Mecánica Computacional Vol. 14, páginas 505-511
compilado por Sergio Idelsohn y Victorio Sonzogni
Editado por AMCA, Santa Fe, 1994

MESH OPTIMIZATION IN THREE DIMENSIONS

Gustavo C. Buscaglia[1,2], Enzo A. Dari[1,2,3] and Pablo Zavattieri[2]

[1] Centro Atómico Bariloche, Comisión Nacional de Energía Atómica
[2] Instituto Balseiro, Universidad Nacional de Cuyo and C.N.E.A.
[3] Consejo Nacional de Investigaciones Científicas y Técnicas

8400 - S.C. de Bariloche - ARGENTINA

ABSTRACT

We discuss an optimization procedure for improving three-dimensional finite element meshes. Our method can be effectively coupled with any automatic mesh generator in order to obtain a triangulation without badly distorted elements. In this way, we are able to weaken the requirements on the generator, allowing it to produce singular elements, and post-process the result to get a valid mesh.
Even though several other methods to improve finite element meshes already exist, the proposed one is the first to incorporate what finite element practitioners have for long known: One bad element renders the mesh useless. We report some very encouraging results with a quite crude algorithm. Our conclusion is that building meshes combining an initial generation step followed by an optimization step makes the meshing of arbitrary domains much more reliable, as compared to the usual approach of precluding distorted elements during the initial generation stage.

INTRODUCTION: HOW IS A 3D MESH BUILT?

The generation of a three-dimensional finite element grid in a general domain is by no means a simple task. For Delaunay-based and Frontal-based strategies, the generation procedure can be divided into the following steps:

Step 1: Geometrical description of the domain. This is usually accomplished by means of a CAD output.
Step 2: Surface mesh generation. A finite element mesh is constructed over the boundary of the domain. An efficient algorithm for this purpose has been recently proposed by Vénere&Arnica [21], it allows for mesh grading according to user's needs.
Step 3: Volume mesh generation. Our experience mainly concerns two of the several existing alternatives for this step, namely Delaunay and Frontal volume generators.

Delaunay scheme

3D.a. Generation of points: The nodes of the triangulation are created according to a pre-specified density function. Our current choice is the 3D extension of the algorithm introduced by Dari&Vénere [7].
3D.b. Triangulation: The Delaunay mesh containing the nodes is generated, by Bowyer-Watson algorithm. Other variants create the nodes while generating the mesh [15,22].
3D.c. Boundary recovery: The Delaunay mesh is not in general compatible with the boundary mesh generated in Step 2. The mesh must thus be modified to recover boundary edges and boundary faces. The procedure we are using is described in [4], other possibilities can be found in [9,22].

Frontal scheme

3F.a. Triangulation: Using the boundary mesh as initial front, advance the front generating new nodes and triangles until the whole of the domain is filled and the front collapses.

QUALITY: HOW GOOD IS A 3D MESH?

Not all meshes are suitable for finite element calculations. The approximation error depends on the size and the shape of the elements. The element sizes result as a compromise between accuracy and computing cost. The sizes may vary through the domain according to an error estimate or, perhaps, to user's experience. The element shapes set apart good generators from bad ones.

Typically, the element shape appears in the error estimate as a factor h_K / ρ_K, with h_K the diameter of element K and ρ_K the radius of the inscribed sphere. This quotient is thus a measure of the quality of a finite element.

To keep the discussion at a more intuitive level, we will speak about the angles contained in a mesh. More specifically, we will consider 3D meshes consisting of linear tetrahedra. It can be proved [4] that *the quotient h_K / ρ_K is bounded from above for all elements of a family of triangulations if and only if the angles of the mesh are bounded from below by some positive constant.* By angles in the previous statement we refer to *both* the angles between adjacent faces and the angles between adjacent edges. Hence, the angles of the mesh can also be used to measure its quality.

We should remark that Kfízek [13,14] has proved that most pernicious to the approximation properties of a finite element method are *large* angles (close to 180 degrees), while small ones can sometimes be harmless (and necessary, e.g. to cover a thin slot).

There exists another alternative, less costly in terms of floating-point operations. Distorted elements can be identified looking for small values of the quotient V_K / h_K^3, where V_K stands for the volume of element K.

We will assume that the boundary mesh does not contain badly distorted elements. The procedure of Vénere&Arnica [21] yields meshes satisfying this condition. The question now is: How good is the quality of the meshes generated following the methods described in the previous section (Step 3)?

DELAUNAY MESHES AND QUALITY

Bad-quality elements that appear in a mesh generated by the Delaunay-based method mainly come from three sources: (1) Abruptly varying point density, (2) Almost planar Delaunay elements (*slivers*), and (3) Distorted elements generated during boundary recovery.

Such elements can have angles of zero degrees and of 180 degrees (within round-off error). In our experience, practically all meshes of more than, say, 30000 tetrahedra, contain at least one element with an angle greater than 179.90 degrees.

If the mesh generation is programmed without care, the number of bad elements can increase significantly. As the nodal positions define the Delaunay mesh, it is important to ensure that the point-generation scheme (Step 3D.a) produces smoothly varying point densities; and to avoid the insertion of points too close to boundary faces. Also, the prescribed point density must be compatible with the density of the surface mesh. Another important detail is that the set of elements that do not pass the Delaunay test must be, *by construction*, connected, precluding round-off errors from producing a disconnected set.

The state of the art is that, even with the best Delaunay-based generator available, bad elements do appear. And the same happens using a frontal method, as we proceed to discuss.

FRONTAL GENERATORS AND QUALITY

The idea of a front-advancing scheme [16,17,18], that begins at the boundary of the domain and generates new points and elements as it progresses inwards, is very elegant and attractive. Moreover, the difficulty of recovering the boundary mesh is avoided. When the time comes to implementation, however, much of this elegance is lost. In fact, notice that not always a new node is created to generate a new tetrahedron (otherwise, the number of faces in the front would grow to infinity). Several tests must thus be performed to determine whether a new point must be created, and where to do so. Even if an already existing node will be used to advance the front, it must be tested if the new edges will intrude in already existing elements and, *also*, if any of the already existing edges will pierce the new faces we are generating. As a result, a frontal generator spends most of its computing time performing floating-point test operations. These tests are "passed" or not according to several user-defined tolerances. If these tolerances are too strict, the generator simply aborts leaving holes inside the domain. Contrariwise, if the tolerances are enlarged, the generator may succeed in filling the volume at the expense of generating some elements of very low quality.

In our experience, the tolerances required to generate a 30000-element mesh make the generator to produce at least two or three elements with zero-degree and 180-degree angles, i.e. no better than the Delaunay-based method.

Other remarkable facts are: (1) Properly programmed frontal schemes are much more expensive in computer time than Delaunay-based schemes, by a factor of at least twenty. (2) Any minor simplification in the geometrical tests may cause the self-penetration of the advancing front and the collapse of the algorithm. It must be kept in mind that, when dealing with hundreds of thousands of elements, whatever may happen *will* occur.

WHAT TO DO WITH BAD-QUALITY ELEMENTS?

Considering that 3D mesh generators produce elements not suitable for calculation, the question arises of what to do with them. This has motivated intensive research on mesh-improving techniques. Much experience in smoothing meshes also comes from the area of deformable domains (free-boundaries, interfaces, etc., in metal forming or flow problems). In this later case, elements with high initial quality distort as they follow the movement of the domain's boundary.

Except for interior slivers (Baker [1] has shown how to remove a sliver from a Delaunay mesh inserting one additional node), bad elements must generally be enhanced modifying the nodal positions. Kennon [11] and Tezduyar *et al* [20] (among others) have developed quite effective node-repositioning strategies, based on auxiliary elastic problems. These strategies are clever generalizations of the spring systems frequently used for 2D meshes [e.g.,18], but all elastic-like schemes fail to give a correct 3D mesh when concave or high curvature boundaries are present.

As we have seen, mesh smoothing is crucial for 3D meshes. As a consequence, a new line of research has appeared under the name of *mesh optimization* [3,5,6,12,19]. Mesh optimization is a particular kind of smoothing, which relocates nodes so as to maximize some mesh-quality function, following an optimization procedure over the space of nodal coordinates. Cabello *et al* [3], Stamatis&Papailiou [19] and Zhang&Trépanier [23] define the mesh quality as an average of the qualities of the elements contained in the mesh. This quality is a smooth function of the nodal positions, and they solve the maximization problem by conjugate-gradient or steepest-descent methods. Their results are quite good, but only 2D cases are considered and some surprises could appear in 3D. A similar averaging approach had already been succesfully applied by Kennon&Dulikravich [12] in three dimensions to structured grids.

In [5,6], Dari&Buscaglia have proposed to use, as quality of the mesh, the quality of its *worst* element, i.e.

$$Q_{global} = \min_K \, Q_K$$

This choice incorporates a well-known fact: One unacceptable element renders the mesh useless. The problem is then that of maximizing an objective function of the min type. As min functions are non-differentiable, they used a quite crude node-by-node algorithm (described later). Their method succeeded in producing good quality grids using as initial points both frontal and Delaunay 3D meshes, without altering the node-adjacency structures. In this way, it was shown that as-generated meshes can be much enhanced by nodal repositioning alone, what is a non-trivial assertion in three dimensions. Several examples, with meshes of up to 200000 tetrahedral elements, can be found in [2].

Instead of moving the nodes, it could be possible to improve the quality of a mesh by modifying its node-adjacency structure. Moreover, one could ask about the best topological structure so that, combined with nodal relocation, a mesh of optimal quality is obtained. The *mesh relaxation* algorithm of Frey&Field [8] seems to answer this question in *two* dimensions. Some attempts have been performed by Dari&Buscaglia [5], but with limited success. The correct extension of the Frey&Field algorithm to 3D remains an open question.

A MESH OPTIMIZATION ALGORITHM

We are currently investigating the following algorithm:

1. Determine the element of lowest quality in the mesh, K_{worst} . If this quality is already greater than some acceptance criterion, stop.
2. Find the n-th order neighboring nodes to K_{worst}. Its zero-order neighbors are the four nodes contained in K_{worst}. Its first-order ones are those nodes belonging to elements adjacent to the zero-order neighboring nodes, and so on. Let N_n be the number of n-th order neighbors.

Define now, as the *cluster* C_I associated to a node I the set of elements that share this node. The next step is

3. Sweep the N_n clusters associated with each neighboring node, modifying the nodal positions one at a time, according to the *local cluster optimization* rule, so as to improve the quality of the cluster.
4. Go back to 1.

In this way, we are at each step optimizing only a part of the complete mesh, comprising the worst element and its n-th order neighbors. Any restriction in the movement of nodes, such as those belonging to exterior or interior boundaries, is handled without difficulty during the local optimization of those clusters associated with constrained nodes. We now concentrate on one fixed vertex, and make explicit our algorithm for local cluster optimization. The quality of each element is defined as V_K / h_K^3. As the quality of the cluster is the minimum quality of the elements contained in it, and thus non-differentiable, standard optimization schemes would not work. We are at the moment using the following (rather crude) one:

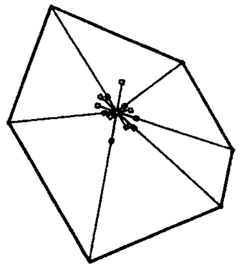
Figure 1: Sampling positions for local cluster optimization in 2D.

We associate to the cluster C two scalar quantities, a primary quality q_1, defined by $q_1 = \min_{K \in C} Q_K$ (minimum quality of the cluster) and a secondary one q_2, defined as the *average* quality in C. Both q_1 and q_2 are continuous functions of the position x_I of the central node I of the cluster. We next introduce a set of sampling positions. Let x_I^o be the original position of node I, then, for every vertex V of the cluster define two sampling points located at $x_{sp} = \pm \alpha\, x_V + (1 \mp \alpha) x_I^o$, with α a small parameter, typically 0.05. Notice that x_I^o is itself a sampling position. In two dimensions, the set of sampling points defined in this way would look as in Fig. 1. The local cluster optimization rule we have implemented is the following:

Local cluster optimization rule: *Evaluate q_1 for all sampling positions, and choose as updated position the sampling position that maximizes this primary quality. Whenever the maximum value of q_1 is shared by two sampling points, select the one with maximum secondary quality q_2.*

The introduction of the secondary quality q_2 is necessary because the point where q_1 is maximum is generally non-unique, and the algorithm needs a second criterion to avoid spurious iterations among equivalent points.

AN EXAMPLE: MESH GENERATION INSIDE A CUBE

Let us illustrate the previous exposition through an elementary example: A unit cube with homogeneous target density. We proposed as desired element sizes 1/8 for the first mesh (CUBE1) and 1/16 for the second (CUBE2). The surface meshes provided by the generator can be seen in Fig. 2. Instead of including lengthy tables, we will discuss the most striking features of the results.

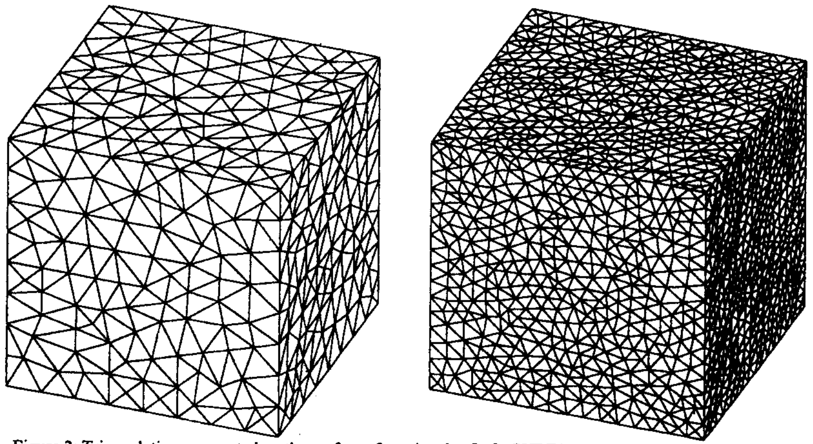

Figure 2: Triangulations generated on the surface of a unit cube. Left: CUBE1, average edge-length: 1/8, 1104 triangles. Right: CUBE2, average edge-length: 1/16, 4432 triangles.

Initial triangulation: Both the Delaunay and frontal generators succeeded in generating the meshes. Delaunay ones will be labeled CUBE1D (770 nodes, 2918 elements) and CUBE2D (5190 nodes, 23488 elements), while those obtained by the frontal method CUBE1F (723 nodes, 2441 elements) and CUBE2F (4079 nodes, 16665 elements). The frontal generator took 55 times more computing time than the Delaunay-based one. In these original meshes, the minimum dihedral angles ranged between zero (CUBE2D) and 2.56 degrees (CUBE1F), while the maximum one took values between 180 (CUBE2D) and 176.58 (CUBE1F). Also, the ratio of the volume of the largest element to that of the smallest one, which should have been close to 1.0 because the density was specified as homogeneous, rose to 51 in mesh CUBE1D, 440 in mesh CUBE1F, practically infinity in mesh CUBE2D, and 239 in mesh CUBE2F. These are clear evidences of what we were referring to when discussing the actual meshes provided by automatic generators. Although we are speaking of *our* generators, we know from informal talks with colleagues that many existing generators would produce in this situation a ratio maximum volume to minimum volume of at least 80. In [10], Jin&Tanner reported values in the range 15-30.

Optimization: We then went on to optimize the four meshes obtained in the previous step. We tried several orders of neighborhood, to determine how many nodes around the worst element one should move to attain acceptable qualities. Typically, after 20 to 40 iterations the scheme converged to a mesh with a minimum dihedral angle of 6 degrees and a maximum one of 165 (we do not include edge-to-edge angles in the discussion because they were much better than dihedral angles). This is another advantage of the V_K / h_K^3 definition of quality, it puts more emphasis on angles close to 180 degrees than in angles close to zero. Most impressive was the effect of optimization on mesh CUBE2D. Its initially singular elements were improved to the point that the dihedral angles of the optimized mesh ranged between 10.3 and 159.7 degrees. The maximum/minimum volume ratios after optimization lied between 24 and 300. Turning now to the effect of allowing n-th order neighboring nodes to move, our results are not conclusive. For some meshes, increasing n from 0 up to 3 resulted in consistently better angles. For other ones, this dependence was rather weak, and even a deleterious effect of increasing n beyond 1 or 2 could be observed. This behavior will be further discussed in the following section.

DISCUSSION

In node-by-node mesh optimization algorithms, like ours, the nodes are moved so as to improve the quality of the cluster associated to that node. If the objective is to maximize the quality of the worst element of the mesh, it is not clear where to move those nodes not belonging to that element. It is possible (and likely) that, to improve the worst element, one should deteriorate some neighboring clusters. But our algorithm will always try to maximize the cluster's quality, and therefore the best mesh will in many cases not be attained.

To overcome this difficulty, the optimization method must build search directions not consisting of moving just one point, but instead a certain number of them or, perhaps (as in [3,19]), the whole mesh. Our future research will follow this direction. We would like to retain the non-differentiability of the mesh quality function, in order to concentrate in those elements of lowest quality. To keep the computations at a local level, we will not optimize the whole mesh simultaneously, but instead, as before, some n-th order neighborhood of the worst element. From the preliminary results reported above, reasonable values for n could be 1 or 2. This leads, at each step, to a non-differentiable optimization problem in 60-180 variables.

Also important is to amend the node-adjacency structure of the mesh. Several topological operations are possible: (a) Any edge shared by three tetrahedra can be replaced by a face and vice versa; (b) any node shared by four elements can be deleted; (c) any octahedron of the mesh can be retriangulated into four tetrahedra, choosing among three possibilities. The question is how to decide if a topological change is convenient; in other words, which is the best topological structure of a mesh of tetrahedra? In 2D we know that equilateral triangles fill the space, and thus one looks for structures in which each node is shared by six elements. Up to now, we know of no formal criterion to discern if some 3D structure is better or worse than any other.

ACKNOWLEDGMENTS: This work was partially suported by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), and by Convenio Argentino-Brasileño de Mecánica Computacional (CONICET(Argentina)-CNPq(Brasil)). Our thanks to D.L.Arnica and M.J.Vénere for helpful discussions and for providing the surface mesh generator TRISURF.

REFERENCES

1. T.J.Baker, "Element quality in tetrahedral meshes", presented at the 7th Int. Conf. on Finite Element Methods in Flow Problems, Huntsville, Alabama, USA, 1989.
2. G.C.Buscaglia and E.A.Dari, "Improving the quality of three-dimensional finite element meshes: An optimization-based method", to be presented at PACAM'IV Fourth Pan American Congress of Applied Mechanics, 3-6 January 1995, Buenos Aires, Argentina.
3. J.Cabello, R.Löhner and O.-P.Jacquotte, "Recent improvements of a variational method for the optimization of directionally stretched unstructured meshes", in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, edited by N.P.Weatherill *et al*, Pineridge Press, 1994, pp. 163-175.
4. E.A.Dari, Doctoral Thesis, Instituto Balseiro, 1994.
5. E.A.Dari and G.C.Buscaglia, "Topics on finite element meshes for problems with moving boundaries", in Finite Elements in Fluids, New Trends and Applications (Edited by K.Morgan, E.Oñate, J.Periaux, J.Peraire and O.C.Zienkiewicz), Pineridge Press, 1993, pp. 726-735.
6. E.A.Dari and G.C.Buscaglia, "Mesh optimization: How to obtain good unstructured 3D finite element meshes with not-so-good mesh generators", Struct. Optim., in press.
7. E.A.Dari and M.J.Vénere, "A node placement method for 2D automatic mesh generation", Latin Amer. Appl. Res., Vol. 21, 1991, pp. 275-282.

8. W.H.Frey and D.A.Field, "Mesh relaxation: A new technique for improving triangulations", Int. J. Numer. Meth. in Eng., Vol. 31, 1991, pp. 1121-1133.
9. P.L.George, "Mailleur 3D sous contraintes pour la méthode des éléments finis", in Segundo Congreso Franco-Chileno y Latinoamericano de Matemáticas Aplicadas, Santiago, Chile, December 1989.
10. H.Jin and R.I.Tanner, "Generation of unstructured tetrahedral meshes by advancing front technique", Int. J. Numer. Meth. in Eng., Vol. 36, 1993, pp. 1805-1823.
11. S.R.Kennon, "Supersonic inlet calculations using an upwind finite-volume method on adaptive unstructured grids", AIAA Paper 89-0113, 27th Aerospace Sci. Meeting, Reno, USA, 1989.
12. S.R.Kennon and G.S.Dulikravich, "Generation of computational grids using optimization", AIAA J., Vol. 24, 1986, pp. 1069-1073.
13. M.Krízek, "On semiregular families of triangulations and linear interpolation", Appl. of Math., Vol. 36, 1991, pp. 223-232.
14. M.Krízek, "On the maximum angle condition for linear tetrahedral elements", SIAM J. Numer. Anal., Vol. 29, 1992, pp. 513-520.
15. J.-D.Müller, P.L.Roe and H.Deconinck, "A frontal approach for internal node generation in Delaunay triangulations", Int. J. Numer. Meth. Fluids, Vol. 17, 1993, pp. 241-255.
16. J.Peraire, "A finite element method for convection dominated flows", Ph.D.Thesis, Dept. Civil Eng., Univ. College of Swansea, 1986.
17. J.Peraire, K.Morgan and J.Peiró, "Unstructured mesh methods for CFD", I.C. Aero Report 90-04, 1990, Imperial College, UK.
18. J.Peiró, "A finite element procedure for the solution of the Euler equations on unstructured meshes", Ph.D.Thesis, Dept. Civil Eng., Univ. College of Swansea, 1989.
19. A.G.Stamatis and K.D.Papailiou, "An unstructured grid optimization method", in Finite Elements in Fluids, New Trends and Applications (Edited by K.Morgan, E.Oñate, J.Periaux, J.Peràire and O.C.Zienkiewicz), Pineridge Press, 1993, pp. 676-685.
20. T.E.Tezduyar, S.Aliabadi, M.Behr, A.Johnson and S.Mittal, "Parallel finite element computation of 3D flows - Computation of moving boundaries and interfaces, and mesh update strategies", Preprint 93-042, Army High Performance Computing Res. Ctr., Univ. of Minnesota, USA, 1993.
21. M.J.Vénere and D.L.Arnica, "Surface finite element mesh generation", presented at 4th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Swansea, Wales, UK, 6-8 April 1994.
22. N.P.Weatherill and O.Hassan, "Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints", Int. J. Numer. Meth. Eng., Vol. 37, 1994, pp. 2005-2039.
23. H.Zhang and J.-Y.Trépanier, "An algorithm for the optimization of directionally stretched triangulations", Int. J. Numer. Meth. in Eng., Vol. 37, 1994, pp. 1481-1497.