Tetrahedral Grid Generators and the Eigenvalue Calculation with Edge Elements

Generadores de malla tetraédricos y el cálculo de Eigenvalores con elementos de contorno

Gerardo Mario Ortigoza Capetillo
Facultad de Ingeniería Universidad Veracruzana
Calzada Adolfo Ruiz Cortines s/n, Fracc. Costa Verde Boca del Río Ver, México
E-mail:gortigoza@uv.mx

Article received on September 03, 2008; accepted on February 25, 2009

Abstract. 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.
Tetrahedral grid generators are mainly designed for nodal based finite elements so an edge numbering is required. Two different edge numbering schemes are tested with six different grid generators. 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: Tetrahedral grid generators, edge elements, RCM ordering, generalized eigenproblem.

Resumen. En este trabajo se investigan algunos aspectos computacionales del cálculo de eigenvalores con elementos de contorno tales como la importancia del generador de mallas y la numeración de nodos y lados. Como muestran los ejemplos, la estructura esparcida de las matrices de masa y momentos es altamente influenciada por la numeración de los lados.
Generadores de mallas en tetraedros son diseñados principalmente para elementos finitos basados en los nodos, así una numeración de los lados es requerida. Se realizaron pruebas con dos esquemas de enumeración de los lados con seis generadores de mallas distintos. Una reducción de banda significante puede obtenerse con una combinación apropiada de esquema de numeración de los lados con el método empleado por el generador de malla. Más aún un algoritmo de reordenamiento como el RCM puede mejorar la reducción de ancho de banda lo cual es necesario para reducir los requerimientos de almacenamiento.

Palabras Clave: Generadores de mallas en tetraedros, elementos de contorno, reordenamientos RCM, valores propios generalizados.

1 Introduction

In electromagnetics, 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. The finite element method with edge elements has been used to solve these kind of problems; some of the advantages of edge elements include: divergence free (elimination of spurious nonphysical solutions), 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 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 as $O(N)$, where N denotes

the degrees of freedom of the problem. Moreover, storage can be reduced by minimizing the bandwidth of the connectivity matrix; thus, generalized eigensolvers that take advantages of the banded structure are highly desirable. The work is organized as follows: in section 2 we introduce the finite element formulation for eigenvalue problems in electromagnetics by using edge elements (three dimensional Whitney elements), section 3 shows the influence of the mesh generator in the structure of the mass and stiffness matrices, in section 4 we present the use of the RCM ordering algorithm to reduce the bandwidth of the matrices and section 5 includes conclusions of this work.

2 The edge elements for the eigenvalue calculation

The problem of calculating resonant frequencies of three-dimensional cavities can be formulated either by using the **E** or the **H** fields ([1], [2]). Let us consider the vector wave equation

$$\nabla\times\left(\frac{1}{\mu_\gamma}\nabla\times E\right)-k_c^2\varepsilon_\gamma E=0 \qquad \textbf{(1)}$$

Where $E=E_x\hat{x}+E_y\hat{y}+E_z\hat{z}$, μ and ε_γ are the permittivity and the permeability respectively of the material.

A. Finite Element Formulation

In order to get the weak formulation let us multiply equation (1) by a vector testing function $\mathbf{W}_n$ and integrate over the volume **V** of the cavity [3]

$$\int_V\left(W_n\cdot\left(\nabla\times\left(\frac{1}{\mu_\gamma}\nabla\times E\right)\right)-k_c^2\varepsilon_\gamma W_n\cdot E\right)dv=0 \qquad \textbf{(2)}$$

by using the identity

$$A\cdot(\nabla\times B)=(\nabla\times A)\cdot B-\nabla\cdot(A\times B) \qquad \textbf{(3)}$$

equation (2) can be written as

$$\int_V(\nabla\times W_n)\cdot\left(\frac{1}{\mu_\gamma}\nabla\times E\right)dv=$$
$$k_c^2\varepsilon_\gamma\int_\Omega W_n\cdot E\,dv+\int_V\nabla\cdot\left[W_n\times\left(\frac{1}{\mu_\gamma}\nabla\times E\right)\right]dv \qquad \textbf{(4)}$$

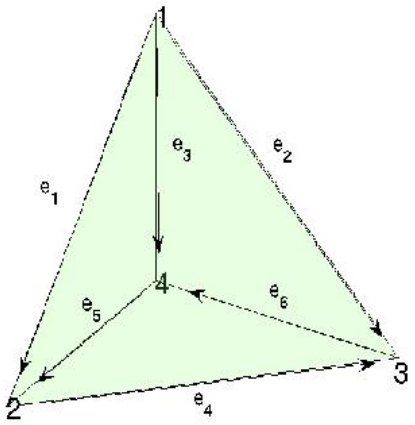

Fig. 1. Configuration of tangential edge elements

Table 1. Edge definition for a tetrahedral element

Edge i	Node i_2	Node i_2
1	1	2
2	1	3
3	1	4
4	2	3
5	4	2
6	3	4

by using the divergence theorem and the identity $(A\times B)\cdot\hat{n}=-A\cdot(\hat{n}\times B)$ we have

$$\int_V(\nabla\times W_n)\cdot\left(\frac{1}{\mu_\gamma}\nabla\times E\right)dv=$$
$$k_c^2\varepsilon_\gamma\int_\Omega W_n\cdot E\,dv-\int_S W_n\cdot\left[\hat{n}\times\left(\frac{1}{\mu_\gamma}\nabla\times E\right)\right]ds \qquad \textbf{(5)}$$

here **V** indicates integration over the volume, S over the outer surface, $\hat{n}$ is the outward unit vector normal

to the surface. For a cavity bounded by perfectly electric conducting electric conductor, the field as well as the testing function $\mathbf{W}_n$ has to be zero on the outer surface; hence the last term on the right-hand side vanishes. Thus we have

$$\int_V \frac{1}{\mu_\gamma}(\nabla \times W_n) \cdot (\nabla \times E)\, dv = k_c^2 \varepsilon_\gamma \int_V W_n \cdot E dv \tag{6}$$

The electric field in a single tetrahedral element is represented as

$$E = \sum_{m=1}^{6} \varepsilon_m W_m \tag{7}$$

here $W_m = l_m \left(L_{m_1} \nabla L_{m_2} - L_{m_2} \nabla L_{m_1} \right)$, l_m is the length of edge m connecting nodes m_1 and m_2; L_{m_1} and L_{m_2} are the simplex coordinates associated with nodes m_1 and m_2. Fig. 1 and table I show the definition of the edges. In order to obtain the finite element formulation we substitute equation (4) into (3) to get

$$\frac{1}{\mu_\gamma} \sum_{m=1}^{6} \int_\Delta (\nabla \times W_n) \cdot (\nabla \times W_m)\, \varepsilon_m dv = k_c^2 \sum_{m=1}^{6} \varepsilon_\gamma \int_\Delta (W_m \cdot W_n)\, \varepsilon_m dv \tag{8}$$

n = 1, · · · 6, here _ denotes integration over the volume of the tetrahedron. This can be written in matrix form as

$$\left[S_{mn}^{el} \right][e] = k_c^2 \left[M_{mn}^{el} \right][e] \tag{9}$$

where the element matrices are given by

$$\left[S_{mn}^{el} \right] = \frac{1}{\mu_\gamma} \int_\Delta (\nabla \times W_m) \cdot (\nabla \times W_n)\, dv \tag{10}$$

and

$$\left[M_{mn}^{el} \right] = \varepsilon_\gamma \int_\Delta (W_m \cdot W_n)\, dv \tag{11}$$

that after a loop over all the tetrahedrons we obtain a global eigenmatrix equation

$$[S][e] = k_c^2 [M][e] \tag{12}$$

3 The influence of the Mesh Generator

The most popular element shapes employed for three-dimensional applications are tetrahedrons; this is due that tetrahedral element is the simplest tessellation shape for modeling three dimensional geometries and is also well suited for automatic mesh generation. To investigate the influence of the mesh generator we considered six different tetrahedral grid generators: initmesh [4] (Femlab), Tetgen [5], Distmesh [6], Qmg [7], Gambit [8] (Fluent) and Ansys [9]. Initmesh is a Matlab function of the Femlab package that implements a Delaunay tetrahedralization algorithm, Tetgen is a mesh generator that uses constrained Delaunay tetrahedralization, Distmesh is based on an iterative continuous smoothing method, Qmg uses a quadtreebased algorithm and finally Ansys and Gambit use an advancing front method. We start our discussion with some observations about the sparsity pattern of the stiffness S and mass M matrices. In order to efficiently allocated storage the number of edges (degrees of freedom) can be calculated by using the formula provided by Hoole [10]. At [11] a bound for the number of nonzero entries of the stiffness and mass matrices for triangular meshes was given, however the analogy does not hold in the three dimensional case. For vector elements the unknowns are still associated with the edges of the elements; but in 3d an edge either on the boundary or at the interior of the computational domain can be shared for more than two tetrahedrons which make difficult to determine the number of neighboring edges. 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 none provides the edge numbering), thus we need to convert node numbering into edge numbering. Here we follow the two simple schemes by Jin [1].

To describe the numbering schemes we take as an example the four elements tetrahedral mesh at Fig. (2).

its element-to-node connectivity array is given at table 2. For the first scheme denoted by Sc_1, an indicator (product of the nodes) to each edge is defined and the array of

1	2	3	4
4	1	3	5
1	6	3	5
2	1	6	3

Table 2. Four element tetrahedral mesh

Indicator	Nodes	Elements
2	12	1
4	14	1
3	13	1
8	24	1
6	23	1
12	34	1
4	14	2
12	34	2
20	45	2
3	13	2
5	15	2
15	35	2
6	16	3
3	13	3
5	15	3
18	36	3
30	56	3
15	35	3
2	12	4
12	26	4
6	23	4
6	16	4
3	13	4
18	36	4

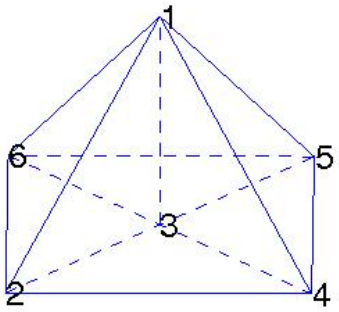

Fig. 2. A four elements tetrahedral mesh

Table 3. Table of edges

Table 4. Table of sorted edges

Indicator	Nodes	Element
2	12	1
2	12	4
3	13	1
3	13	2
3	13	3
3	13	4
4	14	1
4	14	2
5	15	2
5	15	3
6	16	3
6	16	4
6	23	1
6	23	4
8	24	1
12	26	4
12	34	1
12	34	2
15	35	2
15	35	3
18	36	3
18	36	4
20	45	2
30	56	3

Table 5. Edge to node array

Edge	Nodes	Elements
1	12	1
2	13	1
3	14	1
4	15	2
5	16	3
6	23	1
7	24	1
8	26	4
9	34	1
10	35	2
11	36	3
12	45	2
13	56	3

indicators is rearranged by a sorting algorithm. Tables 3 and 4 show the edges and sorted edges.

Most sorting algorithms are very efficient and can perform the task with Nlog(N) operations (N is equal to six times the number of tetrahedrons). Now we proceed to count the edges, here the indicator is used to reduce the number of comparisons, thus we get the edge-to-node/element array given at table V. We finally use this array to get the element-to-edge connectivity array given at table VI.

For the second scheme Sc2, no sorting algorithm is required; we use the element-to-node connectivity array to generate a table of edges displayed at table 7. 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, if it is zero we give it the value of the counter, this algorithm requires 18m (m− 1) operations, here m is the number of edges.

Table 6. Element-edge array scheme 1

Edge 1	Edge 2	Edge 3	Edge 4	Edge 5	Edge 6
1	3	2	7	6	9
3	9	12	2	4	10
5	2	4	11	13	10
1	8	6	5	2	11

Table 7. Edge to node array

Edge 1	Edge 2	Edge 3	Edge 4	Edge 5	Edge 6
12	14	13	24	23	34
14	34	45	13	15	35
16	13	15	36	56	35
12	26	23	16	13	36

Table 8. Element-edge array scheme 2

Edge 1	Edge 2	Edge 3	Edge 4	Edge 5	Edge 6
1	2	3	4	5	6
2	6	7	3	8	9
10	3	8	11	12	9
1	13	5	10	3	11

Table 9. Meshes information

Mesh	Created by	N-nodes	N-elements	N-edges	Size
1	Femlab	1411	6653	8620	0.1312
2	Tetgen	1370	6565	8583	0.1059
3	Distmesh	1288	6478	8325	0.1979
4	Qmg	1508	6818	9003	0.1019
5	Ansys	1387	6570	8514	0.1355
6	Gambit	1504	6857	9026	0.1291

These schemes generate different edge numberings for a given tetrahedralization as seen at tables 5 and 8. In our first experiment we used the different grid generators to define a grid for the simple geometry of a cylinder of radius 1 and height 1 with approximately 6500 elements.

The table 9 displays the information of the meshes. By using the two schemes of edge numbering we calculate the stiffness matrix S for the six meshes. Fig. 3-8 show the sparse structure of S for the two edge numbering schemes (S and M have similar structure). The poorly structured connectivity of an unstructured finite element mesh can lead to poor cache affinity ([12]).

Fig. (9) shows bandwidth for the six meshes with the two schemes. In this plot we refer to bandwidth as the half bandwidth over the number of degrees of freedom. Here we notice no significant changes in the bandwidth for the first two meshes (obtained by Delaunay tetrahedralization methods) with both schemes; similar results are observed with last two. However we note that Sc_1 works better (lower bandwidth) for the mesh generated by Distmesh while Sc_2 works better for the mesh generated by Qmg.

In fact the bandwidth obtained by using Sc_1 is only the 15.8% of the one obtained with Sc_2 for Distmesh; while for Qmg we have the opposite situation the bandwidth obtained with Sc_2 is the 17.96% of the one obtained with Sc_1. The grid generator Distmesh is based on the iterative method of Persson which tries to optimize the node locations by a force-based smoothing procedure while Qmg uses a quadtree method. It seems that grid generators based on Delaunay methods produce no significant changes in the bandwidth size with both edge numbering schemes. However Sc_1 produces a lower bandwidth that Sc_2 for Distmesh, here the iterative method of Persson gives an optimal node numbering for the Sc_1 which is based on a sorting algorithm.

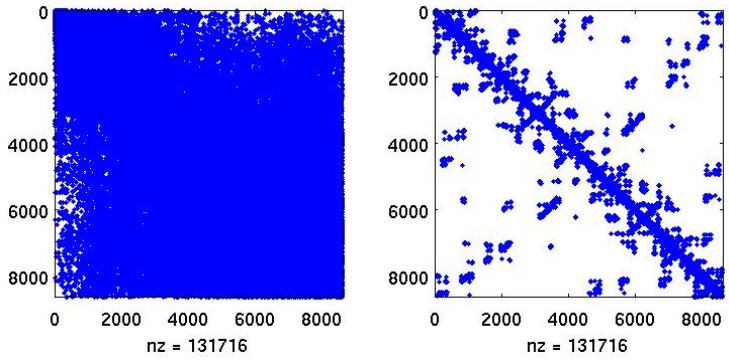

Fig. 3. Matrix Structure Femlab; left Sc 1, right Sc 2

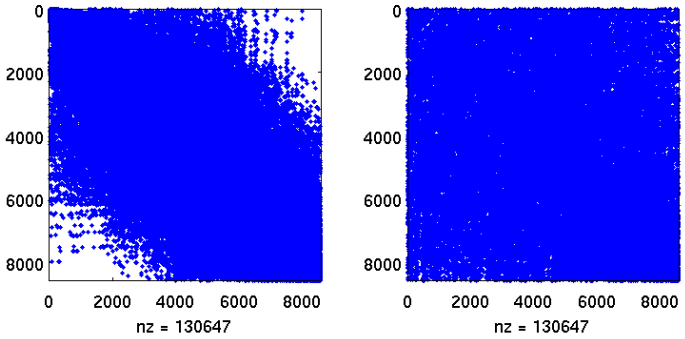

Fig. 4. Matrix Structure Tetgen; left Sc 1, right Sc 2

On the other hand Sc_2 produces lower bandwidth that Sc_1 for Qmg. In this case Sc_2 makes a loop over the elements so it seems that Qmg gives and 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 [13]. 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 cubes containing the geometric model are recursively divided until a desired resolution thus nodes and faces are formed whenever the internal octree structure intersects the boundary; whereas Delaunay generators have random numbering. Fig. 10 shows the mesh obtained by Distmesh. 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 10 summarizes some useful information of the grid generators. Among them, none provides the edge numbering, initmesh (Femlab), Tetgen and Qmg only provide the boundary edges. Here E.N. and B.E. means edge numbering and boundary elements respectively.

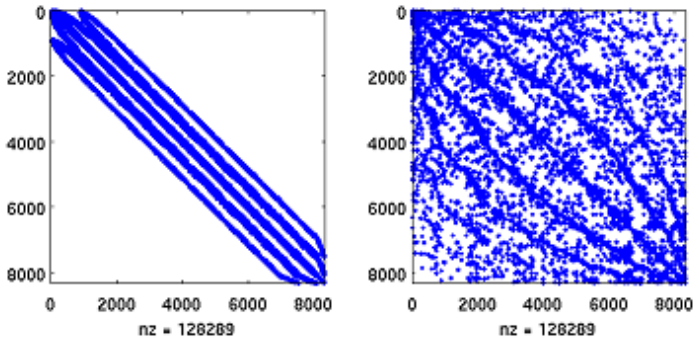

Fig. 5. Matrix Structure Distmesh; left Sc 1, right Sc 2

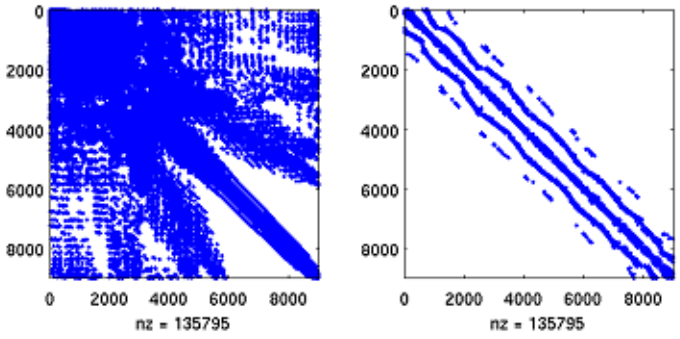

Fig. 6. Matrix Structure Qmg; left Sc 1, right Sc 2

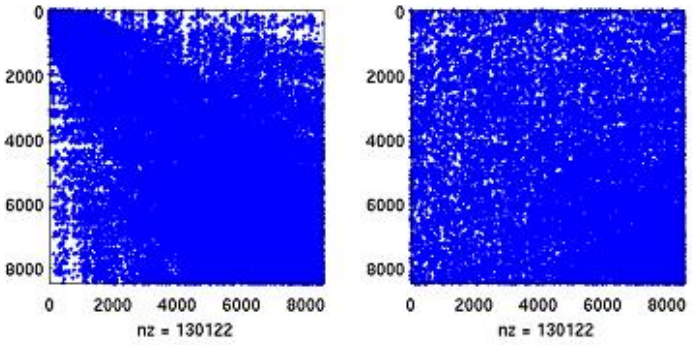

Fig. 7. Matrix Structure Ansys; left Sc 1, right Sc 2

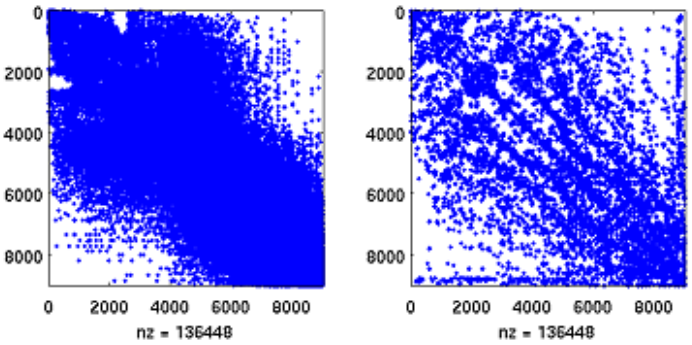

Fig. 8: Matrix Structure Gambit; left Sc 1, right Sc 2

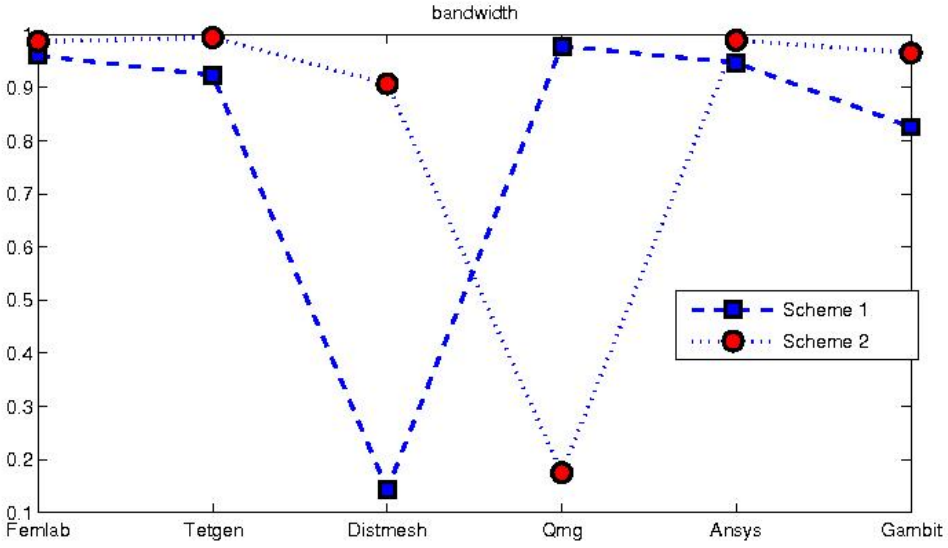

Fig. 9. Bandwidth

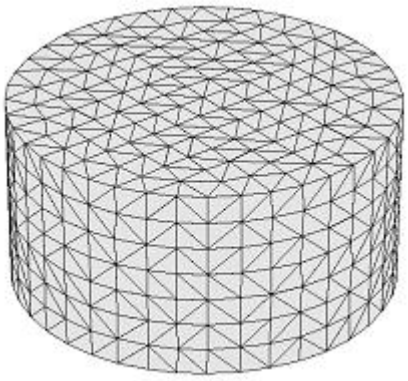

Fig. 10. Mesh for a cylinder

Table 10. Grid generators information

	Availability	**Method**	**E.N.**	**B.E.**	**Language**
Femlab	Commercial	Delaunay	X	Ok	Matlab
Tetgen	OpenSource	Delaunay	X	Ok	C++
Distmesh	OpenSource	Continuous smoothing	X	X	Matlab
Qmg	OpenSource	Quadtree	X	Ok	Matlab-C++
Ansys	Commercial	Advancing front	X	X	User interface
Gambit	Commercial	Advancing front	X	X	User interface

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 ([14],[15]).

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 ([16], [17]). A widely used but rather simple ordering algorithm is the reverse Cuthill-McKee ordering algorithm [18]. The algorithms first find a pseudo peripheral vertex of the graph of the matrix. It then generates a level structure by breadth-first search (bfs) and orders the vertices by decreasing distance from the pseudo peripheral vertex. The cost of bfs is O(|V| + |E|) with |V| and |E| the number of nodes and edges respectively.

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.

On the second one we 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 provide optimal meshes, so the RCM should be considered as part of the grid generator.

A. 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.

Fig. 11 shows the bandwidth reduction produced by the RCM algorithm. At each group the height of the columns represent 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 Distmesh, it seems that the node ordering of the mesh generated by Distmesh is optimal and a RCM reordering is not needed. Note that even though a RCM ordering of the mesh was used the Sc_2 does not provide bandwidth reduction.

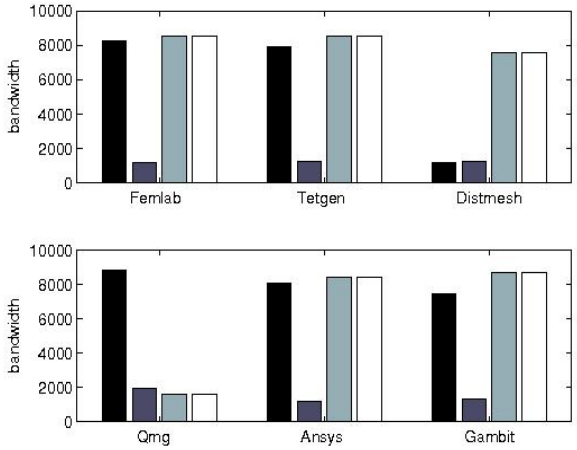

Fig. 11. Reordering the meshes

B. Reordering the matrices

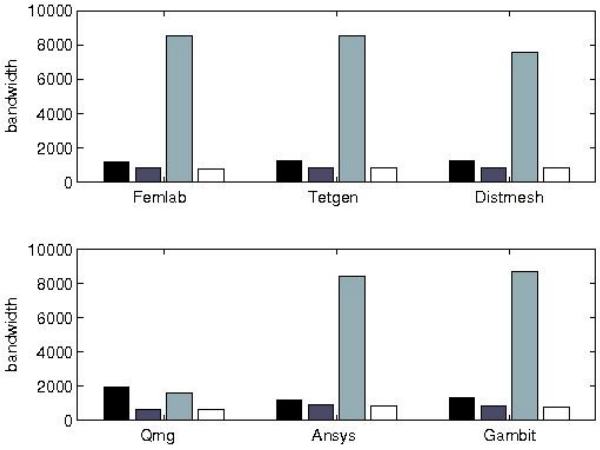

Fig. 12. Reordering the matrices

In this case the RCM ordering is applied after the matrices are assembled. Fig. 12 shows the bandwidth reduction by using RCM to the meshes (rcm1) and to the assembled matrices (rcm2). At each group the height of the columns represents the bandwidth, the first column is the 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 the cases bandwidth reduction is obtained by rcm2.

5 Eigenvalue calculation

After discretization by the edge finite element method we arrive to

$$[S][e] = k_c^2 [M][e] \qquad \textbf{(13)}$$

here we have assumed constant material parameters so the matrices are symmetric. We are now faced with the problem of numerically solve a generalized eigenproblem; one approach is to reduce it to a standard eigenvalue problem by means of congruence transformations and then use an iterative method to calculate the eigenvalues (the resulting eigenproblem amounts to solving the eigenvalues of a symmetric tridiagonal matrix); sometimes this approach is called by using direct solvers, the other approach is to directly write an iterative method for the generalized eigenproblem (iterative solvers). A review of direct solvers for the generalized eigenproblem can be found at ([19]).

Over the years, several numerical methods and software to solve large scale eigenproblems have been developed, for a comprehensible list of software and references we refer to ([20], [21]). A vast majority of the programs are based on the Lanczos algorithm including irbleigs ([22]) and eigs (Matlab implementation of Arpack) [23]; these kind of methods require the invertion of M, if the eigenvalues are badly separated a shift and invert transformation is required. Other programs such as jdqz([24]) and lopbcg([25]) do not required shift-and-invert transformation or the inversion of M but require more user inputs as initial approximations or preconditioners. An alternative matlab program that does not require user's inputs is eigifp ([26]), which uses an inverse free preconditioned Krylov subspace projection method. 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 [23]. We investigate the performance of this solver in

the cases of banded and nonbanded sparse matrices. For this end, we consider the eigenvalue calculation of the resonances of a closed rectangular empty cavity 1cm × 0.5cm×. Here the mesh has 1547 nodes, 7416 elements and 9709 edges. The first cutoff wavenumbers are given by 5.23, 7.01, 7.55, 7.56, and 8.16 in agreement with the calculated values in literature.

Even though bandwidth reduction is obtained by the proper choice of the grid generator or rcm ordering, no significant reduction have been observed in the execution time of eigs (arpack) when calculating the eigenvalues (cutoff wave numbers). 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 eigensolvers either direct or iterative must be conducted [27].

6 Conclusions

In this work we have investigated the importance of the grid generator and edge numbering in the eigenvalue calculation with edge elements. 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 element, so an edge numbering is required. Two numbering schemes for the edges were investigated, six grid generators were tested summarizing their suitableness for the edge element formulation. 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 the Sc_1 can improve the bandwidth reduction with all the grid generators except with Distmesh. The ordering of Distmesh is optimal with Sc_1, thus no RCM ordering is required which make this grid generator a suitable choice for edge element formulation.

We remark the point that a RCM ordering of the mesh followed by Sc_2 does not provide bandwidth reduction. Moreover, RCM of the assembled matrices improves the bandwidth reduction reducing the storage requirements (reordering the assembled matrices leaves the eigenvalues invariant) with the downside of requiring the assemble of the matrices. As future work it is due to investigate the performance of the available eigensolvers in order to determine the most suitable one for the kind of generalized eigenproblems arising in electromagnetics.

References

1 **Jianming J. (2002).***The Finite Element Method in Electromagnetics* (2nd ed.). New York: John Wiley and Sons.

2 **Volakis, J. L., Chatterjee, A., & Kempel, L. C. (1998).** *Finite Element Method for electromagnetics: to antennas, microwave circuits, and scattering applications.* New York: IEEE Press.

3 **Reddy, C.J., Deshpande, M.D**., **Cockrell, C. R., & Beck, F. B. (1998).** Finite element method for eigenvalue problems in electromagnetics. (NASA Technical Paper 3485). Hampton, Virginia: NASA Center for AeroSpace Information

4 Multiphysics Modeling and Simulation Software- COMSOL. (s.f.). Retrieved from http://www.comsol.com/

5 **Hang, S., & Gartner, K. (2005).** Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. *14th International Meshing Roundtable*, San Diego, CA, USA, 147–164.

6 **Persson, P.O., & Strang, G. (2004).** A simplemesh generator in matlab. *SIAM Review*, 46(2), 329–345.

7 Qmg 2.0 (s.f.) Retrieved from http://www.cs.cornell.edu/home/vavasis/qmg2.0/qmg2_0_home.html

8 CFD Flow Modeling Software & Solutions from Fluent (s.f.). Retrieved from http://www.fluent.com./

9 Ansys Simulation Driven Product Development (s.f.). Retrieved from http://www.ansys.com/

10 **Hoole, S.R.H., Jayakumaran, S., & Yoganathan S. (1986).** Tetrahedrons, edges and nodes in 3d finite element meshes. *Electronic Letters*, 22 (14), 735–737.

11 **Ortigoza, G. (2009).** Triangular grid generators for the eigenvalue calculation with edge elements. *Revista Mexicana de Física*, 55 (2), 154-160.

12 **Shires D., & Mohan R. (2003).** Optimization and performance of a FORTRAN 90 mpi-based unstructured code on large-scale parallel systems. *The Journal of Supercomputing*, 25 (2), 131–141.
13 A survey of unstructured mesh generation technology retrieved from www.andrew.cmu.edu/user/sowen/survey/
14 **Hansen, P. C. T., Ostromsky, A., Basermann, P.**, **(1994).** Weidner., Reordering of sparse matrices for parallel processing. APPARC PaA3a Technical report,
15 **Burgess D. A., & Giles, M. B. (1997).** Renumbering unstructured grids to improve performance of codes on hierarchical memory machines. *Advances in Engineering Software*, 28 (3), 189–201.
16 **Kaveh, A. & Rahimi Bondarabady, H. A. (2002).** An hybrid method for finite element ordering. *Computers & Structures*, 80 (3-4), 219-225.
17 **Paulino, G. H., Meneses, I.F., Gattass M., & Mukherjee, S.** (1994). Node and element resequencing using the Laplacian of a finite element graph: Part I – General concepts and algorithm. International Journal for Numerical Methods in Engineering, 37(9), 1511–1530.
18 **Cuthill, E., & McKee, J. (1969).** Reducing the bandwidth of sparse symmetric matrices. *ACM Annual Conference 24th national conference*, New York, USA, 157–172.
19 **Lang, B. & Aachen, R. (2000).** Direct solvers for symmetric eigenvalue problems. In Johannes Grotendorst (Ed.) *Modern methods and algorithms of quantum chemistry proceedings*, Jülich, Alemania, 3, 231–259.
20 **Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. & Van der Vorst, H. (2000).** *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Philadelphia: Society for Industrial and Applied Mathematics
21 **Hernandez, V., Román, J.E. Tomas, A., & Vidal, V. (2007).** *A survey of software for sparse eigenvalue problems*. (SLEPc Technical Report STR-6). España: Universidad Politécnica de Valencia.
22 **Baglama, J., Calvetti, D., & Reichel, L. (2003).** Algorithm 827: irbleigs: a matlab program for computing a few eigenpairs of a large sparse hermitian matrix. *ACM transactions on mathematical software*, 29 (3), 337–348,
23 **Lehoucq, R. B., Sorensen, D. C., & Yang, C. (1998).** *Arpack: user's guide: Solution of large-scale eigenvalue problems with implicit restarted arnoldi methods.* Philadelphia : SIAM.
24 **Fokkema D., Sleijpen G., & Van der, H. (1998).** Jacobi-Davidson style qr and qz algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20 (1), 94–125.
25 **Knyazev, A. (2001)** Towards the optimal preconditioned eigensolver: locally conditioned conjugate gradient method. *SIAM Journal of Scientific Computation*, 23 (2), 517–541.
26 **Money J. H. & Ye Q. (2005).** Algorithm 845: Eigifp: a matlab program for solving large symmetric generalized eigenvalue problems. *ACM transactions on mathematical software*, 31 (2), 270–279.
27 **Kaufman L. (2000).** Band reduction algorithms revisited. *ACM Transactions on Mathematical Software*, 26 (4), 551–567.

Gerardo M. Ortigoza C. *is a full time professor and researcher in Electrical and Mechanical Engineering at the Universidad de Veracruz, Veracruz. He got a Ph.D in Mathematics Applied to Industry in Minnesota University, USA. His research interest are in mathematical modelling and numerical simulation*