

A constrained optimization approach to finite element mesh smoothing

V.N. Parthasarathy

General Electric Consulting Services, Albany, NY 12205, USA

Srinivas Kodiyalam

Solid Mechanics Laboratory, General Electric Corporate R&D Center, Schenectady, NY 12301, USA

Received March 1991

Revised May 1991

Abstract. The quality of a finite element solution has been shown to be affected by the quality of the underlying mesh. A poor mesh may lead to unstable and/or inaccurate finite element approximations. Mesh quality is often characterized by the “smoothness” or “shape” of the elements (triangles in 2-D or tetrahedra in 3-D). Most automatic mesh generators produce an initial mesh where the aspect ratio of the elements are unacceptably high. In this paper, a new approach to produce acceptable quality meshes from a topologically valid initial mesh is presented. Given an initial mesh (nodal coordinates and element connectivity), a “smooth” final mesh is obtained by solving a constrained optimization problem. The variables for the iterative optimization procedure are the nodal coordinates (excluding, the boundary nodes) of the finite element mesh, and appropriate bounds are imposed on these to prevent an unacceptable finite element mesh. Examples are given of the application of the above method for 2- and 3-D meshes generated using QUADTREE/OCTREE automatic mesh generators. Results indicate that the new method not only yields better quality elements when compared with the traditional Laplacian smoothing, but also guarantees a valid mesh unlike the Laplacian method.

Introduction

The advent of powerful desktop workstations has spear-headed the development of many commercial mechanical computer-aided engineering (MCAE) tools. The intended purpose of many of these tools is mainly to automate the design cycle. Yet another objective is to minimize the time required to completely design a product. The basis for making many design decisions is the interpretation of the results obtained from an appropriate numerical analyses such as the finite element analysis on the computer representation of the actual part. Fundamental to the analysis is the creation of a geometric model and discretization of the geometric model into elements of desired size. Developments in the interactive graphics arena have greatly simplified the model creation task. The model discretization task, commonly known as mesh generation, has long been recognized as one of the key technologies which is necessary for automating the numerical analysis process and hence, minimizing the design cycle time. Extensive research is being pursued in fully automatic mesh generation due to its potential ability to attack increasingly complex problems and phenomena. Among the popular fully automatic mesh generation techniques are the quadtree based techniques in two dimensions [1], octree based techniques in three dimensions [2], Delaunay Triangulation in both dimensions [3–5], and their combination [6,7].

Research in the theory of finite element analysis has placed increased importance on the size and shape of the elements generated by a mesh generator [8]. The quality of a finite element solution, and hence the final design, has been shown to be affected by the quality of the underlying mesh. A poor mesh may lead to unstable and/or inaccurate finite element approximations. This stringent requirement on the size and shape has been a contributing factor in the selection of a particular mesh generator for a given analysis application. The importance of mesh quality cannot be emphasized more, when one observes efforts in a new direction in mesh generation, where the resulting mesh quality forms the basis of the technique [9].

In this paper, mesh quality measures used in practice are presented. Next, existing mesh quality improvement techniques are discussed. Then, a new approach to mesh quality improvement—a constrained optimization approach—is presented. Finally, the application of this new method to some complex 2-D and 3-D problems is illustrated. Although the applications have been restricted to 2-D triangular and 3-D tetrahedral meshes, the proposed method is independent of element types.

Mesh quality measures

Mesh quality is often characterized by the “smoothness” or “shape” of the elements (triangles/quadrilateral in 2-D or tetrahedra/hexahedra in 3-D). Various measures have been developed in the literature to quantify the smoothness of a mesh. For example, in a planar quadrilateral mesh, the aspect-ratio, i.e., the ratio of maximum side to minimum side is commonly used. In addition to the aspect-ratio, the change in elemental areas and the angles subtended at a node point are also used. Some researchers have used a normalized, linear weighted combination of these measures. In the case of planar triangular meshes, in addition to the aspect-ratio, a comprehensive measure of normalized ratio of the circumscribed-circle radius to the inscribed-circle radius of the triangle (see Fig. 1) is commonly used. Normalizing is done with respect to the corresponding ratio of an equilateral triangle. In a 3-D hexahedral mesh, the volume of the elements and the orthogonality of the element faces coincident at a node are considered. For a 3-D tetrahedral mesh, since there exists no orthogonality measure the evaluation of the mesh quality has mainly depended on a normalized ratio of the circumscribed-sphere radius to the inscribed-sphere radius. Nevertheless, other ratios (see Fig. 2) that have been used in conjunction with this are the normalized ratio of the circumscribed-sphere to the maximum side of the tetrahedron and the normalized ratio of maximum side of the tetrahedron to the inscribed-sphere radius [8]. Once again, all normalizations are carried out with respect to similar quantities of an equilateral tetrahedron.

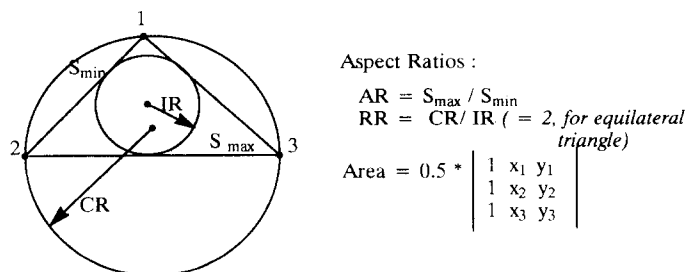
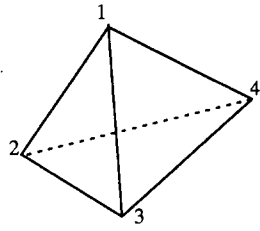


Fig. 1. Quality measures for 2-D triangular elements.



Aspect Ratios :

$$AR_1 = S_{\max} / S_{\min}$$

$$AR_2 = CR / S_{\min}$$

$$RR = CR / IR \quad (= 3, \text{ for equilateral tetrahedron})$$

$$\text{where } IR = \frac{4 * \text{Volume}}{\sum \text{Triangle_Face_Area}_i}$$

$$\text{Volume} = 1/6 * \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$

Fig. 2. Quality measures for 3-D tetrahedral elements.

A “thin” element is formed when the lengths (areas) of any two sides of a triangle (of any three faces of a tetrahedron) is much greater than the length of the third side of the triangle (area of the fourth face of the tetrahedron). A “flat” element is formed when the lengths (areas) of the three sides (four faces) are of the same order, but the three (four) forming nodes are near-collinear (near-coplanar). Both, the thin and flat elements are considered “bad” and are often the targets for many mesh improvement schemes. In this work, it was decided to use the ‘radius-ratio’ criterion for evaluating the aspect-ratio of triangles/tetrahedrons due to its unique ability to characterize, both “flat” and “thin” triangles/tetrahedrons, as having a high aspect ratio value, while other measures were able to characterize only either one.

Topological and geometric mesh validity

A finite element mesh is considered *topologically valid* when all finite element entities in the mesh are classified correctly against the domain geometry. For instance, self-intersecting meshes result in a topologically invalid mesh. Topological validity assurance, performed subsequent to meshing, has been found necessary as some meshing algorithms (e.g., Delaunay algorithm) do not recognize the domain topology information. More details on how to detect and resolve topologically incompatible meshes can be found in [6].

A finite element mesh is considered *geometrically valid* only when *all* elements in the mesh have a non-zero, positive area/volume, given that a consistent counter-clockwise ordering of nodes has been followed for all 2-D elements (triangles) in the mesh. A mesh can be geometrically valid while its quality in terms of the discussion in the previous section can be poor. In an effort to make a poor mesh better some algorithms render the mesh geometrically invalid. Such meshes are a result of zero or negative area/volume elements, frequently caused by an illegal placement of a node with respect to its neighbors. For example, a node placed outside the domain can cause a negative element. Another common cause (in 3 – D) is the placement of a node inside a neighboring element, essentially creating an inside-out element. This condition has been observed mainly amongst nodes lying in the interior of the domain.

To be suitable for any numerical analysis, a finite element mesh needs to be topologically *and* geometrically valid. Though well shaped elements is a necessary condition for any mesh, requirements on the element size/shape is largely dictated by the particular physical phenomena being investigated. The concern of this paper is to address new methods to improve the mesh quality of a geometrically valid/invalid, but topologically valid mesh, using the measures described in the previous section as the basis, while ensuring/restoring its geometric validity.

Smoothing or mesh quality improvement

In general, it is preferred that the elements in a mesh be as equi-angular as possible with a gradual change in area/volume. Nodes which result in a bad patch of elements have been referred to as ‘irregular’ nodes [9]. Stated otherwise, smoothness of a mesh is a function of position of the nodal points. In the previously cited fully automatic mesh generation, the definition of ‘automatic’ encompasses the automatic generation of the interior nodes. These algorithms can be domain sensitive since the same discretizing algorithm is used irrespective of the domain being discretized. Therefore, it is commonly realized that automatic mesh generators *can* produce an initial mesh which has many irregular nodes and hence elements with unacceptably high aspect ratios. In order to improve the overall mesh quality, the nodes are repositioned so as to reduce its degree of irregularity. This repositioning operation is called mesh smoothing and is an integral part of essentially all mesh generators.

Literature reveals only one oft-used technique for mesh smoothing—the Laplacian smoothing. In a 2-D mesh patch shown in Fig. 3, this technique iteratively moves every node to the centroid of the surrounding polygon. In some examples, as will be seen later, this procedure deteriorates mesh quality in one region, while improving in other regions of the domain. More importantly, the primary disadvantages of Laplacian smoothing are: (i) the potential movement of the nodes to the exterior of the domain in the presence of non-convex local geometry; and (ii) positioning a node inside the volume of some neighbor element in 3-D meshes. Thus, Laplacian smoothing *can* render a mesh unsuitable for analysis. Several other techniques developed to avoid this problem are mere derivatives of Laplacian smoothing. They include: (i) area-weighted Laplacian smoothing where a node is moved toward larger area elements in the patch; and (ii) length-weighted Laplacian smoothing where the node is moved toward the larger contributing peripheral node. Although these modifications have been somewhat successful in ameliorating the problems, they do not guarantee success for any arbitrary mesh.

Clearly, for guaranteed success, the solution lies in restricting a node’s movement, both in distance and direction. Constrained movement of a node requires evaluation of many geometric quantities and topological information of the nodes and elements connected to the node being smoothed. In terms of quality–cost trade-offs, this translates to the fact that one has to pay an increased computational expense for an “intelligent” mesh quality improvement algorithm. Efforts in this direction are also seen in [10].

A well-known geometric check is the evaluation of the area of the elements—a negative value, as mentioned before, indicates that the node has been positioned outside the domain. Therefore, a necessary constraint on the smoothing operation is that all elements of a mesh must have a positive area/volume. Also, it has to reposition the nodes in such a way as to minimize/maximize the particular mesh quality measure being used. A closer look suggests the

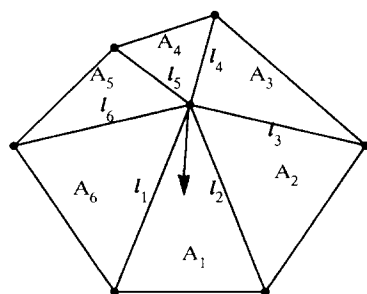


Fig. 3. Laplacian smoothing and its variants.

$$\text{Simple :} \\ x_i^{\text{new}} = 1/n \sum x_i^{\text{old}}$$

$$\text{Area-Weighted :} \\ x_i^{\text{new}} = \frac{1}{n} \frac{\sum A_i x_i^{\text{old}}}{\sum A_i}$$

$$\text{Length-Weighted :} \\ x_i^{\text{new}} = \frac{1}{n} \frac{\sum l_i x_i^{\text{old}}}{\sum l_i}$$

formulation of mesh smoothing as a numerical optimization problem, by stating that a smooth mesh has a minimal change in elemental shape and non-negative area/volume at a given node. Impressive results using a similar approach, for structured computational grid generation for computational fluid dynamics problems, were reported in [11].

Mesh smoothing as an optimization problem

Given an initial topologically-valid mesh (nodal coordinates and element connectivity), a “smooth” final mesh is obtained by solving a constrained optimization problem. The nonlinear constrained optimization problem is stated in the following form: Find the vector of problem parameters, \mathbf{D} , that will

$$\text{Minimize } F(\mathbf{D}) = \sqrt{(1/\text{nel}) \sum_k^{\text{nel}} \text{AR}_k * \text{AR}_k} \quad \text{objective function}$$

Subject to:

$$g_j(\mathbf{D}) \leq 0, \quad j = 1, m \quad \text{inequality constraints} \quad (1)$$

$$d_i^l \leq d_i \leq d_i^u, \quad i = 1, n \quad \text{bounds on design variables}$$

where nel is the number of elements; m is the number of inequality constraints; and n is the number of design variables ($= 2$ or $3 \times$ number of nodes)

In this particular problem, the root mean square (RMS) value of the “aspect-ratio” (AR) of all the elements in the initial mesh is treated as the objective function to be minimized, with inequality constraints, $g_j(\mathbf{D})$, imposed on the area of each individual element (volume in the case of 3-D mesh) to be greater than a positive minimum. For example, an area inequality constraint for element 12 will be calculated as follows:

$$g_{12}(\mathbf{D}) = \delta - A_{12} < 0.0,$$

where A_{12} is the area of element 12, and δ is a preset positive value, specified by the user as the minimum area of any element in the mesh. In the normalized form $g_{12}(\mathbf{D})$ is evaluated as $(1 - A_{12}/\delta) < 0.0$.

The particular aspect-ratio measure used in this paper is the radius-ratio (RR), for reasons explained in the previous section and whose formulae are given in Figs. 1 and 2. The variables for the iterative optimization procedure are the nodal coordinates, i.e., $x_i, \bar{y}_i (z_i)$ of each node, (excluding the boundary nodes) of the finite element mesh. Appropriate bounds are imposed on these to prevent an unacceptable finite element mesh.

The constrained minimization is performed using the modified method of feasible directions algorithm [12]. The vector of design variables, \mathbf{D} , is updated using the formula:

$$\mathbf{D}^q = \mathbf{D}^{(q-1)} + \alpha^* \mathbf{S}^{(q)} \quad (2)$$

where, q is the iteration number, \mathbf{S}^q is the vector of search direction and α^* is a scalar move parameter. The usable-feasible search direction is computed by solving the following subproblems:

(i) In the case when there are no active or violated constraints, a Fletcher–Reeves conjugate direction algorithm is used to find the search direction as:

$$\mathbf{S}^q = -\nabla F(\mathbf{D}^{q-1}) + \beta \mathbf{S}^{q-1} \quad \text{where} \quad \beta = \frac{|\nabla F(\mathbf{D}^{q-1})|^2}{|\nabla F(\mathbf{D}^{q-2})|^2} \quad (3)$$

(ii) In the case where some of the constraints are active, but not violated, it is necessary to find a search direction \mathbf{S}^q , that minimizes the objective function without violating any of the currently active constraints. An inequality constraint is considered active if it is more positive than -0.05 and is treated as violated if it is greater than 0.003 . It is important to note that constraints, $g_j(\mathbf{D})$, are normalized to the order of magnitude of unity. The direction finding problem is posed as:

Find the search vector, \mathbf{S}^q , that will:

$$\text{Minimize } \nabla F(\mathbf{D}^{q-1}) \cdot \mathbf{S}^q$$

Subject to:

$$\begin{aligned} \nabla g(\mathbf{D}^{q-1}) \cdot \mathbf{S}^q &\leq 0 \quad j \in J \\ \mathbf{S}^q \cdot \mathbf{S}^q &\leq 1 \end{aligned} \quad (4)$$

where J is the set of active constraints.

(iii) In the case when there are violated constraints, the direction finding problem is posed as:

Find the search vector, \mathbf{S}^q , and artificial variable, W , that will:

$$\text{Minimize } \nabla F(\mathbf{D}^{q-1}) \cdot \mathbf{S}^q - \phi W$$

Subject to:

$$\begin{aligned} \nabla g(\mathbf{D}^{q-1}) \cdot \mathbf{S}^q + \theta_j W &\leq 0 \quad j \in J \\ \mathbf{S}^q \cdot \mathbf{S}^q + W^2 &\leq 1 \end{aligned} \quad (5)$$

where θ_j is called the “push-off” factor since this determines the distance from the violated constraint and ϕ is initially a small positive number that is gradually increased in order to bring the design to feasible region. J denotes the set of critical constraints at a design point \mathbf{D} . More details of the Fletcher–Reeves algorithm and the modified method of feasible directions can be obtained from Ref. [12].

In the present study, a multi-criteria objective function is also considered to produce acceptable quality meshes. The minimization problem determines Pareto optimal [13] solutions of the problem by considering a weighted performance index of different objectives. Pareto optimal solutions can be obtained by solving the minimization problem;

$$F(\mathbf{D}^{\text{opt}}) = \min_{\mathbf{D}} (\mu_1 f_1(\mathbf{d}) + \mu_2 f_2(\mathbf{d})) \quad (6)$$

subject to constraints on the area (volume in the case of 3-D problems) of each individual element. In eqn. (6),

$$f_1(\mathbf{d}) = \sqrt{(1/\text{nel}) \sum_k^{\text{nel}} (\mathbf{A}\mathbf{R}_k * \mathbf{A}\mathbf{R}_k)} \quad (7)$$

$$f_2(\mathbf{d}) = \sqrt{(1/\text{nel}) \sum_k^{\text{nel}} (g_k * g_k)} \quad (8)$$

$$\mu_1, \mu_2 \geq 0, \quad \mu_1 + \mu_2 = 1.0 \quad (9)$$

To each set, the weighting factors corresponds a different compromise solution of the problem involving trade-offs between the different objective functions.

Implementation

The iterative mesh optimization program architecture is shown in Fig. 4. Model geometry is represented through an in-house geometry and topology software utility system called TAGUS. At General Electric's Corporate Research and Development Center, the 3-D octree based automatic mesh generator, OCTREE, capable of meshing arbitrarily complex 0-D, 1-D, 2-D and 3-D domains, has been developed and successfully implemented [6]. The mesh generator works on the principle of (i) recursive spatial decomposition to discretize a 3-D domain into octants and (ii) a 3-D point-set meshing technique based on Delaunay principles. The generated mesh data is stored in a hierarchical data structure called Finite Element Data Structure, FEDS. A rich set of operators is available to manipulate and query the finite element data and its associativity. In the present work, an initial un-smoothed mesh stored in the FEDS format forms the starting point. Additional algorithms have been implemented to compute the necessary geometric details of the finite element entities such as the area of triangular faces, volume of tetrahedra, the circumscribed- and the inscribed-radius of triangles and tetrahedra. These details are then used to compute the single/multi-objective and constraint functions, referred in eqns. (1) and (6). The numerical optimizer, DOT [14], is then used to solve the constrained optimization problem. The optimized nodal locations of the smoothed mesh are finally stored in FEDS for generating input to the analysis program.

Results and discussion

Three examples were successfully tested with the implementation described earlier. Only interior nodes of each example were considered for optimization. Important details of all the examples are furnished in Table 1. It is seen that smoothing consumes more than 50% of the total meshing process. The large time is clearly offset by a guaranteed valid mesh. Moreover, such timings are typical for optimization-based methods, as is seen from [10].

Example 1: 2-D simply connected domain—square

The first example was a uniform square whose initial mesh is shown in Fig. 5(a). The optimized mesh (i) with ratio of maximum to minimum side lengths; and (ii) radius ratio for

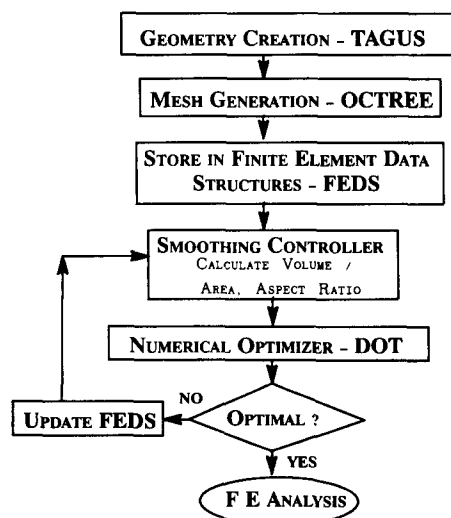


Fig. 4. Mesh optimizing program implementation.

Table 1
Example details

Example	No. of design variables	No. of constraints	CPU time (seconds)	Smoothing as a % of total time
Square	98	124	4.0	62.3
Square with holes	90	160	13.4	58.5
Cube-in-cube	114	580	1902	> 95.0

computing the aspect ratio is shown in Figs. 5(b) and 5(c) respectively. Multi-objective optimization as described earlier in this paper was also used to optimize the initial mesh. The resulting smoothed mesh, shown in Fig. 5(d), was identical to the one shown in Fig. 5(c). The convergence history for this example is shown in Fig. 10(a). The CPU time for this example was 4 seconds on a DECstation 3100.

Example 2: 2-D multi-connected domain—square with holes

The second test example was a 2-D multi-connected geometry—an application from injection molding. The initial mesh and its corresponding optimized mesh is shown in Figs. 6(a) and 6(b) respectively. Zones in the initial mesh that need improvement are the regions near the holes. Mesh quality improvement is obvious from the optimized mesh, shown in Fig. 6(b).

A simple-Laplacian-smoothing on the initial mesh resulted in a mesh shown in Fig. 7(a). From Figs. 6(b) and 7(a), it is obvious that for the given initial mesh, the optimizer had improved the mesh quality near the interior boundaries, where Laplacian smoothing de-

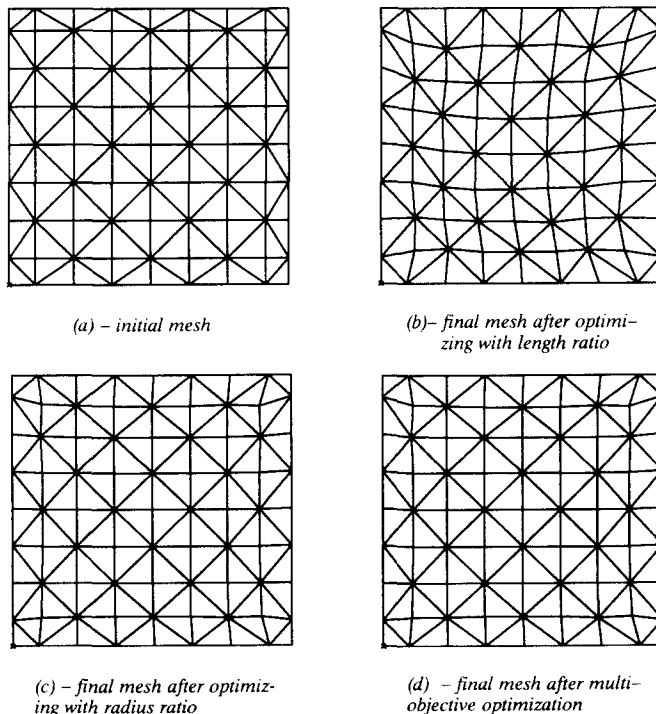


Fig. 5. 2-D simply connected example—square.

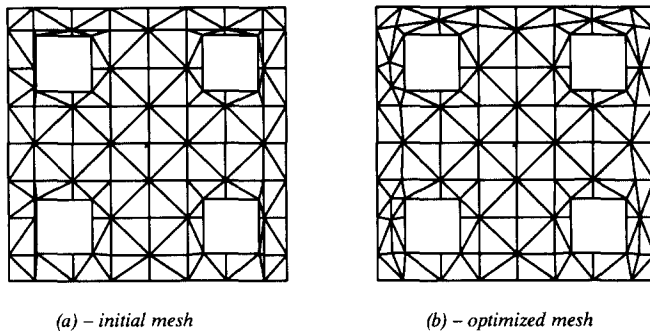


Fig. 6. 2-D multi-connected example—square with holes.

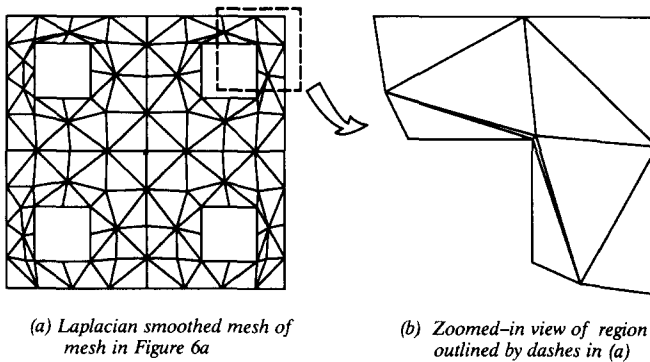


Fig. 7. 2-D multi-connected example—Laplacian smoothing.

teriorated the mesh quality. The result of using this Laplacian smoothed mesh as input to the optimizer is shown in Fig. 8(b).

Finally, an experiment with an *invalid* mesh as input to the optimizer was also performed. An interior node in the mesh shown in Fig. 7(a) was manually positioned outside the domain as shown enlarged in Fig. 8(a). This resulted in an *invalid* mesh as two elements had non-positive areas (or Jacobian). The complete optimized mesh for this experiment is shown in Fig. 8(b). This experiment demonstrated the inherent ability of the present scheme to *untangle* overlapped meshes.

All optimizations were carried out with normalized radius ratio as the measure of aspect ratio. A positive minimum area constraint (δ) of 0.1% of the domain area was imposed.

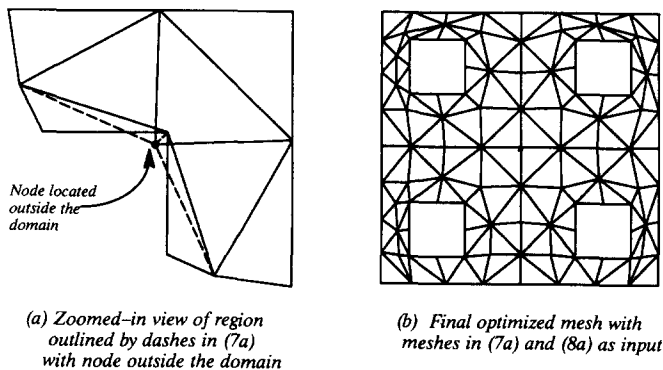


Fig. 8. 2-D multi-connected example—untangling of invalid mesh.

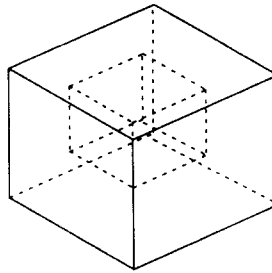


Fig. 9. 3-D example—cube-in-cube.

Maximum radius-ratio in the initial mesh was 22.489, whereas a ratio of 1.0 is considered good. Maximum aspect-ratio in the optimized mesh was 2.7103. Multi-objective optimization resulted in identical results. The convergence history for this example is shown in Fig. 10(b). CPU timing for this example was 13.4 seconds.

Example 3: 3-D Multi-connected domain—cube-in-cube

To demonstrate the smoothing scheme's application to 3-D, a cube within a cube, shown in Fig. 9, was meshed and optimized. The initial mesh had elements with a maximum normalized radius-ratio of 18.22 and a minimum volume of 0.0003% of the domain volume. Such small volume elements typically pose problems in numerical analyses. A simple Laplacian smoothing resulted in a geometrically invalid mesh, unsuitable for analysis. On the contrary, the optimized mesh was not only valid, but also showed improved characteristics in terms of both a lower maximum normalized radius-ratio of 4.88 and a higher minimum volume of 0.1% of the domain volume. The CPU time for this example was 31.7 minutes. Once again, the positive minimum volume constraint (δ) was set to 0.1% of the domain volume. The convergence history for this example is shown in Fig. 10(c). Unlike the previous examples, meshes before and after smoothing are not shown as it is extremely difficult to visually comprehend a cluster of tetrahedrons in the domain interior—plotted using a non-hidden line algorithm.

The large CPU time may lead to us false conclusions about the time growth rates. The CPU time in itself can be influenced by the following:

- (i) number of constraints, i.e., elements in the mesh;
- (ii) number of design variables, i.e., (nodes to be smoothed * dimension of the domain);
- (iii) 'worseness' or 'badness', i.e., the maximum aspect-ratio and minimum volume;
- (iv) the more than an order of magnitude increase in computational expense for function evaluation (calculation of tetrahedra volumes, the inscribed- and the circumscribed-sphere radius computations).

The optimization time *per se* in the overall smoothing time is the summation of just that portion after the function evaluation is performed and the optimizer is invoked and before another function evaluation in the iterative scheme. The time for function evaluation does grow linearly with the number of constraints. Also, the function evaluation for a single constraint is dependent on the dimension of the problem. For instance, in 3-D, the cost/constraint is at least many times more expensive than in 2-D. Therefore, to correctly predict the growth rate with respect to the number of nodes, more investigation is necessary. Moreover, the worseness of the input mesh—which is independent of the number of nodes—can affect the total time positively or adversely.

Nevertheless, it is clear that a strategy to minimize the number of constraints in higher dimensional problems is necessary. Since the first version of this paper, this has been done by

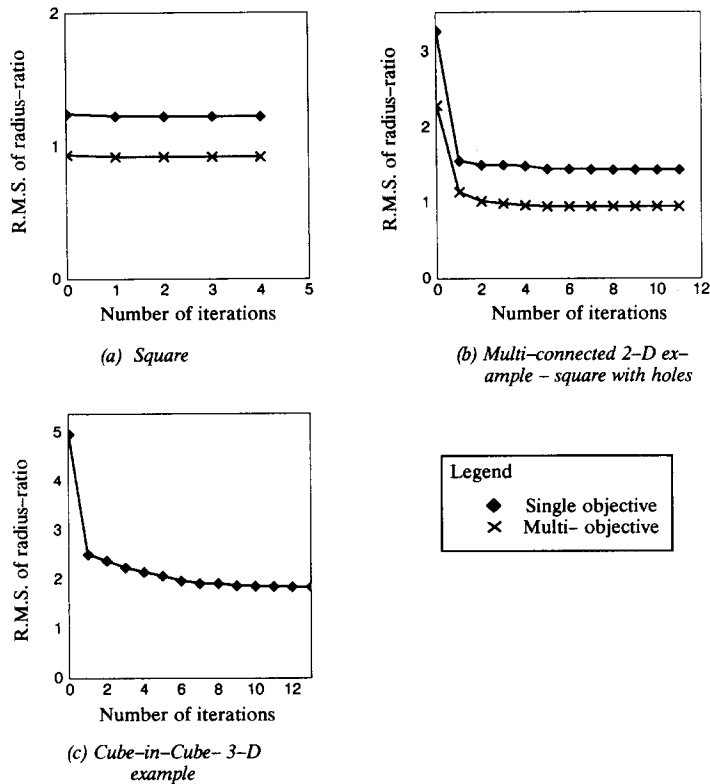


Fig. 10. Optimization convergence history for various examples.

solving a local optimization problem on a node-by-node basis for 3-D problems. The results thus far have been encouraging and will be published later.

Conclusions

A new method, based on constrained optimization, to produce acceptable quality meshes from an initial mesh is proposed. The method is demonstrated using some simple and complex 2-D/3-D examples. The new method shows promise as an alternative to the conventional Laplacian smoothing techniques in its ability to produce comparably better quality meshes and more importantly untangle geometrically invalid initial meshes. The method requires increased computational efforts for global 3-D problems. Further investigation is required to reduce the computational expense.

Acknowledgements

The authors are thankful to Mr. Peter M. Finnigan, Manager, Mechanical Design Methods Program at the GE-CRD, for his continued support and encouragement to pursue this work. The authors would also like to thank the reviewer of the paper for bringing to our attention some areas of the initial version of the paper which needed more clarity.

References

- [1] P.L. BAEHMANN, S.L. WITTCHEN, M.S. SHEPHARD, K.R. GRICE and M.A. YERRY, "Robust geometrically based, automatic, two dimensional mesh generation", *Int. J. Num. Methods Eng.*, **24**, pp. 1043–1078, 1987
- [2] M.S. SHEPHARD and M.A. YERRY, "Finite element mesh generation for use with solid modeling and adaptive analysis", in: *Solid Modeling with Computers*, edited by M.S. PICKETT and J.W. BOYSE, Plenum Press, pp. 53–80, 1984.
- [3] T.J. BAKER, "Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation", *Eng. with Computers*, **5**, pp. 161–171, 1989.
- [4] J.C. CAVENDISH, D.A. FIELD and W.H. FREY, "An approach to automatic three-dimensional finite element mesh generation", *Int. J. Num. Methods Eng.*, **21**, pp. 329–347, 1985.
- [5] V.N. PARTHASARATHY, Solution contour based adaptive triangular grid generation, Master's Thesis, Dept. of Mech. Eng., University of Miami, Florida, 1989.
- [6] C.M. GRAICHEN, A.F. HATHAWAY and V.N. PARTHASARATHY, The OCTREE Theoretical Report, GE-CRD Report, Schenectady, NY, January 1991.
- [7] W.J. SCHROEDER and M.S. SHEPARD, "A combined octree/Delaunay method for fully automatic 3-D mesh generation", *Int. J. Num. Methods Eng.*, **29**, pp. 37–55, 1988.
- [8] T.J. BAKER, "Element quality in tetrahedral meshes", *Proc. 7th Int. Conf. on Finite Element Methods in Flow Problems*, Huntsville, AL, April 3–7, 1989.
- [9] T.D. BLACKER and M.B. STEPHENSON, Paving: A New Approach to Automated Quadriateral Mesh Generation, Sandia National Laboratories Report, No. SAND90-0249, 1990.
- [10] M.S. SHEPHARD, H.L. COUGNY and M.K. GEORGES, Explicit Node Point Smoothing within Octree, Report no. 10-1990, SCOREC, RPI, Troy, NY, 1990.
- [11] S.R. KENNON and G.S. DULIKRACH "Generation of computational grids using optimization", *AIAA J.*, **24** (7) pp. 1069–1073, 1986.
- [12] G.N. VANDERPLAATS, *Numerical Optimization Techniques for Engineering Design: with Applications*, McGraw-Hill, New York, 1984.
- [13] S. ADALI, "Pareto optimal design of beams subjected to support motions", *Comput. Struct.*, **16**, pp. 297–303, 1983.
- [14] G.N. VANDERPLAATS, *DOT User's Manual*, Version 2.0, VMA Engineering, Goleta, CA, 1990.