Triangular grid generators for the eigenvalue calculation with edge elements

G.M. Ortigoza Capetillo
Facultad de Ingeniería Universidad Veracruzana,
Calzada Adolfo Ruiz Cortines s/n, Fracc. Costa Verde Boca del Río Ver, México,
Phone (229) 921-61-21, 921-87-55123,
Fax (229) 921-65-32
e-mail:gortigoza@uv.mx

Recibido el 7 de noviembre de 2008; aceptado el 1 de octubre de 2009

In this work we investigate some computational aspects of the eigenvalue calculation with edge elements, those include: the importance of the grid generator and node-edge numbering. As the examples show, the sparse structure of the mass and stiffness matrices is highly influenced by the edge numbering of the different grid generators tested. Significant bandwidth reduction can be obtained by the proper combination of the edge numbering scheme with the grid generator method. Moreover, an ordering algorithm such as the Reverse Cuthill-McKee can improve the bandwidth reduction which is necessary to reduce storage requirements.

Keywords: Triangular grid generators; edge elements; RCM ordering; generalized eigenproblem.

En este trabajo investigamos algunos aspectos computacionales del cálculo de eigenvalores con elementos de borde, entre los que se incluye la importancia del generador de mallas y la numeración de nodos-lados. Como muestran los ejemplos la estructura esparcida de las matrices de masa y momento es altamente influenciada por la numeración de los lados de los diferentes generadores de mallas probados. Una reducción de ancho de banda notable puede obtenerse mediante la combinación apropiada del esquema de numeración de los lados con el método empleado por el generador de mallas. Más aún, una renumeración como el algoritmo Reverse Cuthill-McKee puede mejorar la reducción de ancho de banda lo cual es necesario para reducir requerimientos de almacenamiento.

Descriptores: Generadores de mallas triangulares; elementos de borde; renumeración RCM; eigenproblemas generalizados.

PACS: 02.30.Jr; 02.70.Dh; 41.20.Jb

1. Introduction

In electromagnetism, eigenvalue problems that are often encountered include those of cavity resonance and wave propagation in both closed and open structures, such as metallic waveguides, open and shielded microstrip transmission lines, and optical waveguides or fibers. In these problems, one is interested in determining the resonant frequencies or propagation constants corresponding to eigenvalues and the associated resonant or propagation modes corresponding to eigenvectors.

Calculation of eigenfrequencies in electromagnetic cavities is useful in various applications such as the design of resonators. The importance of the computation of the eigenfrequencies in a cavity resides in the fact that the electromagnetic field within such a cavity can be decomposed into linearly independent modes that oscillate in time at distinct frequencies. The modes are referred to as eigenmodes, the frequencies as eigenfrequencies.

For the resonant frequencies, calculation in simple geometries, analytical techniques [15], scattering matrix formulation [20] and finite differences [12] have been successfully used; however for complex geometries, the geometry can be approximated by a tessellation (provided by grid generators), which makes the finite element method the most appropriate technique.

The finite element method with edge elements has been used to solve these kinds of problems [21]; some of the advantages of edge elements include the facts that: they are divergence free (spurious non-physical solutions are eliminated), interelement boundary conditions are automatically obtained through the natural boundary conditions, edge elements impose the continuity of only the tangential components of the electromagnetic field, and Dirichlet boundary condition can be easily imposed along the edge elements.

Some factors that complicate the finite element solution of the eigenvalue analysis are the sparsity of the matrices and the fact that the method gives rise to generalized eigenproblems $Sv = \lambda Mv$ where only a few selected eigenvalues are desired. The sparse structure of the matrices M and S is highly influenced by the edge numbering provided by the grid generator. Here, sparse matrix techniques are preferable since the storage required increases with $O(N)$, where N denotes the degrees of freedom of the problem. Moreover, storage can be reduced by minimizing the bandwidth of the connectivity matrix.

The work is organized as follows: in Sec. 2 we introduce the finite element formulation for eigenvalue problems in electromagnetism by using edge elements (two dimensional Whitney elements), Sec. 3 shows the influence of the mesh generator in the structure of the mass and stiffness matrices, in Sec. 4 we present the use of the RCM ordering algorithm to reduce the bandwidth of the matrices, Sec. 5 presents the eigenvalue calculation, and finally Sec. 6 present the conclusions of this work.

2. The edge elements for an eigenvalue calculation

Let us consider the eigenvalue calculation of a resonant cavity [8, 21, 23] where the boundary is assumed to be metallic and the interior is air Ω; the eigenvalue problem for the resonant wavenumber is

$$\bigtriangledown \times \bigtriangledown \times E = k^2 E, \quad \text{in} \quad \Omega \tag{1}$$

$$\hat{n} \times E = 0 \quad \text{on} \quad \partial\Omega \tag{2}$$

This eigenvalue problem is obtained by the following assumptions: the electromagnetic field $\hat{E}(x,t)$, $\hat{H}(x,t)$ in the cavity is described by Maxwell equations, the fields are separated into their spatial and temporal components $\hat{E}(x,t) = E(x)e^{i\omega t}$ and $\hat{H}(x,t) = iH(x)e^{i\omega t}$ and the equations are solved for one of the fields [7] (in our case for E).

In order to get the weak formulation, let us multiply Eq. (2) by a vector testing function W_i and integrate over Ω:

$$\int_{\Omega} W_i \cdot \left(\bigtriangledown \times \bigtriangledown \times E - k^2 E \right) dx = 0; \tag{3}$$

now integrating by parts we get

$$\int_{\Omega} (\bigtriangledown \times W_i) \cdot (\bigtriangledown \times E) dx = k^2 \int_{\Omega} W_i \cdot E dx - \int_{\partial\Omega} W_i \cdot (\hat{n} \times \bigtriangledown \times E) ds. \tag{4}$$

For a perfect electric conducting (PEC) boundary, the contour integral vanishes as W_i is set to zero to satisfy the Dirichlet boundary condition. Thus Eq. (4) can be written as

$$\int_{\Omega} (\bigtriangledown \times W_i) \cdot (\bigtriangledown \times E) dx = k^2 \int_{\Omega} W_i \cdot E dx. \tag{5}$$

For general geometries, the cavity is approximated by a triangular tessellation, and in each triangular element the transverse electric field can be expressed as a superposition of edge elements

$$E = \sum_{j=1}^{3} e_j W_j, \tag{6}$$

where $W_j = l_j(L_{j_1} \nabla L_{j_2} - L_{j_2} \nabla L_{j_1})$, L_i is the first order shape function associated with nodes 1,2,3, and l_j is the length of edge j connecting nodes j_1 and j_2.

In order to obtain the finite element formulation we substitute Eqs. (6) into (5) to get

$$\int_{\Delta} \sum_{j=1}^{3} (\bigtriangledown \times W_i) \cdot (\bigtriangledown \times W_j) e_j dx = k^2 \int_{\Delta} \sum_{j=1}^{3} (W_j \cdot W_i) e_j dx \tag{7}$$

for $n = 1, 2, 3$, where Δ denotes integration over the triangular element. By interchanging the integration and summations, Eq. (7) can be written in a matrix formulation

$$[S^l][e] = k^2 [M^l][e] \tag{8}$$

where the local matrices are given by

$$S^l_{ij} = \int_{\Delta} (\bigtriangledown \times W_i) \cdot (\bigtriangledown \times W_j) \tag{9}$$

and

$$M^l_{ij} = \int_{\Delta} W_i \cdot W_j dx; \tag{10}$$

thus after a loop over all the triangles we obtain the global eigen matrix equation

$$[\mathbf{S}][e] = k^2 [\mathbf{M}][e]. \tag{11}$$

3. The influence of the mesh generator

In several applications the geometries are not simple, so it is necessary to discretize the computational domain. Triangles are the most popular simple shapes employed in modelling two-dimensional geometries.

To investigate the influence of the mesh generator we considered six different triangular grid generators: initmesh [17] (pdetool Matlab toolbox), Triangle [22], mesh2d [9], meshnewer [19], Qmg [18] and Ansys [2].

The list of software for mesh generation is very long [16], so we focus our attention on six grid generators based on three main grid generation techniques: Octree, Delaunay and Advancing Front.

Initmesh is a Matlab function that implements a Delaunay triangulation algorithm, Triangle is a mesh generator that uses Ruppert's Delaunay refinement, mesh2d is a 2d constrained Delaunay unstructured mesh generator that provides three different node numbering, meshnewer is based on an iterative continuous smoothing method and Distmesh [13], Qmg uses a quadtree-based algorithm that can triangulate any polyhedral region including nonconvex regions with holes, and finally Ansys uses an advancing front mesh generator.

We begin our discussion with some observations on the sparsity pattern of the stiffness S and mass M matrices. For vector elements the unknowns are associated with the edges of the elements (the number of edges or degrees of freedom can be obtained by Euler's formula with the number of nodes and elements provided by the nodal grid generator). An edge on the boundary of the computational domain has two neighboring edges (the other two that with it form a triangle), while an interior edge has four neighboring edges (an interior edge is shared by two triangles); thus the rows of the stiffness and mass matrices related to interior edges have at most five nonzero entries, while those related to boundary edges have three. A bound for the number of nonzero entries nz is given by $nz = 3 * nbedges + 5 * niedges$, where $nbedges$ and

TABLE I. Meshes information

created by	nodes	elements	edges	size
initmesh	273	496	768	0.1807
Triangle	270	502	771	0.2033
mesh2d	272	506	777	0.1861
meshnewer	308	505	812	0.3043
Qmg	210	370	579	0.2008
Ansys	257	460	716	0.1979

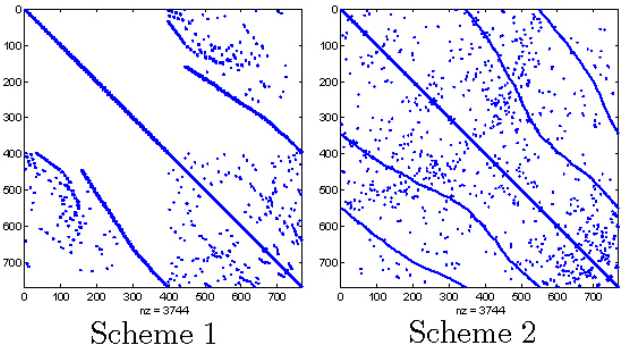

FIGURE 1. Matrix Structure initmesh.

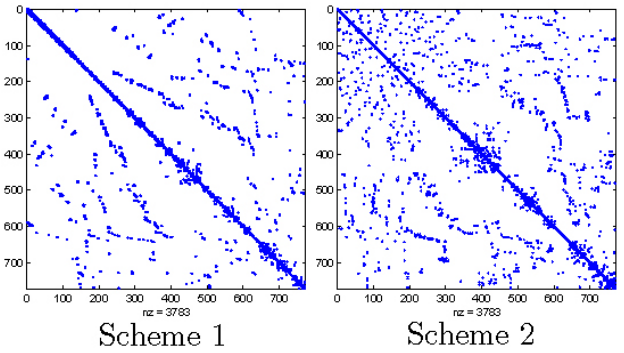

FIGURE 2. Matrix Structure Triangle.

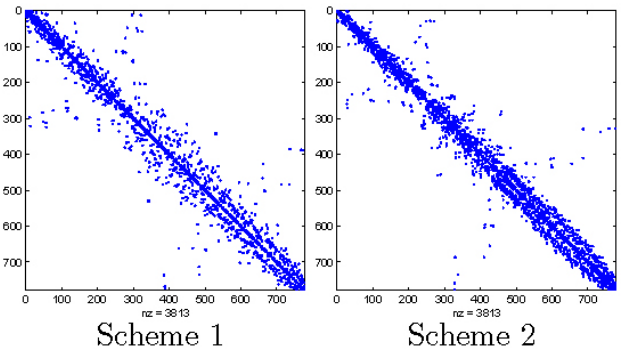

FIGURE 3. Matrix Structure mesh2d.

$niedges$ are the number of boundary and interior edges respectively (some mesh generators as initmesh and Qmg provide a list of the edges on the boundary).

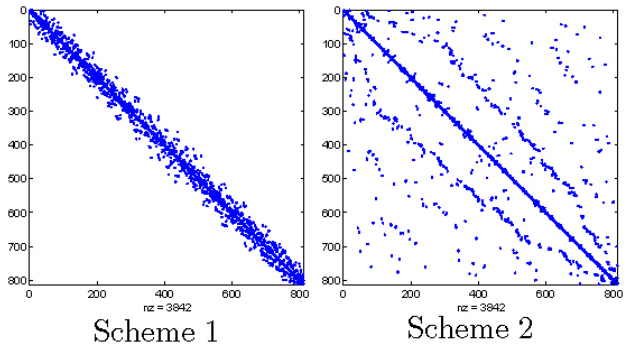

FIGURE 4. Matrix Structure meshnewer.

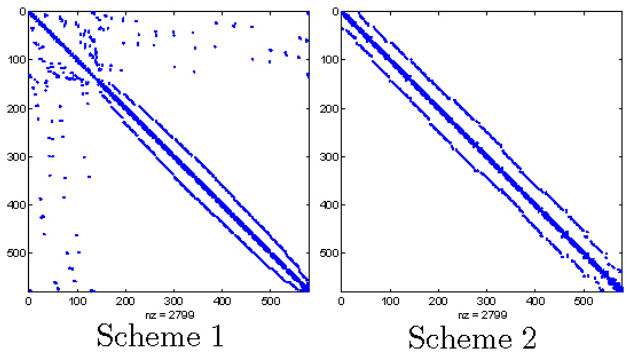

FIGURE 5. Matrix Structure Qmg.

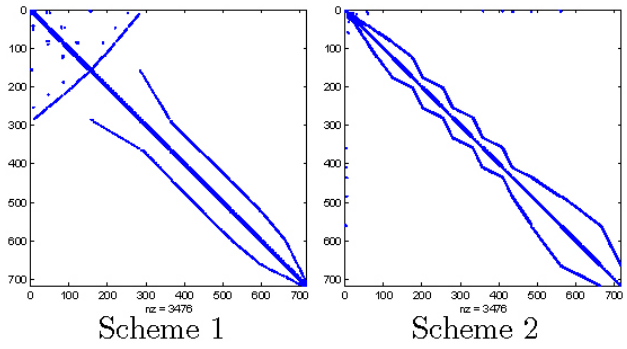

FIGURE 6. Matrix Structure Ansys.

The sparse structure of the matrices **S** and **M** depends on the edge ordering; most of the grid generators do not provide the edge numbering because they were developed for node-based finite elements (among the grid generators tested only Triangle provides the edge numbering), so we need to convert node numbering into edge numbering. Here we follow the two simple schemes given by Jin [8]; for the first scheme denoted by Sc_1, an indicator to each edge is defined and the array of indicators is rearranged by a sorting algorithm (the edge ordering provided by Triangle coincides with the one obtained with this scheme). Most sorting algorithms are very efficient and can perform the task with $nlog(n)$ operations (here n is the number of triangles). For the second scheme Sc_2 no sorting algorithm is required; we use the element-to-node connectivity array to generate a node-to-element array.

Then the element-to-edge array is initialized with zeros and a counter is set to zero; to fill in it we loop over the elements and examine its edges: if the entry is nonzero this edge was already numbered and we go to the next edge, and if it is zero we give it the value of the counter. This algorithm requires n^2 operations. These schemes generate different edge numbering for a given triangulation.

In our first experiment we used the different grid generators to define a grid for the simple geometry of the unit circle with approximately 500 elements. Table I displays the information of the meshes.

By using the two schemes of edge numbering we calculate the stiffness matrix S for the six meshes. Figures 1-6 show the sparse structure of S for the two edge numbering schemes (S and M have basically the same structure).

We refer to bandwidth as the half bandwidth over the number of degrees of freedom. Figure 7 shows bandwidth for the six meshes with the two schemes. Here we notice no significant changes in the bandwidth for the first three meshes with both schemes (all of them obtained by Delaunay triangulation methods); however we note that Sc_1 works better (lower bandwidth) for the meshes generated by Ansys and meshnewer while Sc_2 works better for the mesh generated by Qmg. In fact the bandwidth obtained by using Sc_1 is only 11.48% of that obtained with Sc_2 for meshnewer, and for Ansys the bandwidth obtained with Sc_1 is 48.83% of that obtained with Sc_2, whereas for Qmg we have the opposite situation: the bandwidth obtained with Sc_2 is 9.58% of that obtained with Sc_1. The grid generator meshnewer is based on the iterative method of Persson, which tries to optimize the node locations by a force-based smoothing procedure while the Ansys grid generator uses an advancing front method. It seems that grid generators based on Delaunay methods produce no significant changes in the bandwidth size with either edge numbering scheme. However Sc_1 produces a lower bandwidth than Sc_2 for meshnewer; here the iterative method of Persson gives an optimal node numbering for the Sc_1 which is based on a sorting algorithm. On the other hand Sc_2 produces a lower bandwidth than Sc_1 for Qmg which uses a quadtree method. In this case Sc_2 makes a loop over the elements so it seems that Qmg gives an optimal element ordering. Similar results were observed by testing twenty different geometries, showing that the bandwidth is influenced by the method used for the grid generator [16].

Unstructured grid generators usually create numbers for vertices and cells as they produce them. For a frontal grid generator, the vertices are often numbered in a spiral fashion; for octree methods, squares containing the geometric model are recursively divided until a desired solution is found. Thus nodes and faces are formed whenever the internal octree structure intersects the boundary, whereas Delaunay generators have random numbering.

Figures 8-9 shows the meshes. The meshes produced by the grid generators based in Delaunay methods (initmesh, Triangle, mesh2d) looks very similar; however the meshes produced by meshnewer, Qmg and Ansys look different.

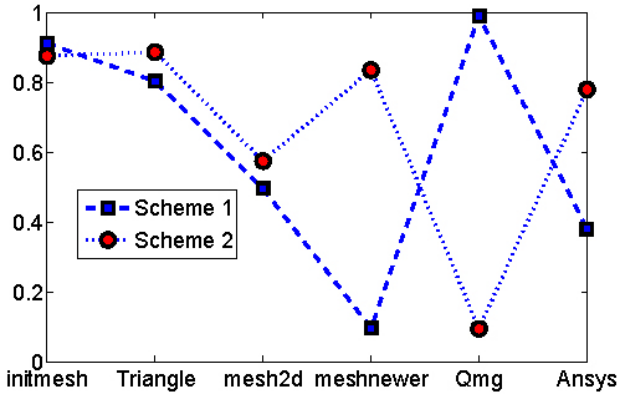

FIGURE 7. Bandwidth.

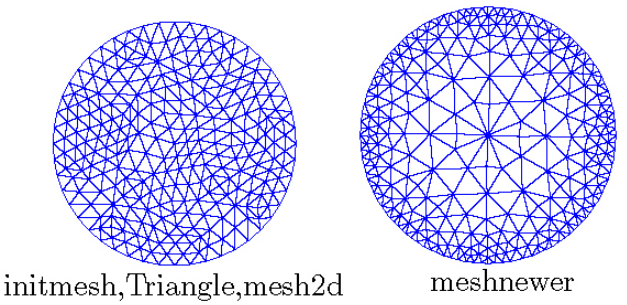

FIGURE 8. Meshes obtained by Delaunay method and continuous smoothing method.

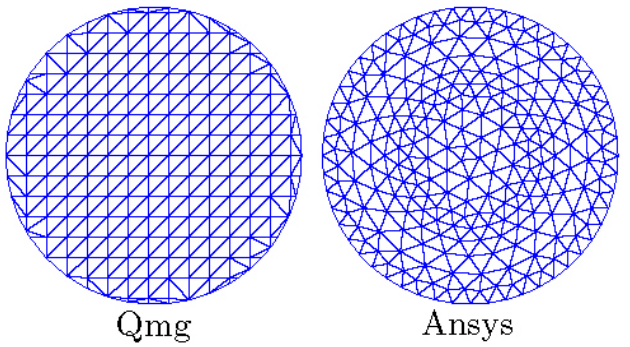

FIGURE 9. Meshes obtained by Quadtree and Advancing Front method.

TABLE II. Grid generators information

		method	E.N.	B.E.	Language
initmesh	C. S.	Delaunay	×	√	Matlab
Triangle	O. S.	Delaunay	√	*	C
mesh2d	O. S.	Delaunay iterative	×	*	C
meshnewer	O. S.	continuous smoothing	×	×	Matlab
Qmg	O. S.	quadtree	×	√	Matlab C++
Ansys	C. S.	advancing front	×	×	user interface

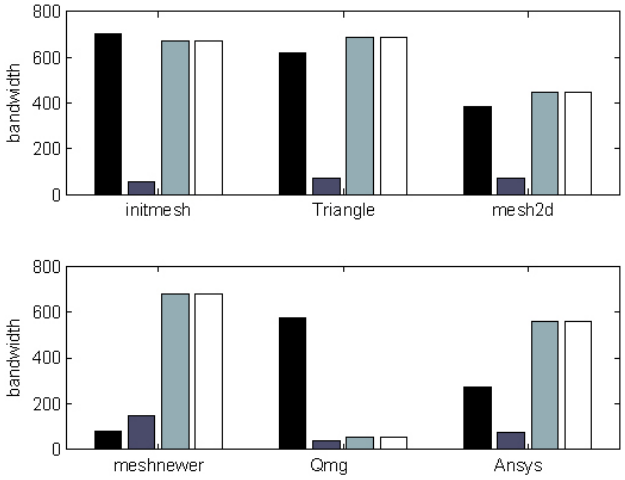

FIGURE 10. Bandwidth reduction by RCM of the meshes.

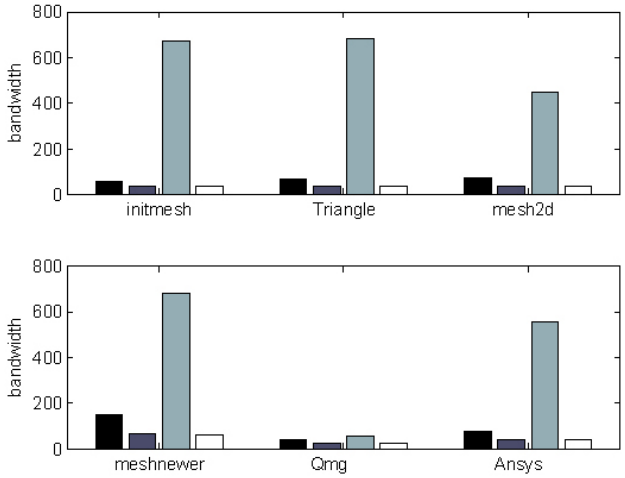

FIGURE 11. Bandwidth reduction by RCM of the matrices.

As we mentioned above, the finite element formulation with edge elements requires the edge numbering to assemble the matrices and the boundary edges to imposed boundary conditions. Table II summarizes some useful information of the grid generators. Here E.N. and B.E. mean Edge Numbering and Boundary Edges respectively; C.S. stands for commercial software while O.S. stands for Opensource software. Among them, only Triangle provides the edge numbering, initmesh and Qmg are the only grid generators that provide the boundary edges, while mesh2d can put marks on the boundary nodes so that boundary edges can be identified.

4. Reordering

Reordering of sparse matrices is essential for good performance on parallel computers. A good reordering algorithm can lead to much better load balance of the computer and thus to a dramatic increase in performance compared to a naive ordering [3]. In order to reduce the bandwidth of the stiffness and mass matrices, an ordering scheme can be used. Nodal ordering for the formation of suitable sparsity patters for the finite element matrices are often performed using graph theory [10].

A widely used but rather simple ordering algorithm is the reverse Cuthill-McKee ordering algorithm [4]. This algorithm first finds a *pseudoperipheral vertex* of the graph of the matrix. It then generates a *level structure* by *breadth first search* and orders the vertices by decreasing distance from the *pseudoperipheral* vertex.

Here we use RCM with two approaches: in the first one the ordering is applied to the graph of the mesh (the nodes and elements) and then we assemble the matrices. In the second one we can assemble the matrices and use the RCM to reorder the rows and columns of the matrices (the eigenvalues remain invariant); a Matlab implementation of this ordering is provided by the function `symrcm`. It is desirable that the grid generator can provided optimal meshes, so the RCM should be considered as part of the grid generator. This is not the case with these grid generators, so we assemble the matrices and reorder them before solving the generalized eigenproblem.

4.1. Reordering the meshes

As we mentioned we generate a mesh, apply the RCM algorithm and then we assemble the matrices. By the nature of the edge ordering schemes, we expect to obtain better results by using Sc_1 after the RCM ordering.

Figure 10 shows the bandwidth reduction produced by the RCM algorithm. At each group the height of the columns represents the bandwidth: the first one is obtained by using Sc_1, the second one is RCM followed by Sc_1, the third is Sc_2, and the fourth is RCM followed by Sc_2. Bandwidth reduction is attained with all grid generators when a RCM followed by Sc_1 is used except with meshnewer; it seems that the node ordering of the mesh generated by meshnewer is optimal and a RCM reordering is not needed. Note that even though a RCM ordering was used, the Sc_2 does not provide bandwidth reduction.

4.2. Reordering the matrices

In this case the RCM ordering is applied after the matrices are assembled. Figure 11 shows the bandwidth reduction by using RCM to the meshes (rcm1) and to the assembled matrices (rmc2). In each group the height of the columns represents the bandwidth; the first column is obtained by rcm1 with scheme 1, the second column is rcm2 with scheme 1, the third column is rcm1 with scheme 2 and the fourth one is rcm2 with scheme 2.

In all cases bandwidth reduction is obtained by rcm2; Sc_2 does not work very well, since it only provides bandwidth reduction with rcm1 for the Qmg grid generator. The rcm ordering is not necessarily the best choice, as it produces a matrix with a narrow bandwidth which fills in almost completely during the Cholesky factorization. On the other Minimum degree [1] produces a structure with large blocks of contiguous zeros which do not fill in during factorization.

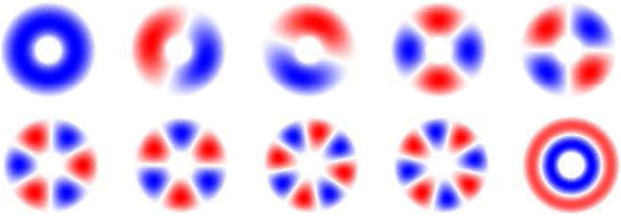

FIGURE 12. From left to right and up to down, the first ten TM modes of a coaxial line.

5. Numerical Calculations

After discretization by the edge finite element method we arrive at

$$\mathbf{S}\vec{e} = k^2\mathbf{M}\vec{e}; \tag{12}$$

here we have assumed constant material parameters so the matrices are symmetric and sparse. Our next task is to numerically solve a generalized eigenproblem; to this end we can take a look at [6], which provides a survey of some free available software for sparse eigenvalue problems. There are two main approaches to solve the generalized eigenproblem *direct solvers and iterative solvers*. In direct solvers a sequence of transformations is applied to reduce the generalized eigenproblem to an standard eigenproblem; here it is desirable to avoid the fill in the Cholesky decomposition. On the other hand iterative solvers work in the original eigenproblem and try to extract selected eigenvalues and eigenvectors from appropriate low dimensional subspaces of R^n.

Perhaps one of the simplest ways to solve generalized eigenproblems is by using the Matlab function `eigs`. This function implements an Implicitly Restarted Arnoldi algorithm [24].

We investigate the performance of this solver in the cases of banded and non-banded sparse matrices. To this end, we consider the eigenvalue calculation of TM modes in a coaxial cable with inner radius r_1 and outer radius r_2. Here $r_2/r_1 = 4$, the mesh has 10014 elements and 15187 edges; matrices S and M have size 15187×15187 and 75271 nonzero elements. Figure 12 shows the first ten TM modes.

The corresponding cutoff wavenumbers are given by 1.0255, 1.1132, 1.1132, 1.3317, 1.3318, 1.6089, 1.6089, 1.9023, 1.9023, 2.0832 in agreement with the calculated values in literature.

6. Conclusions

In this work we have investigated some computational aspects of the eigenvalue calculation with edge elements, including the importance of the grid generator and node-edge ordering. We have observed how the sparse structure of the mass and stiffness matrices is highly influenced by the edge numbering. Grid generators are mainly designed for node based finite elements, so an edge numbering is required. Two numbering schemes for the edges were investigated, and six grid generators were tested summarizing their suitableness for the edge element formulation.

As far as we know, this is the first time that different meshing algorithms haven been compared to show that significant bandwidth reduction can be obtained by the proper combination of the edge numbering scheme with the grid generator method. In fact Sc_2 only gives good results with Qmg (quadtree based); for the other grid generators Sc_1 is a better choice. The RCM reordering of the mesh followed by Sc_1 can improve the bandwidth reduction with all the grid generators except with meshnewer.

The ordering of meshnewer is optimal with Sc_1, so that no RCM reordering is required, making this grid generator a suitable choice for edge element formulation.

No bandwidth reduction is obtained by RCM of the mesh followed by Sc_2. Moreover, RCM of the matrices improves the bandwidth reduction reducing the storage requirements.

The eigensolver was not affected by the bandwidth of the matrices because the command eigs in Matlab solves linear systems internally when the eigenproblem is generalized. This suggests that in order to speed up the computations, a further study with banded generalized eigensolver must be conducted [14] where the eigensolver must take advantage of the banded structure of the matrices [5]. For the direct solvers, it is mandatory to investigate other renumbering schemes (for example minimum degree) and compare their fill in of the Cholesky factorization; on the other hand, for the iterative solvers the influences of the bandwidth reduction of the matrices on the performance of matrix-vector multiplication should be investigated.

1. P.R. Amestoy, T.A. Davis and L.S. Duff , *SIAM Journal on Matrix Analysis and Applications* **17** (1996).
2. Ansys, www.ansys.com
3. D.A. Burgess and M.B. Giles , *Renumbering Unstructured Grids to Improve Performance of Codes on Hierarchical Memory Machines*, Advances in Engineering Software, (1997) 189.
4. E. Cuthill and J. McKee ,*Reducing the bandwidth of sparse symmetric matrices*, Proceedings of the (1969) 24th national conference, pp. 157.
5. C.R. Crawford , *Communications of the ACM*, **16** (1973).
6. V. Hernández, J.E. Román, A. Tomas, V. Vidal, *A survey of Software for Sparse eigenvalue problems*, SLEPc Universidad Politécnica de Valencia, available at www.grycap.upv.es/slepc
7. J.M. Jackson, *Classical Electrodynamics*, third edition, (Wiley 1998).
8. Jin Jianming, *The Finite Element Method in Electromagnetics* (John Wiley and Sons Inc, 2002).

9. B.K. Karamete, *Mechanical Engineering Seminar* Series-4, (Purdue School of Engineering and Technology-IUPUI, 1995).
10. A. Kaveh and H.A. Rahimi Bondarabady, *Computers and Structures* (2002) 219.
11. S.J. Owen, *A survey of unstructured mesh generation technology*, available at http://www.andrew.cmu.edu/user/sowen/survey/
12. S. Dey and R. Mittra, *Microwave and Guided Wave Letters, IEEE* **7** (1997) 273.
13. P.O. Persson and G. Strang, *SIAM Review* **46** (2004) 329.
14. L. Kaufman, *ACM Transactions on Mathematical Software* **26** (2000) 551.
15. Le Wei Li, Zhong-Cheng Li, and Mook-Seng Leong, *IEEE Transactions On Microwave Theory and Techniques* **51** (2003).
16. Owen Steven, *A survey of Unstructured Mesh Generation Technology*, technical report available at www.andrew.cmu.edu/user/sowen/survey
17. Pde toolbox Matlab www.mathworks.com
18. Qmg 2.0, Stephen A. Vavasis (Cornell University 1999).
19. Matlab Mesh2d by Darren Engwirda, 2005-07, available to download at http://www.mathworks.com/matlabcentral/fileexchange
20. J.M. Nielson, P.E. Latham, M. Caplan, and W.G. Lawson, *IEEE Transactions on Microwave Theory and Techniques* **37** (1989).
21. C.J. Reddy , M.D. Deshpande, C.R. Cockrell, and F.B. Beck, *Finite Element Method for Eigenvalue Problems in Electromagnetics* (NASA technical report 3485).
22. J.R. Shewchuk, *Appl. Comp. Geom.* **1148** (1996) 203.
23. J.L. Volakis, A. Chatterjee, and L.C. Kempel, *Finite Element Method for electromagnetics: with applications to antennas, microwave circuits, and scattering* (IEEE Press 1998).
24. R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicit Restarted Arnoldi Methods* (SIAM Publications, 1998).