemental matrices. This must take into account the *direction* of the edge. A method is discussed later in this paper that ensures that this is always consistent; otherwise, one must simply keep track of the relative positive or negative sense of each degree of freedom. For LT/QN elements, each edge has an additional degree of freedom, and the same applies. There are also two degrees of freedom per face. Again, one simply enforces continuity when assembling the matrix; similar comments with respect to the sense of the faces apply. For higher-order elements, there are also volume-centered degrees of freedom. These are not shared by adjoining elements, and thus do not impact on the matrix-assembly process.

3. Element Shape

The question of element *shape* is not a cut-and-dried issue. In three dimensions, the choices include tetrahedral, brick, hexahedral, and prismatic elements. The reason is that different element shapes are optimal for addressing different problems. At the risk of over-simplifying the issue, a one-off code designed for a particular purpose, with a regular geometry, will be much quicker to develop using brick elements, without any loss of accuracy, but obviously with a loss of generality. Tetrahedral elements ("tets") involve more work, but have the major advantage of being the only shape into which an arbitrary geometrical region can reliably be decomposed: as such, for a general-purpose code, or one likely to evolve into such a code, tets are the element of choice. Prismatic elements (essentially extruded triangles) are especially appropriate for problems generated by extruded surfaces. A good example here is the FEM/MoM treatment of conformal antennas, where prismatic elements have been widely used [17,18]. Pyramidal elements are useful especially in transition regions in brick meshes (e.g., between finer and coarser meshes).

4. Evaluating the Elemental FE Matrices

Irrespective of whether the variational-functional or Galerkin-weighted residual formulations are used, two integrals of two energy-related quantities over the volume (or over the surface, for two-dimensional analysis) of each element are required:

$$S_{kl}^{e} = \int_{V} (\nabla \times \vec{w}_k \cdot \nabla \times \vec{w}_l)\, dV , \qquad (1)$$

$$T_{kl}^{e} = \int_{V} (\vec{w}_k \cdot \vec{w}_l)\, dV , \qquad (2)$$

with $\vec{w}_k$ and $\vec{w}_l$ being the vector basis functions. These matrices have various names, including the *Dirichlet* matrix and *metric*, respectively (and also stiffness and mass matrices, although these latter names are not relevant in electromagnetics). Closed-form results are available for these integrals over tets, bricks, and prisms in the standard texts; for tets, there are also some very useful formulae in papers by Lee and Mittra [19] and Savage and Peterson [20]. The latter was extended by Davidson and Hansmann [21]. Explicit results for other element shapes may be found in [1, 4, 22]. For element orders exceeding "second" order (linear tangential/quadratic normal), the explicit formulae become extremely cumbersome, and quadrature methods are recommended: see [23].

5. Data Structures

We have already seen that there are some significant choices to be made before a line of code is written. Some choices greatly simplify code development, but at the cost of making subsequent extensions of the code much more difficult. As an example, the choice to "hard-wire" brick elements into a code impacts on the edge-numbering scheme, which, in turn, impacts on the element-assembly process, and also manifests itself in the post-processing step, where the field is finally computed. Some of these matters appear quite simple–for instance, a brick has twelve edges, a tet has six–but once this is hard-wired into the code, correctly changing all the references to the number of edges per element can be difficult, unless it has been correctly coded from the start with this in mind. Furthermore, if one intends to subsequently add higher-order elements, it is wise to plan this from the start, although it considerably complicates the original planning. (Arguably, it may easier to address these issues using an object-oriented language, but the objects themselves must still be carefully thought through).

Before programming starts, it is useful to establish the major data structures that will be needed. For a mesh with N_n nodes, N_e elements, N_{edge} edges, and N_f faces, the major data structures required will include at least:

vertices Dimensioned as $(N_n,3)$. This stores the (x,y,z) coordinates of each vertex (node).

nodes Dimensioned as $(N_e,4)$. This stores the four nodes associated with each element.

element_faces Dimensioned as $(N_e,4)$. These are the global faces associated with each element.

edge_nodes Dimensioned as $(N_{edge},2)$. This stores the global nodes that each edge connects.

face_nodes Dimensioned as $(N_f,3)$. This stores the global nodes comprising each face.

materials Dimensioned as N_e. This stores the material number. Another (usually very much smaller) data-structure will be required to store the actual constitutive parameters for each material.

dof Dimensioned as N_{edge} for Whitney elements. These are the degrees of freedom. For higher-order codes, it is useful to discriminate among the various types: for instance, using the hierarchical scheme of [24, 19], one would have four vectors, viz., dof_e1, dof_e2, dof_f1, and dof_f2, as the degrees of freedom corresponding to the edge and face basis functions, each of two types.

Two major data structures omitted here (deliberately) are the $[S]$ and $[T]$ matrices. To fully exploit the power of the FEM, sparse storage schemes must be used; this is beyond the scope of the present paper.

Before any electromagnetic analysis starts, it is necessary to build lists of edges and faces, as well as inter-element connectivity. Before starting this, it is *essential* to adopt consistent conventions for edge and face numbering, both locally and globally. A convention that the author has used successfully is to sort the nodes in each element into ascending global order: this ensures that when edges are assigned, they are always directed from lower to higher node numbers, and thus the edges shared by two or more elements always have the same vector sense. All the local edge-numbering schemes in use in the literature are consistent [19, 20] (taking into account that some number from 0, and some from 1), although the face-numbering schemes are not [19, 20]. Which one is adopted is of no import, so long as one is consistent. (Note that the vector *sense* of the edges is not always consistent in the literature: for example, edge 5 in [19] has the opposite local sense to [1].)

Building the interconnectivity data is primarily a problem in list searching. Various tricks can be used to accelerate this, such as

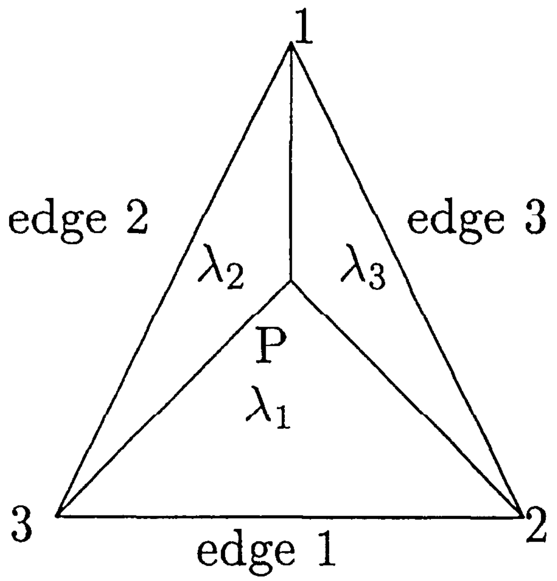

Figure 1a. The simplex coordinates associated with point P within the element.

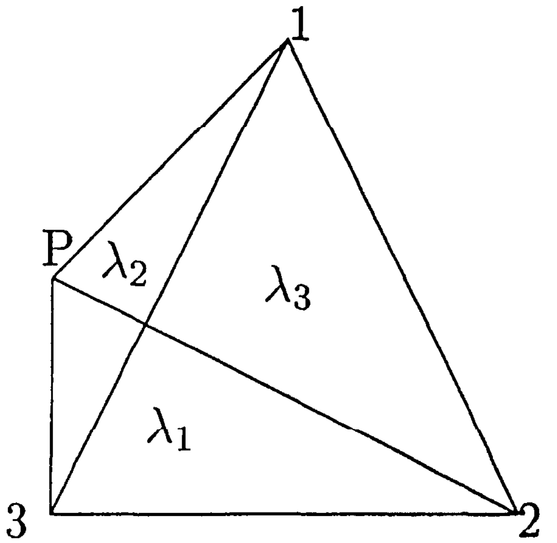

Figure 1b. The simplex coordinates associated with point P outside the element.

the use of "hashing" tables. For instance, when searching for connected elements, one can build a hash table with four entries for each element: each entry corresponds to the sum of the three nodes comprising each of the four faces. If none of another element's hash values is the same, the elements cannot be connected (assuming that none of the node's numbers are negative!). If the hashing entries are identical, then one searches through the nodes of the face to see if they are individually identical, in which case the face–and hence elements–are connected. (Again, adopting a convention of numbering faces consistently from lower to higher global node numbers can be helpful.) The speed-up here comes from the fact that, usually, only a few faces will have identical entries in the hash table (although this obviously depends on the hashing function used), and it is much faster to initially search only four entries in the hash table per element, rather than all twelve nodes. Many commercial meshers can also generate element-interconnectivity lists, but this tends to tie one's code into specific packages.

Some pre-processing algorithms for FE codes have been discussed before in this column: see, for example, [25].

6. Mesh Generation

Brick elements are particularly attractive for research codes, since writing a mesher for specific problems is relatively straightforward. Prismatic meshes, which extrude triangular meshes, are also a candidate for doing-it-yourself, since two-dimensional triangular meshers are of only moderate complexity. However, tetrahedral meshing is *not* a trivial issue, and using a commercial package for this is recommended. There are a number available: the author has used both *FAM* (now *CADFIX*) from *FEGS* in the UK, and *FEMAP*, from Enterprise Software Products in the USA.

7. Linear Algebra

Although the issue of matrix solvers for full matrices continues to attract some attention in the MoM community, the linear-algebra problems are largely solved in this regard, with the possibly exception of the question of iterative solvers. (This does not apply to the issue of fast methods, which is an active and fruitful research area, at present.) The same is not true, however, with regard to the sparse linear algebra required by an FEM code, where many challenges remain. Deterministic (driven) problems require the solution of a sparse system of linear equations. Direct-decomposition methods require techniques for predicting fill-in: the best reference here is [26]. The fill-in is problematic for several reasons. It is necessary to predict (either approximately or exactly) the fill-in pattern; there is a concomitant, and sometimes quite large, increase in memory requirement, and the indirect addressing required makes the algorithms difficult to optimize. (Various mesh re-numbering schemes are available to attempt to minimize fill-in, but indirect addressing cannot be avoided.) In practice, iterative solvers are generally preferred. One place where direct decomposition cannot apparently be avoided in FE analysis is when using the shift-invert mode of eigenvalue solvers, which extracts eigenvalues near a particular value, rather than the smallest or largest ones.

Eigenanalysis problems require the solution of a generalized eigenvalue problem, which is computationally expensive. Work has recently been completed at Stellenbosch on the use of the implicitly restarted Arnouldi technique, implemented within the public-domain *ARPACK* library [27]. For the present, readers are referred to [4, Chapter 9]: of the standard FE texts in electromagnetics, it presently offers the most comprehensive and up-to-date discussion of sparse-linear-algebra issues for the FEM in CEM.

Finally, we comment that for initial testing work, standard full-matrix linear-algebra routines from the *LAPACK* library may be profitably used. Although obviously very inefficient in terms of both memory and run time, this permits the FE part of the code to be debugged without also having to cope simultaneously with the additional bugs that sparse-matrix storage schemes can introduce.

8. Post-Processing

Once the FEA is complete, the vector degrees of freedom need to be post-processed to yield meaningful field data. Unlike interpolatory nodal-based elements, where a degree of freedom typically represents a field component at a particular node, hierarchical vector elements reconstruct a physically meaningful field only when summed together. Given the degrees of freedom and the corresponding basis functions, the field $\vec{E}(x,y,z)$ can be computed at any point within the element. Explicitly for CT/LN elements on tetrahedral, this is

$$\vec{E}(x,y,z) \approx E_{12}w_{12} + E_{13}w_{13} + E_{14}w_{14} + E_{23}w_{23} + E_{24}w_{24} + E_{34}w_{34}, \quad (3)$$

with E_{ij} being the degrees of freedom, and w_{ij} being the basis functions, as previously discussed. Note again here that the lengths must be included–or not!–in w_{ij}, in a fashion consistent with the usage when deriving Equations (1) and (2). (This is obvious, but easy to overlook, since the lengths are often implied but not consistently retained in some of the literature. If the lengths are *not* correctly included in the basis function, the result, on a non-uniform mesh, will be a field with the correct general shape, but with a very "spiky" behavior. The author spent a rather frustrating time a while back tracking this particular error down.)

The basis functions w_{ij} are, of course, functions of position. The simplex coordinates and their gradients must be computed at

Table 1. The eigenmodes (FEM and analytical) for a circular cavity.

Eigenmode (Multiplicity)	Analytical	FEM
TM_{010} (single)	2.405	2.418
TM_{110} (pair)	3.824	3.851
		3.851
TM_{210} (pair)	5.123	5.159
		5.160
TM_{020} (single)	5.507	5.543
TE_{111} (pair)	6.542	6.395
		6.400
TM_{011} (single)	6.734	6.705
TE_{211} (pair)	6.975	6.924
		6.954

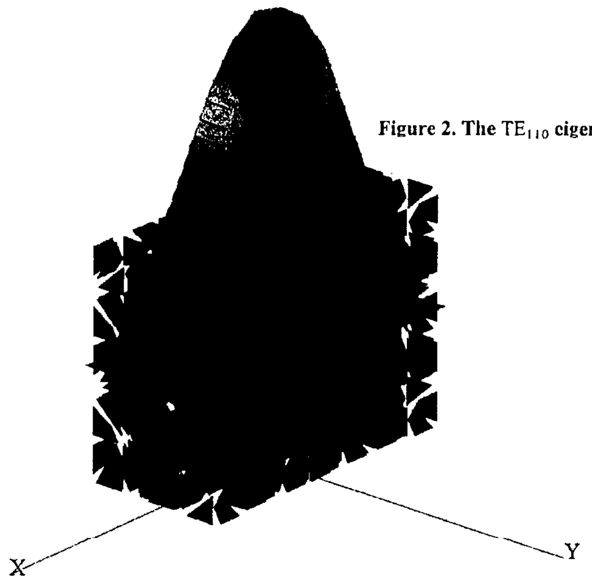

Figure 2. The TE_{110} eigenmode for a rectangular cavity.

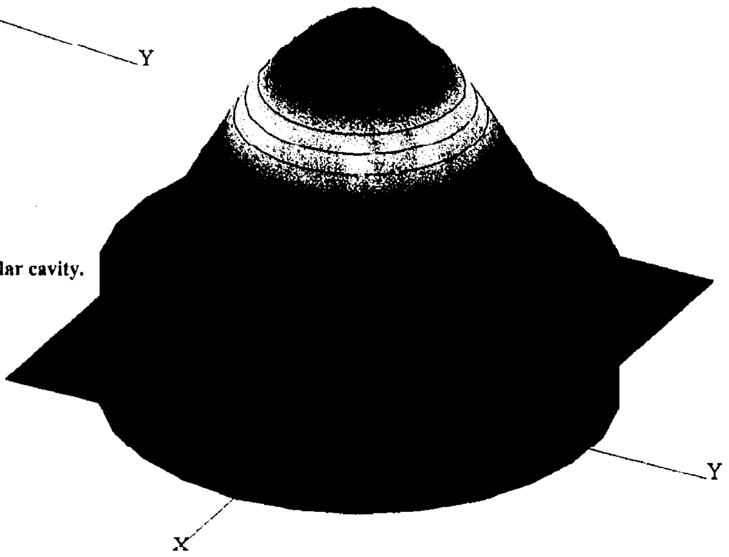

Figure 3. The TM_{010} eigenmode for a circular cavity.

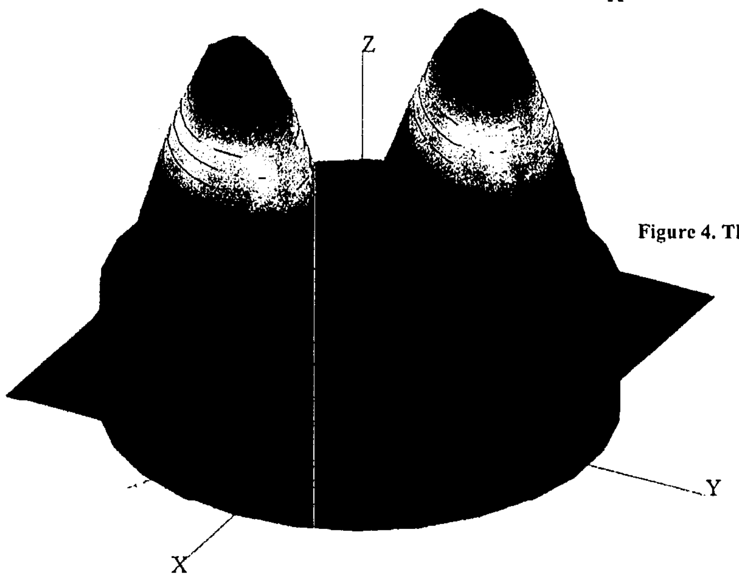

Figure 4. The TM_{110} eigenmode for a circular cavity.

the specified point (x,y,z), a straightforward operation, to compute the $w_{ij}(x,y,z)$ terms. But a problematic geometrical issue emerges in a tetrahedral mesh: does the point (x,y,z) lie within element i? This may need to be evaluated both for a large number of elements per field point, and the search process then repeated for each field point, so a reasonably efficient algorithm is clearly required. The following algorithm has been used to determine this, via a two-step procedure:

- Firstly, check if the cube defined by the extremal coordinates of the nodes of element i contains the point (x,y,z). Mathematically, the requirement is that $x_{\min} \le x \le x_{\max}$ and $y_{\min} \le y \le y_{\max}$ and $z_{\min} \le z \le z_{\max}$. Here, $x_{\min}$ and $x_{\max}$ are the minimum and maximum x coordinates of the nodes; similarly for y and z. This is a very fast check, since nothing need be computed.

- If the point lies within this cube, compute the four simplex coordinates λ_i. Provided these lie in the range $[0,1]$, the point lies within the element.

Proof that all the simplex coordinates must lie in this range for the point to be within the element follows simply from the geometrical property that each simplex coordinate is computed from a determinant that is six times the (signed) volume of the tetrahedron defined by the point in question and three other nodes of the element. (It is not often made clear in the FE literature that the "volume" defined by the determinant, e.g., [2, p. 164], is a *signed* quantity. Even more confusingly, the literature often uses the real volume and this quantity interchangeably. For points inside the element, the signs cancel to produce positive values for the simplex coordinates. The sign depends on the clockwise or counter-clockwise numbering of the nodes.) Usually, the simplex coordinates are normalized by the signed volume of the element, so that $\sum \lambda_i = 1$. If a point lies outside the element, the "extra" volume thus contributing to some of the simplex coordinates must be cancelled by a negative value of at least one other. If the point lies far from the element, at least one simplex coordinate may also exceed 1.

This is demonstrated in two dimensions in Figure 1 for the simplex coordinates associated with a point P, located either inside or outside the element. (In this two-dimensional case, the determinant defines twice the signed area, but otherwise the argument is identical, and rather easier to see in two dimensions than in three). Note that in the latter case, λ_1 and λ_3 between them also comprise the area that defines λ_2; the vertices defining λ_1, λ_2, and λ_3 are $\{2,3,\mathrm{P}\}$, $\{3,\mathrm{P},1\}$, and $\{\mathrm{P},1,2\}$, respectively. Clearly, for $\sum \lambda_i = 1$, λ_2 must be negative to cancel the area already included in λ_1 and λ_3.

There are some additional complexities that must be kept in mind in the actual code. Firstly, the finite precision of the calculations produces a value within ε of 0 (or 1), where ε is the machine precision (this is dependent on both the computer and the choice of single- or double-precision storage); the code needs to take this into account. Secondly, it is quite possible for such field points to lie on the faces between elements. In this case, either the search is stopped as soon as an element is found containing point (x,y,z) and the field on this element is computed; or all elements are searched, and the field computed is the average of all the fields on the faces of the connected elements. Due to the nature of especially the lowest-order constant-tangential/linear-normal basis functions, it is possible for considerable field discontinuities to exist on element boundaries. However, these are not necessarily incorrect, especially in the presence of different materials, where the normal component must indeed be discontinuous, so one must be cautious of simply blindly averaging!

Some visualization results, computed with an FE code (*FEMFEKO* [28]) using the above post-processing algorithms are presented in Figure 2 for the TE_{110} eigenmode in a rectangular cavity [20]. The E_z field component is shown. (Results for the eigenvalues were presented in [20]; the present author's code predicts the eigenvalues with similar accuracy.) The cavity dimensions were 1 m × 0.5 m × 0.75 m, with PEC walls. The FE mesh had 765 tetrahedral elements, with an average edge length of 0.177 m. The results shown used LT/QN elements, with 4162 degrees of freedom.

Results for a circular cavity, with radius 1 m and height 0.5 m and PEC walls, are shown in Figures 3 and 4 for the two lowest eigenmodes, TM_{010} and TM_{110}, respectively. (The latter is degenerate; there is a pair of TM_{110} modes, corresponding to $\cos\phi$ and $\sin\phi$ variation, respectively. This is correctly predicted by the code; only one of these two modes computed is shown here.) Again, the E_z field component is shown. The FE mesh for this example had 497 tetrahedral elements, with an average edge length of 0.323 m. The results shown use LT/QN elements, with 2242 degrees of freedom. It is also important to note that the eigenvalues predicted by the code can be placed in one-to-one correspondence with analytically known results [29, p. 214]; see Table 1. As expected with vector elements, the zero-approximate modes (of which there are 314 for this mesh, the largest with value 0.06802) do not thus corrupt the eigenvalue spectrum of interest.

9. Conclusion

The FEM is a powerful method, but implementing a three-dimensional vector FEM code from scratch is not a trivial matter. To produce a FEM code able to perform only the simplest cavity eigen-analysis requires at least months of effort; fortunately, subsequent extensions can grow in an evolutionary fashion from the initial work. This paper has discussed a number of theoretical and practical issues, which are either not readily available in the literature, or rather obscure. It is hoped that this paper will simplify the task of program development for others wishing to unlock the potential of this method for the first time.

10. Acknowledgments

The work this paper is based on originated during the author's sabbatical visit, in 1997, to Trinity College (TC) and the Cambridge University Engineering Department (CUED), Cambridge, England. The author gratefully acknowledges his hosts during this period, viz., Drs. R. L. Ferrari (TC) and A. C. Metaxas (CUED). Aspects of this paper have been influenced by discussions with many other researchers; the inputs of Drs. D. C. Dibben and D. H. Malan (CUED), Mrs. R. H. Geschke (née Hansmann)

(University of Stellenbosch), and Prof. A. F. Peterson (Georgia Tech) are particularly appreciated.

11. References

1. J. Jin, *The Finite Element Method in Electromagnetics*, New York, John Wiley and Sons, 1993.

2. P. P. Silvester and R. L. Ferrari, *Finite Elements for Electrical Engineers, Third Edition*, Cambridge, Cambridge University Press, 1996.

3. A. F. Peterson, S. L. Ray, and R. Mittra, *Computational Methods for Electromagnetics*, Oxford and New York, Oxford University Press and IEEE Press, 1998.

4. J. Volakis, A. Chatterjee, and L. Kempel, *Finite Element Method for Electromagnetics: Antennas, Microwave Circuits, and Scattering Applications*, Oxford and New York, Oxford University Press and IEEE Press, 1998.

5. M. Salazar-Palma, T. K. Sarkar, L. E. García-Castillo, T. Roy and A. Djordjevi , *Iterative and Self-Adaptive Finite-Elements in Electromagnetic Modelling*, Boston, Artech House, 1998.

6. J. C. Nedelec, "Mixed Finite Elements in $\Re^3$," *Numerische Mathematik*, **35**, 1980, pp. 315-341.

7. P. P. Silvester and G. Pelosi, *Finite Elements for Wave Electromagnetics*, New York, IEEE Press, 1994.

8. R. Coccioli, T. Itoh, G. Pelosi, and P. P. Silvester, "Finite-Element Methods in Microwaves: A Selected Bibliography," *IEEE Antennas and Propagation Magazine*, **38**, December 1996, pp. 34-48.

9. R. D. Graglia, D. R. Wilton, and A. F. Peterson, "Higher Order Interpolatory Vector Bases for Computational Electromagnetics," *IEEE Transactions on Antennas and Propagation*, **AP-45**, March 1997, pp. 329-342.

10. J. P. Webb, "Edge Elements and What They Can Do for You," *IEEE Transactions on Magnectics*, **MAG-29**, March 1993, pp. 1460-1465.

11. A. F. Peterson and D. R. Wilton, "Curl-Conforming Mixed-Order Edge Elements for Discretizing the 2D and 3D Vector Helmholtz Equation," in T. Itoh, G. Pelosi, and P. P. Silvester (eds.), *Finite Element Software for Microwave Engineering*, New York, John Wiley and Sons, 1996, Chapter 5.

12. J. B. Davies, "Complete Modes in Uniform Waveguide," in T. Itoh, G. Pelosi, and P. P. Silvester (eds.), *Finite Element Software for Microwave Engineering*, New York, John Wiley and Sons, 1996, Chapter 1.

13. J. C. Nedelec, "A New Family of Mixed Finite Elements in $\Re^3$," *Numerische Mathematik*, **50**, 1986, pp.57-81.

14. T. V. Yioultsis and T. D. Tsiboukis, "Development and Implementation of Second and Third Order Vector Finite Elements in Various 3-D Electromagnetic Field Problems," *IEEE Transactions on Magnectics*, **MAG-33**, March 1997, pp. 1812-1815.

15. L. S. Andersen and J. L. Volakis, "Hierarchical Tangential Vector Finite Elements for Tetrahedra," *IEEE Microwave and Guided Wave Letters*, **8**, March 1998, pp. 127-129.

16. L. S. Andersen and J. L. Volakis, "Development and Application of a Novel Class of Hierarchical Tangential Vector Finite Elements for Electromagnetics," *IEEE Transactions on Antennas and Propagation*, **AP-47**, January 1999, pp. 112-120.

17. T. Ozdemir and J. L. Volakis, "Triangular Prisms for Edge-Based Vector Finite Element Analysis of Conformal Antennas," *IEEE Transactions on Antennas and Propagation*, **AP-45**, May 1997, pp.788-797.

18 L.C. Kempel, "Implementation of Various Hybrid Finite Element-Boundary Integral Methods: Bricks, Prisms and Tets," *Proceedings of the 15th Annual Review of Progress in Applied Computational Electromagnetics*, March 1999, Monterey, California, pp. 242-249.

19. J-F. Lee and R. Mittra, "A Note on the Application of Edge-Elements for Modeling Three-Dimensional Inhomogeneously-Filled Cavities," *IEEE Transactions on Microwave Theory and Techniques*, **MTT-40**, September 1992, pp. 1767-1773.

20. J. S. Savage and A. F. Peterson, "Higher-Order Vector Finite Elements for Tetrahedral Cells," *IEEE Transactions on Microwave Theory and Techniques*, **MTT-44**, June 1996, pp. 874-879.

21. D. B. Davidson and R. H. Hansmann, "Hierarchical 2D and 3D Vector Finite Elements for Electromagnetic Wave Eigenvalue Problems," *Proceedings of the 15th Annual Review of Progress in Applied Computational Electromagnetics*, March 1999, Monterey, California, pp. 518-521.

22. R. D. Graglia and I. Gheorm, "Higher Order Interpolatory Vector Bases on Pyramidal Elements," *IEEE Transactions on Antennas and Propagation*, **AP-47**, May 1999, pp. 775-782.

23. J. S. Savage and A. F. Peterson, "Quadrature Rules for Numerical Integration over Triangles and Tetrahedra," *IEEE Antennas and Propagation Magazine*, **38**, June, 1996, pp. 100-102.

24. J. S. Savage, "Comparing High Order Vector Basis Functions," *Proceedings of the 14th Annual Review of Progress in Applied Computational Electromagnetics*, March, 1998, Monterey, California, pp. 742-749.

25. D. C. Ross, "Some Finite-Element Preprocessing Algorithms for Electromagnetic Scattering," *IEEE Antennas and Propagation Magazine*, **AP-35**, June, 1993, pp. 68-71.

26. I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford, Oxford University Press, 1986.

27. R. B. Lehoucq, D. C. Sorensen, and C. Yang, "ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods," October 1997; available at http://www.caam.rice.edu/software/ARPACK.

28. D. B. Davidson, R. H. Geschke, and F. J. C. Meyer, "Vector Based Higher-Order 3D Finite Element Simulation of Microwave Cavities," *Proceedings of IEEE AFRICON'99*, Cape Town, South Africa, September 1999, pp. 1053-1058.

29. R. F. Harrington, *Time-Harmonic Electromagnetic Fields*, New York, McGraw-Hill, 1961.