Tetrahedral Mesh Improvement Using Swapping and Smoothing

Lori A. Freitag

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439
Phone: 630-252-7246
Fax: 630-252-5986
freitag@mcs.anl.gov

Carl Ollivier-Gooch

Department of Mechanical Engineering
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
Phone: 604-822-1854
Fax: 604-822-2403
cfog@mech.ubc.ca

Abstract. Automatic mesh generation and adaptive refinement methods for complex three-dimensional domains have proven to be very successful tools for the efficient solution of complex applications problems. These methods can, however, produce poorly shaped elements that cause the numerical solution to be less accurate and more difficult to compute. Fortunately, the shape of the elements can be improved through several mechanisms, including face- and edge-swapping techniques, which change local connectivity, and optimization-based mesh smoothing methods, which adjust mesh point location. We consider several criteria for each of these two methods and compare the quality of several meshes obtained by using different combinations of swapping and smoothing. Computational experiments show that swapping is critical to the improvement of general mesh quality and that optimization-based smoothing is highly effective in eliminating very small and very large angles. High-quality meshes are obtained in a computationally efficient manner by using optimization-based smoothing to improve only the worst elements and a smart variant of Laplacian smoothing on the remaining elements. Based on our experiments, we offer several recommendations for the improvement of tetrahedral meshes.

Keywords. Mesh Improvement, Local Reconnection, Mesh Smoothing, Optimal Smoothing

1. Introduction

The use of unstructured finite element and finite volume solution methods is increasingly common for application problems in science and engineering. Regardless of the particular solution technique employed, the computational domain must be decomposed into simple geometric elements. This decomposition can be accomplished by using available automatic mesh generation tools. Unfortunately, meshes generated in this way can contain poorly shaped or distorted elements, which cause numerical difficulties during the solution process. For example, we know that as element dihedral angles become too large, the discretization error in the finite element solution increases;[1] and as angles become too small, the condition number of the element matrix increases.[2] Thus, for meshes with highly distorted elements, the solution is both less accurate and more difficult to compute. This problem is more severe in three dimensions than in two dimensions, because tetrahedra can be distorted to poor quality in more ways than triangles can. Compared with triangular meshes, tetrahedral meshes tend to have a larger proportion of poor-quality elements and to have elements that are more severely distorted.

Algorithms for unstructured mesh improvement fall into three basic categories:

1. point insertion/deletion to refine or coarsen a mesh or to improve the local length scale of the mesh,[3, 4, 5, 6]

2. local reconnection to change mesh topology by face or edge swapping for a given set of vertices,[6, 7, 8, 9] and

3. mesh smoothing to relocate mesh points to improve mesh quality without changing mesh topology.[10, 11, 12]

In this article, we follow a two-pronged approach to improve the quality of tetrahedral meshes, swapping mesh faces and edges to improve connectivity and smoothing vertex locations to improve tetrahedron shape. Face- and edge-swapping techniques are widely used, and we give only a brief overview of the methods used. We apply swapping both as an initial step in mesh improvement and in a targeted way to remove the poorest-quality tetrahedra from the mesh. In this context, we define "poor-quality" tetrahedra as those with dihedral angles that are among the smallest or largest in the mesh or with solid angles among the largest in the mesh; such tetrahedra are candidates for removal.

The most common approaches to mesh smoothing are variants on Laplacian smoothing.[13] While these smoothers are often effective, they operate heuristically with no effort to locate points specifically to improve some quality measure. In this article, we present an optimization-based smoothing algorithm for tetrahedral meshes that is effective in eliminating extremal angles in the mesh. For five test cases, we show that the highest quality meshes are obtained by using a combination of swapping and optimization-based smoothing. In addition, we show that meshes of comparable high quality can be obtained at a low computational cost by combining a variant of Laplacian smoothing with optimization-based smoothing.

The remainder of the article is organized as follows. In Section 2, we describe the swapping techniques that we use. In Section 3, we describe a mesh smoothing algorithm using

local optimization. We then present the results of numerical experiments on several test meshes. Mesh quality improvement by swapping, vertex smoothing, and combinations of swapping and smoothing is presented, and several recommendations are offered. Finally, in Section 5, we offer concluding remarks and directions for future research.

2. Local Mesh Reconfiguration Techniques

Local mesh reconfiguration techniques change the connectivity of part of a simplicial mesh to improve mesh quality. These techniques can be divided into two classes: face swapping and edge swapping.

2.1 Face Swapping

Face swapping reconnects the tetrahedra separated by a single interior face. Each interior face in a tetrahedral mesh separates two tetrahedra made up of a total of five points. A large number of non-overlapping tetrahedral configurations are possible with these five points, but only two can be legally reconnected. These two cases are shown in Figure 1. On the left is a case in which either two or three tetrahedra can be used to fill the convex hull of a set of five points. Switching from two to three tetrahedra requires the addition of an edge interior to the convex hull. On the right of the figure is a configuration in which two tetrahedra can be exchanged for two different ones. The shaded faces in the figure are coplanar, and swapping exchanges the diagonal of the coplanar quadrilateral. The two coplanar faces must either be boundary faces or be backed by another pair of tetrahedra that can be swapped two for two. Otherwise, the new edge created by the two for two swap will not be conformal.

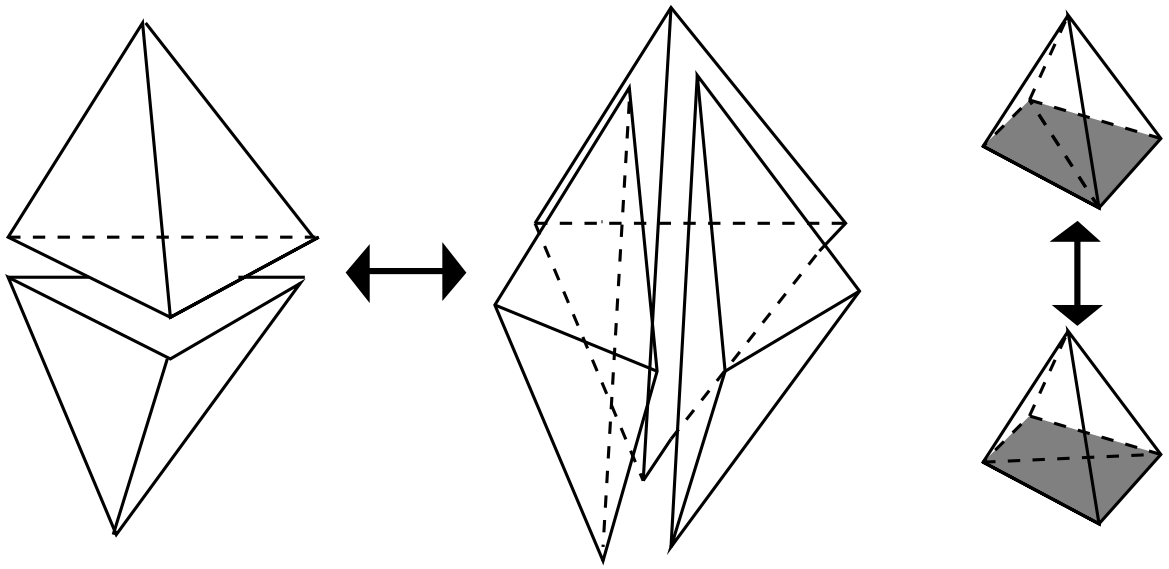

Figure 1: Swappable configurations of five points in three dimensions

Because each reconfigurable case has only two valid configurations,[14] a quick comparison to find the one with the higher quality is possible. If the configurations are of equal quality,

we select the two-tet configuration when choosing between two- and three-tet configurations, and we choose not to swap in the two for two reconfiguration case.

2.2 Edge Swapping

Edge swapping reconfigures N tetrahedra incident on an edge of the mesh by removing that edge and replacing the original N tetrahedra by $2N-4$ tetrahedra. The reconfiguration is performed only if every new tetrahedron has better quality than the worst of the N original tetrahedra. As an example, consider the replacement of five tetrahedra incident on an edge with six. The left side of Figure 2 shows five tetrahedra incident on edge TB, which is normal to the paper. The five original tetrahedra are (01BT), (12BT), (23BT), (34BT), and (40BT). The right side of the figure shows one possible reconfiguration of this submesh into six tetrahedra, (012T), (021B), (024T), (042B), (234T), and (243B). The final configuration is uniquely specified by listing the "equatorial triangles", those that are not incident on either of vertices T and B. In this case, those triangles are (012), (024), and (234).

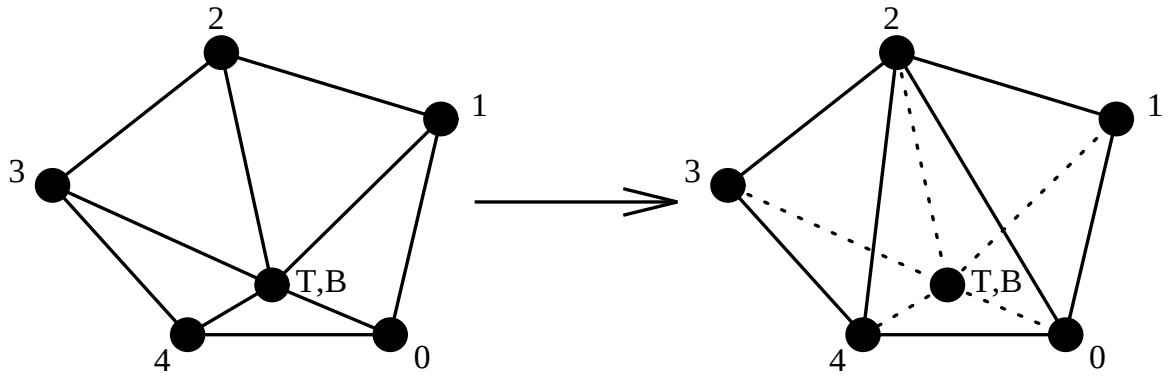

Figure 2: Edge swapping from 5 tetrahedra to 6

In principal, edge swapping could be used to replace, for example, 12 tetrahedra with 20, but in practice we have found that the number of transformations that improve the mesh declines dramatically with increasing N. In particular, for practical cases 7-for-10 transformations are rare, and consequently we have not investigated these techniques for $N > 7$.

The primary challenge for efficient implementation of edge swapping techniques is to determine which reconfiguration of the N original tetrahedra has the highest quality, where the quality of a configuration is equal to the lowest quality of its tetrahedra. We could determine the lowest quality tetrahedron in each configuration by computing the quality of each tetrahedron for that configuration, then repeat this process for each configuration. However, as Table 1 shows, the number of unique tetrahedra is never larger than and can be much smaller than the number of configurations times the number of tetrahedra in each. Therefore, to implement edge swapping efficiently, we evaluate the quality of each unique tetrahedron only once and store that value for use when evaluating the configuration quality.*

*Tetrahedra with negative volume are assigned an arbitrarily large negative quality.

A further efficiency gain is possible by noting that tetrahedra in the final configurations appear in pairs above and below "equatorial" triangles, such as tetrahedra (012T) and (021B) in the example above. If the quality of (012T) is worse than that of the original configuration, we know that (012T) and (021B) can not be part of the final configuration; therefore, we need not evaluate the quality of tetrahedron (021B).

Table 1: Number of unique tetrahedra and possible configurations for edge swapping

Tets Before	Tets After	Config-urations	Tets × Configs	Unique Tets
3	2	1	2	2
4	4	2	8	8
5	6	5	30	20
6	8	14	112	40
7	10	42	420	70

Another key implementation issue is storage of all possible new configurations. As noted above, these configurations can be specified by their "equatorial" triangles. However, for convenience we choose to store the connectivity information for the new submesh — face-to-vertex, face-to-cell, and cell-to-face data. This is a substantial amount of data per configuration,[†] and code testing must verify that the data is entered correctly. To reduce the size of this task, we take advantage of the fact that configurations can be grouped into topological classes each represented by a *canonical configuration*. For example, in the case of 5-for-6 swapping shown in Figure 2, all possible reconfigurations have the same topology as the case shown. For each canonical configuration, we store connectivity information for the new configuration and the number of unique rotations. Rotated configurations are easily derived from the canonical configuration. Figure 3 shows the equatorial triangles in the canonical configurations for $4 \leq N \leq 7$, including the number of unique rotations for each.

We use edge swapping in two ways. The first is as a supplement to face swapping. Consider again the configuration of Figure 2. Face swapping will be unsuccessful for face (0TB) because the submesh formed by tetrahedra (0BT1) and (4BT0) is not convex. If the tetrahedron (1BT4) existed, then the face swapping routines would consider a 3-for-2 swap. Since this is not the case, edge swapping is invoked to determine whether removing edge (TB) is advantageous. After edge swapping, each face of the submesh is tested to determine whether further improvements in the mesh can be made with face swapping.

We also use edge swapping in a separate procedure specifically designed to remove poor quality tetrahedra. This procedure, called BATR for *BA*d *T*etrahedron *R*emoval, examines each tetrahedron in the mesh and attempts to remove those with large or small dihedral angles or large solid angles. In this context we define limits on good angles as follows.

- A *large dihedral angle* is larger than the greater of 150° and the maximum dihedral angle in the mesh minus 20°.

[†] We represent it using $23N - 56$ integers.

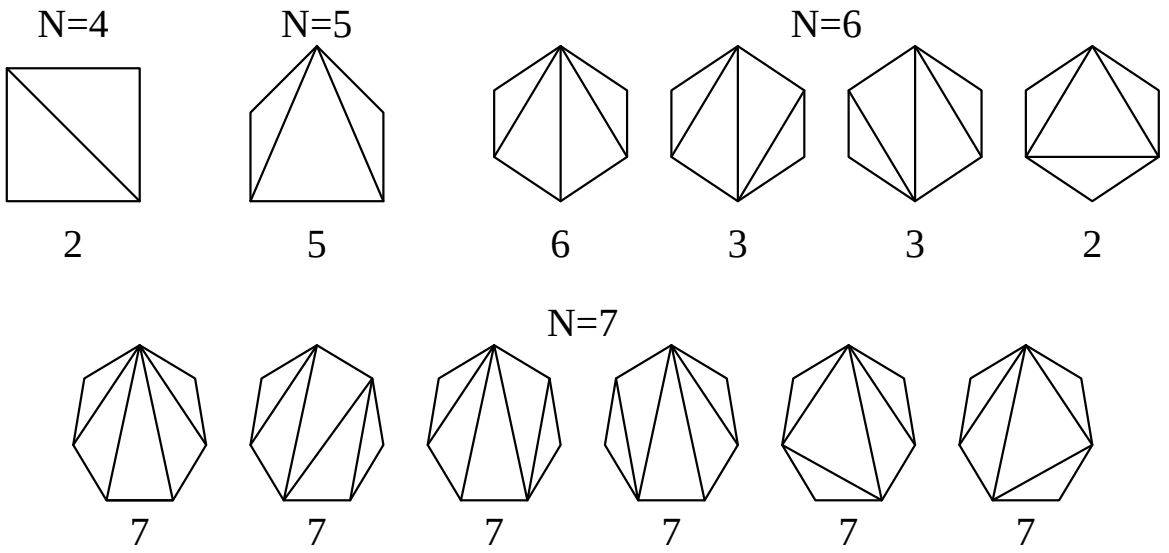

Figure 3: Equatorial triangles after edge swapping in the canonical configurations for $4 \leq N \leq 7$, including the number of unique rotations for each

- A *small dihedral angle* is smaller than the lesser of 30° and the minimum dihedral angle in the mesh plus 10°.
- A *large solid angle* is larger than the greater of 240° and the maximum solid angle in the mesh minus 60°.

These definitions imply that we always seek to remove the tetrahedra with the worst quality measures while keeping a reasonable limit on the number of tetrahedra we try to eliminate at any one time. Once a poor-quality tetrahedron has been identified, we attempt to remove an edge of the tetrahedron from the mesh. Edges having bad dihedral angles are tried first. If edge swapping fails, we attempt to swap each face of the tetrahedron. If a pass through the mesh using this approach results in reconfiguration of one or more local sub-meshes, another pass is made, checking only tetrahedra that are not known to be well-shaped. Typically two to three passes are required. This procedure is quite effective in removing poor quality tetrahedra whose neighbors are of good quality. In cases where poor-quality tetrahedra are adjacent to each other, valid reconfigurations are often impossible to find by edge swapping.

3. An Optimization Approach to Mesh Smoothing

Perhaps the most commonly used mesh-smoothing technique is a local algorithm called Laplacian smoothing.[13, 15] This technique adjusts the location of each mesh point to the arithmetic mean of its incident vertices so that

$$x_{free} = \frac{\sum_{i \in V} x_i}{|V|}, \quad y_{free} = \frac{\sum_{i \in V} y_i}{|V|}, \quad z_{free} = \frac{\sum_{i \in V} z_i}{|V|}, \tag{1}$$

where V is the set of incident vertices and x, y, and z are the spatial coordinates of each vertex. This method is computationally inexpensive, but it does not provide any mechanisms

that guarantee improvement in element quality. In fact, it is possible to produce an invalid mesh containing elements that are inverted or have negative volume.

Freitag et al. proposed a low-cost, optimization-based alternative to Laplacian smoothing that guarantees valid elements in the final mesh.[16] Several results were given that demonstrated the effectiveness of this method compared with Laplacian smoothing for two-dimensional, triangular meshes. Like Laplacian smoothing, the optimization algorithm is local and uses the union of elements that are adjacent to the free vertex as the solution space. Thus, it can be used as the core of an efficient parallel algorithm. They presented a P-RAM computational model for parallel implementation based on coloring heuristics. This model resulted in correct parallel execution and a low run-time bound, and experimental data showed very good scalable performance on 1 to 64 processors on the IBM SP supercomputer.

In this section we extend this algorithm to three-dimensional tetrahedral meshes and note that the algorithm is useful for hexahedral meshes as well. A parallel algorithm analogous to the two-dimensional algorithm has been developed for the three-dimensional case, but we do not focus on that aspect of our work here. In this article we describe the formulation of the optimization method and give some useful measures of mesh quality for tetrahedral meshes. The same formulation applies for hexahedral meshes, but different mesh quality measures must be used. As in the two-dimensional case, we formulate the problem using analytic expressions for local mesh quality written in terms of free vertex position. Typical measures for three-dimensional tetrahedral meshes that have an analytic expression include measures of the dihedral angles, measures of the solid angles, and element aspect ratios. Any combination of these can be used within the framework of the optimization method presented here. Our algorithm seeks to maximize the minimum value of the mesh quality measure; minimizing the maximum value of the quality measure can be done by negating the function value. We require that function and gradient evaluations dependent on free vertex position be provided by the user.

The optimization algorithm for each local subproblem is similar to a minimax technique used to solve circuit design problems.[17] We briefly review the formulation of the problem here and refer interested readers to a more complete description in our previous paper.[16] To facilitate the discussion of problem formulation, we first introduce some useful notation:

- $\mathbf{x}$: the position of the free vertex
- $f_i(\mathbf{x})$: an analytic function for a mesh quality measure that in general is nonlinear. For example, if we consider maximizing the minimum dihedral angle of the mesh, each tetrahedron will have six function values for each location of the point $\mathbf{x}$. Let the entire set of function values, f_i, be $\mathcal{S}$.
- $\mathbf{g}_i(\mathbf{x})$: the analytic gradient of the mesh quality measure corresponding to $f_i(\mathbf{x})$i. The gradient evaluation code was created using the ADIC automatic differentiation software package developed at Argonne National Laboratory.[18]
- $\mathcal{A}$: the set of functions that achieve a minimum at point $\mathbf{x}$ (the active set)
- $\mathbf{x}^*$, $\mathcal{A}^*$: the optimal solution and the active set at $\mathbf{x}^*$

As the location of $\mathbf{x}$ changes in the solution space, the minimum function value in the corresponding submesh is given by the composite function

$$\phi(\mathbf{x}) \;=\; \min_{i \in S} \; f_i(\mathbf{x}). \tag{2}$$

We illustrate the character of this function by showing a one-dimensional slice through a typical function ϕ in Figure 4. Note that each $f_i(\mathbf{x})$ is a smooth, continuously differentiable function and that multiple function values can obtain the minimum value. Hence, the composite function $\phi(\mathbf{x})$ has discontinuous partial derivatives where two or more of the functions f_i obtain the minimum value, that is, where the active set $\mathcal{A}$ changes.

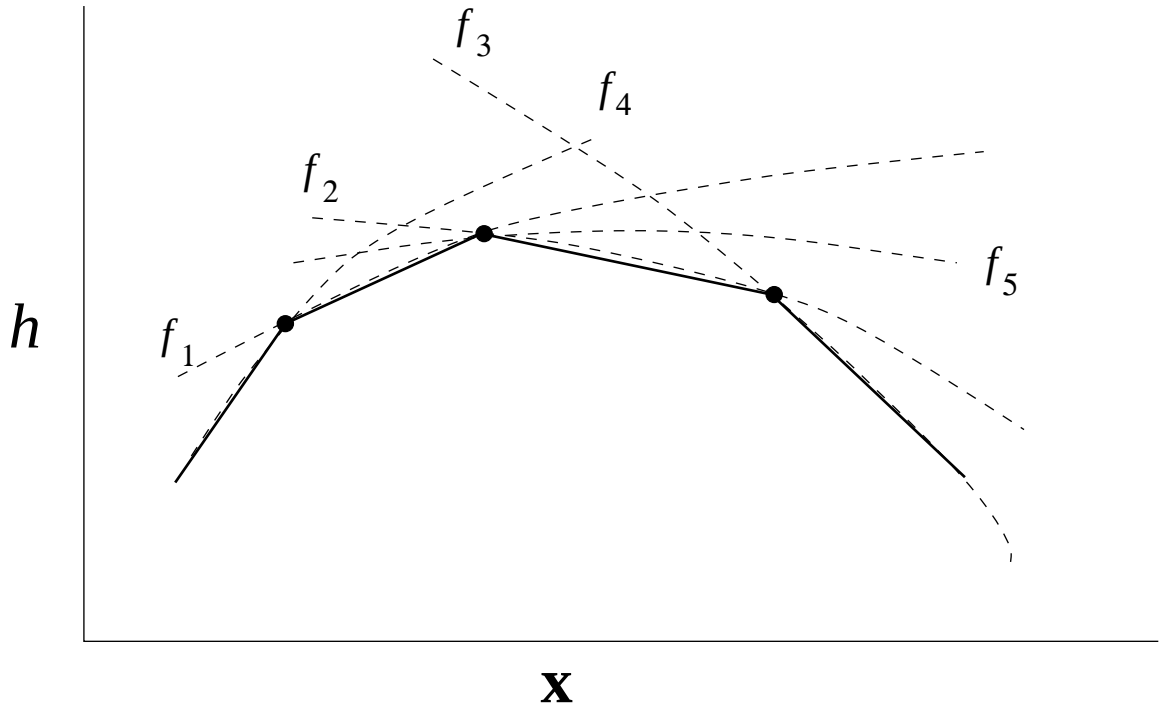

Figure 4: A one-dimensional slice through the nonsmooth function $\phi(x)$

We solve the nonsmooth optimization problem (2) using an analogue of the steepest descent method for smooth functions. The search direction $\bar{\mathbf{g}}$ at each step is computed by solving the quadratic programming problem

$$\min \;\; \bar{\mathbf{g}}^T \bar{\mathbf{g}} \quad \text{where} \;\; \bar{\mathbf{g}} = \sum_{i \in \mathcal{A}} \beta_i \mathbf{g}_i(\mathbf{x})$$

$$\text{subject to} \;\; \sum_{i \in \mathcal{A}} \beta_i \;=\; 1, \quad \beta_i \geq 0$$

for the β_i. This gives the direction of steepest descent from all possible convex linear combinations of the gradients in the active set at $\mathbf{x}$. The line search subproblem along $\bar{\mathbf{g}}$ is solved by finding the linear approximation of each function $f_i(\mathbf{x})$ given by the first-order Taylor series approximation,

$$f_i(\mathbf{x} + \delta) \;=\; f_i(\mathbf{x}) \;+\; \delta^T \mathbf{g}_i(\mathbf{x}).$$

Using this information, we can predict the points at which the active set $\mathcal{A}$ will change by computing the intersection of each linear approximation with the projection of the current active function in the search direction. The distance to the nearest intersection point from the current location gives the initial step length, α. The initial step is accepted if the actual improvement exceeds 90 percent of the estimated improvement or the subsequent step results in a smaller function improvement. Otherwise α is halved recursively until a step is accepted or α falls below some minimum step length tolerance.

The optimization process is terminated if one of the following conditions apply: (1) the step size falls below the minimum step length with no improvement obtained; (2) the maximum number of iterations is exceeded; (3) the achieved improvement of any given step is less than some user-defined tolerance; or (4) the Kuhn-Tucker conditions of nonlinear programming

$$\sum_{i \in \mathcal{A}^*} \lambda_i \mathbf{g}_i(\mathbf{x}^*) = 0$$

$$\sum_{i \in \mathcal{A}^*} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i \in \mathcal{A}^*$$

are satisfied indicating that we have found a local maximum $\mathbf{x}^*$.[17]

Amenta et al.[19] have shown that this technique is equivalent to a generalized linear programming (GLP) technique and can therefore be solved in linear time. In addition, the convex level set criterion for solution uniqueness of the GLP technique can be applied to these algorithms. Amenta et al. determine the convexity of the level sets for a number of standard mesh quality measures in both two and three dimensions.

We note that other optimization-based smoothing techniques have been developed by researchers in the mesh generation and computational geometry communities. These methods differ primarily in the optimization procedure used or in the quantity that is optimized. For example, Bank[20] and Shephard and Georges[21] propose similar techniques for triangles and tetrahedra, respectively. In these methods, an element shape quality measure, $q(t)$, is defined based on a ratio of element area (volume) to side lengths (face areas). In each case, $q(t)$ is equal to one for equilateral elements and is small for distorted elements. The free vertex is moved along the line that connects its current position to the position that makes $q(t)$ equal to one for the worst element in the local submesh. The line search in this direction is terminated when two elements have equal shape measure. We note that this does not necessarily guarantee that the optimal local solution has been found.

All the techniques mentioned above, including the one described in this article, optimize the mesh according to element geometry. In contrast, Bank and Smith [22] propose two smoothing techniques to minimize the error in finite element solutions computed with triangular elements with linear basis functions. Both methods use a damped Newton's method to minimize the interpolation error or the a posteriori error estimates for an elliptic partial differential equation. The quantity minimized in both of these cases requires the computation of approximate second derivatives for the finite element solution as well as the shape function $q(t)$ for triangular elements mentioned above. Although the authors present convergence histories for these techniques, no timing results are given that quantify the cost and effectiveness of these techniques compared with the geometric techniques mentioned earlier.

4. Computational Experiments

We now present computational results for five test cases: two randomly generated meshes and three meshes generated for application problems. For each test mesh, a standard starting mesh was generated, and all comparison cases began with that same mesh. The random meshes were each generated in a cube with points incrementally inserted at random in the interior. Each point was connected to the vertices of the tetrahedron containing it, with points near an existing face or edge in the tetrahedralization projected onto that face or edge. No swapping or smoothing was done as these initial meshes were generated, and mesh quality was correspondingly poor at the outset. The first case, `rand1`, has 1086 points approximately equally distributed through the domain and 5104 tetrahedra. The second random mesh, `rand2` has 5086 points clustered at the center of the cube by selecting random numbers from a Gaussian deviate and 25704 tetrahedra.

The third and fourth test meshes were generated in the interiors of a tire incinerator and a tangentially-fired (t-fired) industrial boiler, respectively. Interior points were inserted at the circumcenter of cells that were larger than appropriate, based on an automatically computed local length scale, or that had a large dihedral angle and had a volume larger than a user-defined tolerance. After the insertion of each point, nearby faces were swapped by using the in-sphere criterion to improve local mesh connectivity. After all points were inserted, all faces were swapped by using the minmax dihedral angle criterion. The tire incinerator mesh initially has 2570 vertices and 11098 tetrahedra, and the t-fired mesh has 7265 vertices and 37785 tetrahedra. Because these meshes were generated more sensibly than the random meshes, initial quality is much better than in the random cases.

The final test case is a mesh generated around the ONERA M6 wing attached to a flat wall. This is a standard geometry for testing three-dimensional compressible flow solution algorithms. This particular mesh is coarse, having 6,000 vertices and 31,978 tetrahedra. Hence, the initial mesh quality is poor, especially near the junction of the wing and the wall.

For each test case, we present results for mesh quality using dihedral angles as a quality measure.[‡] The maximum and minimum dihedral angles over the entire mesh are given as an indication of how poor the worst elements are. To give quantitative information about the number of poor tetrahedra, we also give the percentage of dihedral angles falling below 6, 12, and 18 degrees and above 162, 168, and 174 degrees. To improve the readability of this section, several tables of results that support our conclusions and recommendations but do not add new information are included in a separate appendix.

Recommendations for mesh improvement are made on the premise that computational PDE codes behave poorly when very small or very large dihedral angles are present, and that, therefore, the goal of mesh improvement is to remove such angles.

4.1 Improvement Using Local Reconnection Techniques Only

The first experiment compares the effectiveness of several local reconnection strategies in improving mesh quality for the random meshes. We use three geometric quality measures

[‡] We prefer not to use measures such as aspect ratio that are misleading for anisotropic meshes.

to determine whether to locally reconnect a tetrahedral mesh: maxmin sine of dihedral angle, minmax dihedral angle, and in-sphere. The maxmin sine of dihedral angle criterion chooses the configuration that maximizes the minimum sine of the dihedral angles of the tetrahedra in the submesh. The minmax dihedral angle criterion minimizes the maximum dihedral angle of the tetrahedra. The in-sphere criterion, appropriate only for face swapping, selects the configuration in which no tetrahedron in the five-point local submesh contains the other point in its circumsphere. This leads to a locally Delaunay tetrahedralization in the sense that no face in the mesh has incident cells that violate the in-sphere criterion and are reconfigurable. For all criteria, however, the optimum reached by face and edge swapping is almost certainly local rather than global. Recent work by Joe[9] describes a more advanced technique for improving mesh quality by local transformations. This approach notwithstanding, it is not known whether the global optimum can be reached by *any* series of local transformations.

Table 2 shows mesh quality results for the initial `rand1` mesh and the same mesh following local reconnection. Results are given for eight different local reconnection strategies: (1) face swapping to minimize the maximum dihedral angle; (2) face swapping to maximize the minimum sine of the dihedral angles; (3-4) each of the above preceded by a pass of in-sphere face swapping; and (5-8) each of the four cases given above with edge-swapping allowed when the minmax angle and maxmin sine criteria are used. We note that strategies for which the final step is a pass of in-sphere swapping leave very poor extremal angles and large numbers of bad dihedral angles, and so are not considered here.

Recommendation 1 *Never use the in-sphere criterion during the final pass of face swapping. The in-sphere criterion performs poorly in practice with respect to extremal angles.*

In all cases, edge swapping proves to be beneficial; it improves the worst dihedral angle in the mesh, the number of bad angles in the mesh, or both. Using in-sphere face swapping as a first step dramatically improves the distribution of dihedral angles at the expense of degrading the extremal angle. The results for mesh `rand2` are similar and included in Appendix A.

Recommendation 2 *Edge swapping is a beneficial supplement to face swapping and should be used.*

4.2 Improvement Using Mesh Smoothing Techniques Only

Our baseline smoothing technique for comparison is Laplacian smoothing, which moves each vertex to the average of the location of its neighbors, provided that this point does not result in an invalid mesh. Also, we have implemented a "smart" variant of Laplacian smoothing, which requires that the local mesh quality be improved before accepting a change in vertex location. Any local quality criterion suitable for use with the optimization-based smoothing can be used in this context.

For optimization-based smoothing, we present results for five different objective functions:

Table 2: Mesh quality improvement for `rand1` with swapping

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	% Dihedral Angles < 12°	% Dihedral Angles < 18°	% Dihedral Angles > 162°	% Dihedral Angles > 168°	% Dihedral Angles > 174°
Initial	0.32°	178.97°	1.41	4.90	9.86	2.40	1.08	0.24
Without Edge Swapping								
Minmax angle	0.54°	178.97°	0.76	3.20	7.40	1.21	0.46	0.11
Maxmin sine	0.54°	178.97°	0.68	2.94	6.94	1.27	0.49	0.12
In-sphere, then minmax angle	$3.6 \cdot 10^{-6}$°	180.00°	0.45	1.48	3.28	0.58	0.30	0.11
In-sphere, then maxmin sine	$3.6 \cdot 10^{-6}$°	180.00°	0.46	1.48	3.26	0.60	0.32	0.12
With Edge Swapping								
Minmax angle	0.54°	178.97°	0.24	1.42	4.19	0.36	0.15	0.034
Maxmin sine	0.54°	178.97°	0.15	0.96	3.16	0.43	0.15	0.033
In-sphere, then minmax angle	0.26°	177.88°	0.15	0.78	2.25	0.24	0.10	0.025
In-sphere, then maxmin sine	0.38°	179.08°	0.16	0.68	1.98	0.28	0.14	0.051

1. Maximize the minimum dihedral angle (maxmin angle)
2. Minimize the maximum dihedral angle (minmax angle)
3. Maximize the minimum cosine of the dihedral angles (maxmin cosine)
4. Minimize the maximum cosine of the dihedral angles (minmax cosine)
5. Maximize the minimum sine of the dihedral angles (maxmin sine)

We expect nearly identical results, though not necessarily identical convergence behavior, from two pairs of these measures:

$$\begin{aligned} \text{maxmin angle} &\approx \text{minmax cosine} \\ \text{minmax angle} &\approx \text{maxmin cosine.} \end{aligned}$$

Table 3 shows the results of smoothing `rand1` using Laplacian smoothing, smart Laplacian smoothing for two of the five criteria given, and optimization-based smoothing for each of the five criteria. Results for the other smart Laplacian approaches are identical to one of the two smart Laplacian results presented.

The improvement in mesh quality is not as pronounced for smoothing as for swapping because the connectivity is too irregular to allow a truly high-quality mesh. Nevertheless, all the optimization-based smoothing criteria improve mesh quality significantly, especially in the sense of improving the extremal angles. The Laplacian smoother does a poor job of eliminating very bad angles. The smart Laplacian smoothers perform better in this respect, but still are significantly worse than the optimization-based smoothers. Optimization criteria that seek only to force all dihedral angles away from 180 degrees (minmax angle and maxmin

cosine) are unsuccessful in eliminating small dihedral angles, whereas criteria that force dihedral angles away from 0 degrees (maxmin angle and minmax cosine) also succeed in eliminating large dihedral angles. This difference can be important in practice because both large and small angles can affect the quality of the final application solution. Note also that the pairs of smoothing criteria expected to perform comparably behave similarly, with one exception. The minmax cosine criterion does not improve the extremal angles as much as its analog, the maxmin angle criterion. The flatness of the cosine near 0 and 180 degrees prevents rapid quality improvement when moving points in tetrahedra with extremely poor angles, and the optimization code concludes that improvement is too slow to be fruitful. Finally, the maxmin sine criterion, which forces dihedral angles away from both 0 and 180 degrees, is very successful in removing dihedral angles at both extremes. The results for mesh `rand2` are similar and are included in Appendix A.

Table 3: Mesh quality improvement for `rand1` with smoothing

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
Lap: No constraint	0.0026°	179.996°	2.45	6.90	12.21	3.46	1.84	0.63
Lap: Maxmin angle	1.18°	177.43°	0.71	3.23	7.28	1.55	0.58	0.14
Lap: Minmax angle	0.67°	177.43°	0.73	3.44	7.55	1.44	0.54	0.091
Opt: Maxmin angle	4.79°	175.59°	0.11	1.23	6.21	0.71	0.21	0.0065
Opt: Minmax angle	$5.37 \cdot 10^{-4}$°	172.75°	2.71	5.54	9.09	0.16	0.039	0
Opt: Maxmin cosine	0.018°	172.57°	2.31	4.78	8.71	0.17	0.033	0
Opt: Minmax cosine	1.70°	176.21°	0.11	1.26	6.31	0.71	0.19	0.016
Opt: Maxmin sine	4.20°	175.73°	0.085	1.05	6.09	0.60	0.11	0.0033

4.3 Improvement Using the Combined Swapping/Smoothing Approach

Next we show that the gains in mesh quality from local reconnection and smoothing can be made cumulatively. To determine which reconnection strategy performs best in conjunction with smoothing, we compare the results of the four reconnection strategies with edge swapping enabled followed by six passes of optimization-based smoothing using the the maxmin sine criterion for `rand1`. Table 4 shows that smoothing eliminates the worst tetrahedra in all cases. Quantitatively, however, there are important differences. The maxmin sine reconnection criterion consistently outperforms the minmax angle criterion, especially in improving the worst angle in the mesh. Likewise, the use of in-sphere swapping as a first reconnection pass gives a dramatic improvement regardless of the criterion used for the second pass. These results are consistent across a number of test cases.

Recommendation 3 *We recommend that meshes whose connectivity has not been improved during generation be reconnected using in-sphere face swapping, followed by face and edge swapping using the maxmin sine of dihedral angle criterion. For meshes that have initially reasonable connectivity, only the second pass need be performed.*

Table 4: Comparison of the effectiveness of smoothing for four different swapping options (mesh `rand1`, edge swapping enabled, maxmin sine smoothing)

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
Minmax angle	6.74°	171.14°	0	0.24	1.34	0.11	0.0093	0
Maxmin sine	9.74°	168.09°	0	0.030	0.50	0.049	0.0030	0
In-sphere, then minmax angle	9.90°	166.01°	0	0.014	0.15	0.017	0	0
In-sphere, then maxmin sine	12.59°	167.25°	0	0	0.20	0.017	0	0

Table 5 shows the results for `rand1` using the local reconnection procedure given in Recommendation 3 followed by each of the eight smoothing options discussed in the preceding section. The distribution of dihedral angles for each random mesh improves significantly regardless of the choice of smoothing criterion. As was the case with smoothing used alone, all Laplacian smoothers fail to eliminate poorly shaped elements from the mesh. Again we see that the criteria for optimization-based smoothing that seek only to remove large angles from the mesh do not succeed in eliminating small angles. The three other criteria — maxmin angle, minmax cosine, and maxmin sine — give comparable results for these test cases.

Table 5: Mesh quality improvement for `rand1` with both swapping and smoothing

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
Lap: No constraint	0.49°	178.97°	0.099	0.37	0.97	0.19	0.082	0.034
Lap: Maxmin angle	2.19°	175.65°	0.034	0.20	0.82	0.082	0.034	0.0057
Lap: Minmax angle	0.57°	178.88°	0.077	0.30	1.08	0.091	0.051	0.023
Opt: Maxmin angle	14.05°	165.05°	0	0	0.12	0.011	0	0
Opt: Minmax angle	0.024°	159.96°	1.29	2.52	4.48	0	0	0
Opt: Maxmin cosine	0.0066°	176.26°	1.18	2.40	4.31	0.026	0.011	0.0085
Opt: Minmax cosine	15.01°	166.71°	0	0	0.16	0.0085	0	0
Opt: Maxmin sine	12.59°	167.25°	0	0	0.20	0.017	0	0

An important question in any local smoothing algorithm is the number of smoothing passes required to improve the mesh to the point where further improvement is negligible. Table 6 shows the effect of various numbers of smoothing passes with the maxmin sine criterion on `rand1`. Similar results for `rand2` with the maxmin angle criterion can be found in Appendix A. In both cases swapping was used before smoothing. In each case, mesh quality improves only negligibly after the fourth or fifth smoothing pass.

We conclude this subsection with a comparison of the computational efficiency of the various mesh improvement techniques. Table 7 compares timings for mesh improvement using swapping, smoothing, and a combination of the two for `rand2` on a 110 MHz SPARC

Table 6: Effect of the number of optimization passes on mesh improvement (`rand1` with maxmin sine smoothing)

Passes	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	12°	18°	% Dihedral Angles > 162°	168°	174°
0	0.38°	179.08°	0.17	0.68	1.98	0.28	0.14	0.051
1	0.40°	179.21°	0.063	0.19	0.89	0.088	0.046	0.026
2	8.10°	169.55°	0	0.065	0.53	0.043	0.0028	0
3	10.51°	167.20°	0	0.028	0.34	0.026	0	0
4	12.23°	166.88°	0	0	0.25	0.020	0	0
5	12.48°	167.02°	0	0	0.22	0.014	0	0
6	12.59°	167.25°	0	0	0.20	0.017	0	0

5. The times for the swapping-only cases indicate that edge swapping, while very beneficial, is a relatively costly operation. More work is needed to improve the efficiency of this scheme; improvement will mostly likely take the form of reducing the number of cases that the edge swapping algorithm examines in detail. The difference in timings for smoothing with and without swapping is due to several differences in the swapped verses nonswapped mesh. Approximately 30 percent of the Laplacian smoothing steps on the nonswapped mesh result in an invalid mesh as compared with approximately 3 percent on the swapped meshes. Because an invalid step requires less time to execute than the full Laplacian step, the nonswapped mesh can be smoothed more quickly. A similar situation holds for the optimization-based smoothing. Optimization-based smoothing requires an average of approximately 3.9 iterations on the swapped mesh and about 3.7 iterations on the nonswapped mesh. A higher percentage of equilibrium points is found on the swapped mesh than on the nonswapped mesh, and the quality is correspondingly better. For these examples, optimization-based smoothing takes 10 times longer than smart Laplacian smoothing and 25 to 30 times longer than simple Laplacian smoothing.

Table 7: Sample times for mesh improvement (mesh `rand2`)

Case	Swap Time (sec)	Smoothing Time (sec)	Smoothing Calls	Time per Call (msec)	Min. Dihed.	Max. Dihed.
Face Swap only	34.6	—	—	—	$3.52 \cdot 10^{-6}$°	179.83°
Face and Edge Swap	156	—	—	—	0.322°	178.72°
Lap: No constraint	—	8.13	14832	.565	0.014°	179.98°
Lap: Maxmin sine	—	21.9	14832	1.52	0.63°	178.82°
Opt: Maxmin sine	—	211	14832	14.7	1.91°	177.69°
Swap + Lap: No constraint	156	10.1	14832	.704	.0223°	179.96°
Swap + Lap: Maxmin sine	156	30.0	14832	2.08	.322°	178.72°
Swap + Opt: Maxmin sine	156	305	14832	21.2	6.59°	171.32°

4.4 Improvement Using Combined Laplacian/Optimization-Based Smoothing

We would ideally like a smoothing technique that is as effective as optimization-based smoothing techniques at the cost of Laplacian smoothing. In this section we investigate a combined Laplacian/optimization-based smoothing approach. Smart Laplacian smoothing is used as the initial step, and the resulting submesh is evaluated for improvement. If the mesh quality is improved, the step is accepted. An additional test is performed to compare the current quality of the submesh with a user-defined threshold value. If the quality is less than the threshold value, optimization-based smoothing is performed to further improve the mesh, otherwise the smoothing algorithm proceeds to the next submesh. A drawback to using a fixed threshold for optimization-based smoothing is that mesh improvement ceases when the threshold is reached. Therefore, we also consider a floating threshold value that can be reset after each smoothing pass to the worst remaining angle plus some constant. This ensures that optimization-based smoothing always tries to improve a reasonably small number of poor-quality tetrahedra.

In Table 8, we give the results for this combination approach on `rand1`. We first swap each of the meshes using the recommended procedure. We then smooth the mesh using six passes of smart Laplacian, optimization-based smoothing only, and the combination approach for five different threshold values: fixed values of 5 degrees, 10 degrees, 15 degrees, and 30 degrees; and a floating value of 10 degrees for the first pass followed by the worst remaining angle plus five degrees on subsequent passes. We use the maxmin sine criterion for smoothing in all cases. In addition to mesh quality information, we give the average time required to smooth each local submesh. It is clear that the combination approach with a threshold of 10 to 15 degrees results in very high quality meshes at a fraction of the optimization-only smoothing cost. In fact meshes with quality comparable to or better than that obtained with optimization-based smoothing alone can be computed for 1.5 to 2 times the cost of smart Laplacian smoothing. The cost and final mesh quality for the floating threshold falls between the combined approach with thresholds of 10 and 15 degrees. The benefit of using the floating threshold is that no advance knowledge of the attainable extremal angle is required to consistently approach that limit. Similar trends are evident for `rand2` and those results are given in Appendix A.

Recommendation 4 *The local reconnection schemes should be followed by three to four passes of a combined Laplacian/optimization-based smoothing technique with a floating threshold. Quality criteria that tend to eliminate small angles in the mesh are more effective than criteria that tend to eliminate large angles.*

4.5 Improvement of Application Meshes

We now confirm that the mesh improvement recommendations given above for the random meshes are appropriate for the application meshes. First, however, we address an anomaly that appears in an examination of the effect of number of smoothing passes for the tire incinerator mesh. A smoothing history is presented in Table 9; the mesh was reconnected by

Table 8: Mesh quality improvement for combined Laplacian/optimization-based smoothing (mesh `rand1`)

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >			Time (ms)
			6°	12°	18°	162°	168°	174°	
Laplacian	.490°	178.97°	0.099	0.37	0.98	0.19	0.082	0.034	2.01
Optimization	12.59°	167.25°	0	0	0.20	0.017	0	0	18.2
Combined (30)	12.59°	166.98°	0	0	0.088	0.0057	0	0	17.5
Combined (15)	14.06°	164.18°	0	0	0.53	0.020	0	0	3.37
Combined (10)	10.13°	169.20°	0	0.12	0.74	0.040	0.0057	0	2.29
Combined (5)	5.13°	170.72°	0.0057	0.22	0.87	0.074	0.023	0	2.04
Floating	13.42°	164.74°	0	0	0.33	0.0028	0	0	2.95

Table 9: Effect of the number of optimization passes on mesh improvement (`tire` with minmax angle smoothing)

Passes	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
0	3.36°	172.38°	0.065	0.33	0.86	0.035	0.012	0
1	5.20°	164.55°	0.0015	0.026	0.22	0.0045	0	0
2	5.20°	164.25°	0.0015	0.0030	0.091	0.0015	0	0
3	5.20°	161.43°	0.0015	0.0030	0.045	0	0	0
4	5.20°	161.43°	0.0015	0.0030	0.036	0	0	0

using the maxmin sine criterion with edge swapping before smoothing. The smallest dihedral angle in the mesh could not be improved beyond 5.20°, but the optimization-based smoothing code reported a minimum dihedral angle of 14.3° among all submeshes which it attempted to smooth. On careful investigation, we found the cause of this discrepancy: a tetrahedron with all four of its vertices on the boundary of the mesh. As our smoothing algorithm only operates on vertices internal to the mesh, these vertices could not be relocated. When the BATR procedure described in Section 2.2 was invoked after two smoothing passes, the worst tetrahedron in the mesh was removed and the mesh quality improved dramatically, to a minimum dihedral angle of 13.67°; the results are shown in Table 10. Note that the worst angle remaining in this mesh still lies in a tetrahedron with all vertices on the boundary. Although using BATR does not always have such a significant effect, it is categorically recommended because the cost is low and the potential benefit high.

Recommendation 5 *We recommend performing two passes of smoothing followed by a procedure such as BATR to remove the worst tetrahedra from the mesh and finishing with two more passes of smoothing.*

To demonstrate that our recommendations are appropriate for the tire incinerator mesh, we present a set of cases in which one recommendation has been ignored. The baseline mesh improvement scheme for these cases is a pass of face and edge swapping using the maxmin sine

Table 10: Effect of the number of optimization passes and edge swapping on mesh improvement (`tire` with minmax angle smoothing)

Passes	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	12°	18°	% Dihedral Angles > 162°	168°	174°
0	3.36°	172.38°	0.065	0.33	0.86	0.035	0.012	0
1	5.20°	164.55°	0.0015	0.026	0.22	0.0045	0	0
2	5.20°	164.25°	0.0015	0.0030	0.091	0.0015	0	0
BATR	9.44°	164.25°	0	0.0015	0.088	0.0015	0	0
3	13.67°	159.82°	0	0	0.046	0	0	0
4	13.67°	159.82°	0	0	0.038	0	0	0

criterion; two passes of combined Laplacian/optimization-based smoothing with a floating threshold and the maxmin sine criterion; an application of the BATR procedure; and two more passes of smoothing with the same parameters. For each example, Table 11 tells which recommendation was ignored, summarizes the variation from recommended parameters, and gives the resulting mesh quality and total execution time. No angles greater than 168° appear for any of the examples, so two of the usual columns are absent from the table. Only one case that ran faster than the baseline case produces a mesh of comparable quality (fixed threshold, 15°), and the time savings is only 3%. The only case with a better final mesh than the baseline case is the case in which passes of smoothing and BATR alternate; this case improves the maximum dihedral angle in the mesh by 1.3° at a cost of 25% more CPU time. A similar table for the `tfire` mesh is presented in the appendix, with comparable results.

Table 12 shows the improvement in mesh quality achieved for each of the three application meshes using our recommended procedure. For all three cases, mesh quality is improved significantly. The final mesh quality differs dramatically among the three cases, because of the initial topology and point distribution of the meshes. For example, the M6 wing mesh began with a very large number of poor dihedral angles in adjacent tetrahedra. While smoothing improved many tetrahedra, some could not be improved without making a neighboring cell worse, and so no improvement was made.

This clustering of bad tetrahedra is a common occurrence in our final meshes, with the worst cells often sharing vertices, edges, or even faces. Figures 6 and 5 show surface wireframes for the tire incinerator and t-fired boiler, along with the worst tetrahedra—those with dihedral angles less than 18° or greater than 162°. For the t-fired boiler, these tetrahedra fall primarily into a single clump along a corner of the geometry. Figure 7 shows a closeup of a region around the leading edge of the wing at the wall where there is a concentration of poor-quality tetrahedra. Further work is needed to improve quality in difficult cases such as these in which boundary constraints or clustering prevents the improvement of poorly shaped elements.

Table 11: Verification of mesh improvement recommendations for `tire` mesh

Rec.	Variation	Max. Dihed.	Min. Dihed.	% Dihedrals < 6°	% Dihedrals < 12°	% Dihedrals < 18°	% > 162°	Time (sec)
—	Baseline	13.67°	159.82°	0	0	0.038	0	43.3
1	In-sphere swap	9.03°	162.22°	0	0.021	1.20	0.0028	45.6
2	BATR, no edge swap	5.20°	161.43°	0.0015	0.0030	0.16	0	27.3
3	Angle swap	5.20°	161.43°	0.0015	0.0030	0.13	0	38.2
	In-sphere + Angle	11.80°	159.11°	0	0.0015	0.15	0	50.8
	In-sphere + Sine	5.20°	161.43°	0.0015	0.0030	0.10	0	55.9
4	Optimize	13.67°	159.74°	0	0	0.049	0	127.7
	Fixed thresh. 10°	10.19°	164.14°	0	0.011	0.12	0.0015	40.5
	Fixed thresh. 15°	13.67°	159.92°	0	0	0.079	0	42.1
	Lap: Maxmin angle	8.64°	164.38°	0	0.012	0.13	0.0030	45.9
	Maxmin angle	13.67°	161.71°	0	0	0.045	0	48.0
	Maxmin cosine	0.77°	156.14°	0.048	0.19	0.72	0	67.9
5	No BATR	5.20°	161.43°	0.0015	0.0030	0.036	0	40.0
	Multi-BATR	13.67°	158.52°	0	0	0.027	0	54.8

Table 12: Mesh improvement for three application meshes

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	% Dihedral Angles < 12°	% Dihedral Angles < 18°	% Dihedral Angles > 162°	% Dihedral Angles > 168°	% Dihedral Angles > 174°
Tire incinerator before	0.66°	178.88°	0.11	0.54	1.27	0.074	0.035	0.0075
Tire incinerator after	13.67°	159.82°	0	0	0.038	0	0	0
T-fire boiler before	0.048°	179.86°	0.16	0.50	0.99	0.24	0.12	0.037
T-fire boiler after	5.61°	174.15°	0.0013	0.029	0.11	0.019	0.0071	0.00045
ONERA M6 wing before	0.0066°	179.984°	0.78	1.63	2.85	0.57	0.41	0.23
ONERA M6 wing after	0.098°	179.76°	0.16	0.66	1.46	0.17	0.076	0.018

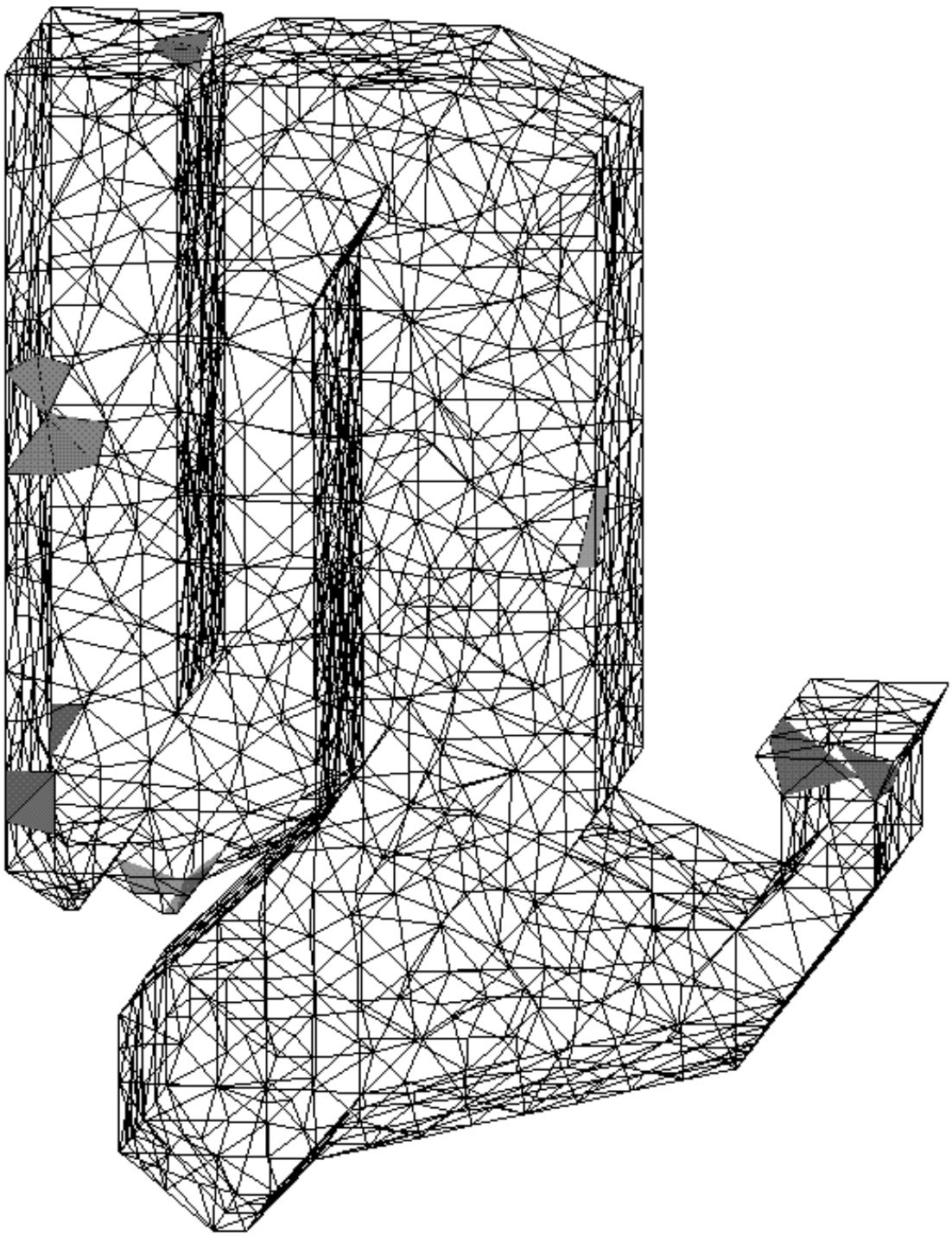

Figure 5: Surface wireframe of tire incinerator mesh with badly shaped tets

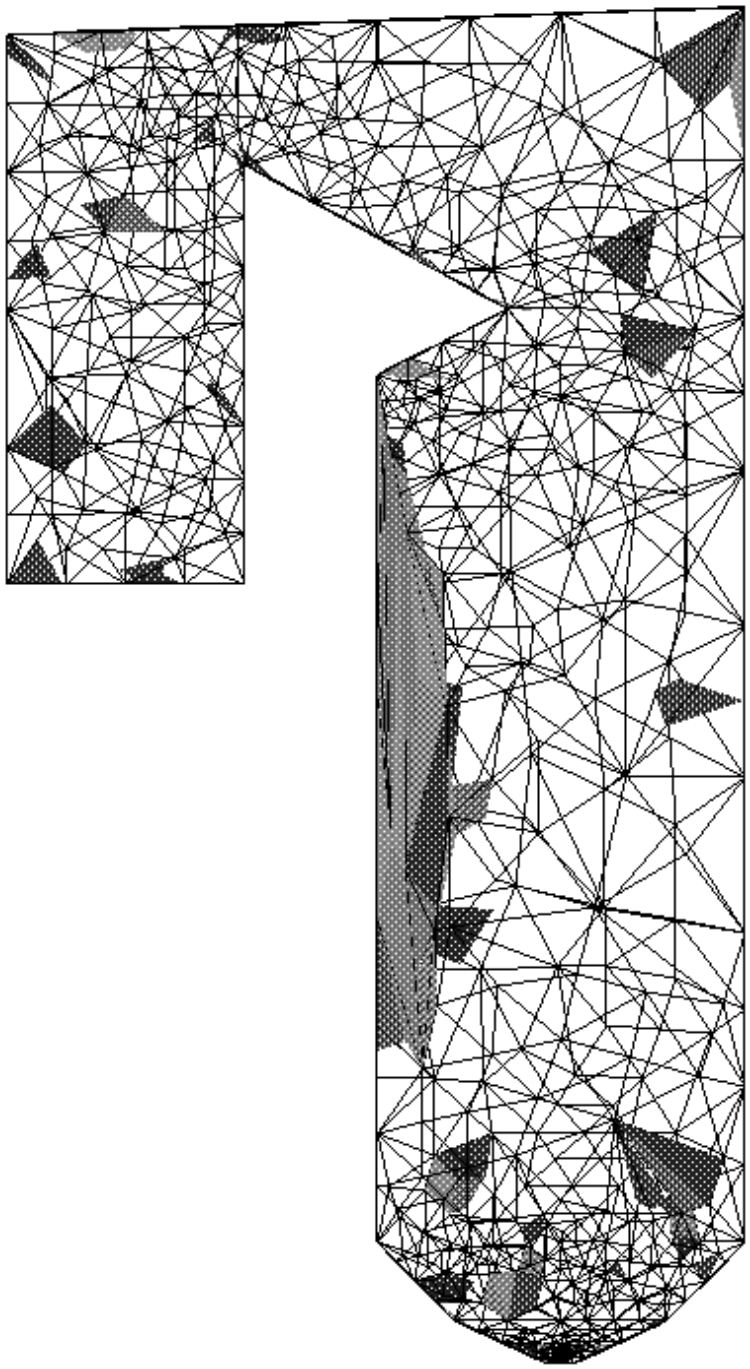

Figure 6: Surface wireframe of tangentially-fired boiler mesh with badly shaped tets

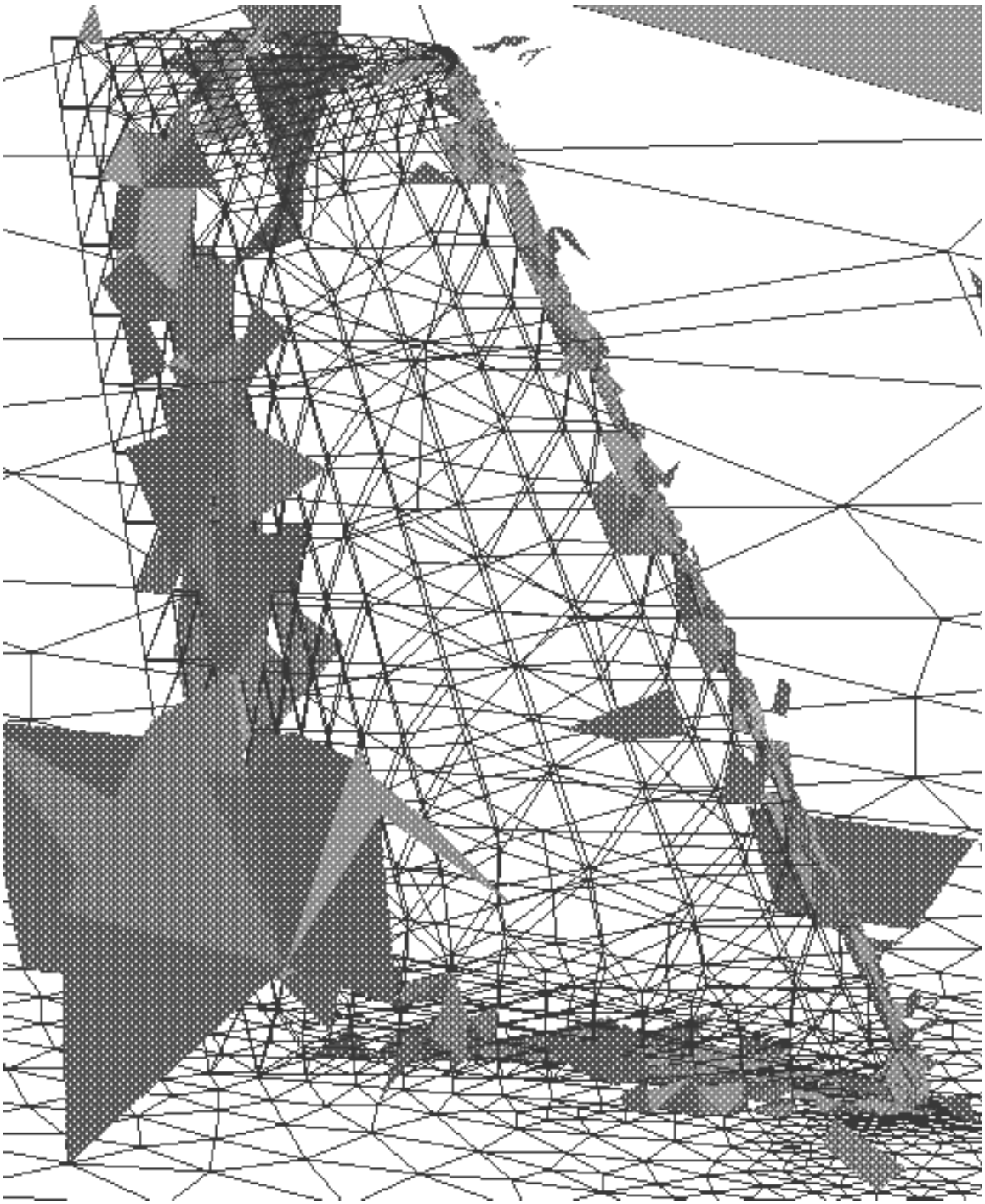

Figure 7: Closeup of leading edge of ONERA M6 wing surface mesh with badly shaped tets

5. Conclusions

In this article we compared combinations of mesh swapping and mesh smoothing techniques used to improve the quality of tetrahedral meshes. Using two random meshes as test cases, we showed that each mechanism fails to give high-quality meshes when used individually; that is, not all of the very large and very small dihedral angles were removed from the meshes. Local reconnection was performed using the minmax dihedral angle and maxmin sine of dihedral angle criteria, with and without edge swapping and with and without a preparatory pass of face swapping using the in-sphere criterion. Both Laplacian and optimization-based smoothing techniques fail to improve the general distribution of angles because they cannot change local mesh connectivity. However, we showed that the cumulative improvement obtained when combining in-sphere and maxmin sine reconfiguration (with edge swapping) followed by optimization-based smoothing results in very high quality meshes. In addition, experiments showed that a combination of smart Laplacian smoothing followed by optimization-based smoothing led to meshes equal in quality to those generated exclusively by optimization-based smoothing at a much lower computational cost. The use of a mesh quality dependent threshold for invocation of optimization-based smoothing was found to be inexpensive as well as guaranteeing the highest practical degree of mesh optimization.

For three application meshes, we demonstrated that the same smoothing techniques are again effective. Of the smoothing criteria considered here, we found that the maxmin sine quality measure was the most consistently effective in eliminating both small and large angles. Also, we showed that in some cases, a tetrahedron can be unimprovable by smoothing because its vertices are on the boundary of the mesh; but the tetrahedron can be removed by edge swapping. For the remaining poor-quality elements that could not be improved using our current techniques, we presented evidence that these tetrahedra tend to be clustered together. In this situation, swapping fails because local reconnection is not legal, and smoothing fails because improving one tetrahedron reduces the quality of a neighbor.

Several enhancements are being incorporated into the mesh improvement software to increase its effectiveness and efficiency. Our current software uses mesh smoothing to improve the quality of the volume mesh once the surface mesh has been generated. We plan to add surface mesh-smoothing capabilities to the optimization-based algorithm by incorporating additional constraints to bind the free vertex to the boundary surfaces. We are also interested in examining optimization-based smoothing with other measures including aspect ratio and solid angles and in developing smoothing measures appropriate for use on anisotropic meshes. We intend to improve the efficiency of our edge-swapping implementation and to investigate the use of more sophisticated local reconnection algorithms, such as that of Joe[9]. For all of the meshes discussed in this article, it is possible to find (possibly very expensive) idiosyncratic combinations of the operations described that result in a significantly better final mesh. Additional work is required to find more powerful mesh improvement techniques that will allow a more effective general prescription for mesh improvement. Finally, further work is needed to quantify the gains, if any, in solution speed and accuracy for computational science problems due to mesh improvement using these techniques.

This software is being incorporated into the SUMAA3d[23] and GRUMMP projects at Argonne National Laboratory, which will provide an integrated framework for parallel un-

structured mesh applications. Therefore, we are working to develop parallel algorithms similar to the framework given previously[16] for three-dimensional mesh smoothing and face swapping methods.

Acknowledgements

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

We thank the reviewers for their helpful suggestions. In particular, the local reconfiguration schemes are much improved because of their comments.

References

[1] I. Babuska and A. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13:214–226, 1976.

[2] I. Fried. Condition of finite element matrices generated from nonuniform meshes. *AIAA Journal*, 10:219–221, 1972.

[3] Randolph E. Bank, Andrew H. Sherman, and Alan Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman et al., editor, *Scientific Computing*, pages 3–17. IMACS/North-Holland Publishing Company, Amsterdam, 1983.

[4] Carl F. Ollivier-Gooch. Multigrid acceleration of an upwind Euler solver on unstructured meshes. *AIAA Journal*, 33(10):1822–1827, 1995.

[5] M. Rivara. Mesh refinement processes based on the generalized bisection of simplices. *SIAM Journal on Numerical Analysis*, 21:604–613, 1984.

[6] Eric Brière de l'Isle and Paul-Louis George. Optimization of tetrahedral meshes. In Ivo Babushka, William D. Henshaw, Joseph E. Oliger, Joseph E. Flaherty, John E. Hopcroft, and Tayfun Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, pages 97–127. Springer-Verlag, 1995.

[7] H. Edelsbrunner and N. Shah. Incremental topological flipping works for regular triangulations. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 43–52, 1992.

[8] Barry Joe. Three-dimensional triangulations from local transformations. *SIAM Journal on Scientific and Statistical Computing*, 10:718–741, 1989.

[9] Barry Joe. Construction of three-dimensional improved quality triangulations using local transformations. *SIAM Journal on Scientific Computing*, 16:1292–1307, 1995.

[10] E. Amezua, M. V. Hormaza, A. Hernandez, and M. B. G. Ajuria. A method of the improvement of 3D solid finite-element meshes. *Advances in Engineering Software*, 22:45–53, 1995.

[11] Scott Canann, Michael Stephenson, and Ted Blacker. Optismoothing: An optimization-driven approach to mesh smoothing. *Finite Elements in Analysis and Design*, 13:185–190, 1993.

[12] V. N. Parthasarathy and Srinivas Kodiyalam. A constrained optimization approach to finite element mesh smoothing. *Journal of Finite Elements in Analysis and Design*, 9:309–320, 1991.

[13] David A. Field. Laplacian smoothing and Delaunay triangulations. *Communications and Applied Numerical Methods*, 4:709–712, 1988.

[14] C. L. Lawson. Properties of n-dimensional triangulations. *Computer Aided Geometric Design*, 3:231–246, 1986.

[15] S. H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[16] Lori A. Freitag, Mark T. Jones, and Paul E. Plassmann. An efficient parallel algorithm for mesh smoothing. In *Proceedings of the Fourth International Meshing Roundtable*, pages 47–58. Sandia National Laboratories, 1995.

[17] C. Charalambous and A. Conn. An efficient method to solve the minimax problem directly. *SIAM Journal of Numerical Analysis*, 15(1):162–187, 1978.

[18] Christian Bischof, Lucas Roh, and Andrew Mauer. ADIC — An extensible automatic differentiation tool for ANSI-C. Preprint ANL/MCS-P626-1196, Argonne National Laboratory, Mathematics and Computer Sciences Division, 1996.

[19] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. In *8th ACM-SIAM Symp. on Discrete Algorithms*, New Orleans, to appear.

[20] Randy Bank. *PLTMG: A Software Package for Solving Ellipitc Parital Differential Equations, Users' Guide 7.0*, volume 15 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1994.

[21] Mark Shephard and Marcel Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.

[22] Randolph E. Bank and R. Kent Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, to appear.

[23] Mark T. Jones and Paul E. Plassmann. Computational results for parallel unstructured mesh computations. *Computing Systems in Engineering*, 5(4-6):297–309, 1994.

Appendix A: Supporting Tables

Table 13: Mesh quality improvement for `rand2` with swapping

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
Initial	0.10°	179.84°	2.57	8.33	14.77	4.24	2.04	0.51
Without Edge Swapping								
Minmax angle	0.57°	179.20°	1.51	5.82	11.52	2.51	1.07	0.21
Maxmin sine	0.57°	179.11°	1.32	5.33	10.86	2.53	1.09	0.21
In-sphere, then minmax angle	$6.0 \cdot 10^{-6}$°	180.00°	0.60	1.82	3.78	0.77	0.43	0.16
In-sphere, then maxmin sine	$3.5 \cdot 10^{-6}$°	180.00°	0.60	1.80	3.75	0.77	0.43	0.16
With Edge Swapping								
Minmax angle	0.57°	178.96°	0.45	2.54	6.40	0.63	0.18	0.031
Maxmin sine	0.57°	178.96°	0.23	1.51	4.82	0.75	0.25	0.046
In-sphere, then minmax angle	0.32°	178.88°	0.18	0.83	2.40	0.23	0.10	0.027
In-sphere, then maxmin sine	0.32°	178.72°	0.11	0.63	1.99	0.26	0.11	0.021

Table 14: Mesh quality improvement for `rand2` with smoothing

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles <			% Dihedral Angles >		
			6°	12°	18°	162°	168°	174°
Lap: No constraint	0.0026°	179.996°	2.45	6.90	12.21	3.46	1.84	0.63
Lap: Maxmin angle	0.64°	178.76°	1.58	6.32	12.35	3.03	1.35	0.26
Lap: Minmax angle	0.51°	178.83°	1.71	6.39	12.35	2.95	1.24	0.23
Opt: Maxmin angle	2.64°	178.35°	0.25	4.49	11.82	1.99	0.59	0.059
Opt: Minmax angle	$5.59 \cdot 10^{-5}$°	174.53°	3.62	7.69	12.93	1.11	0.21	0.00065
Opt: Maxmin cosine	$5.68 \cdot 10^{-5}$°	175.69°	3.35	7.32	12.58	1.03	0.18	0.0045
Opt: Minmax cosine	0.10°	179.84°	0.45	4.50	11.88	2.09	0.71	0.11
Opt: Maxmin sine	2.58°	177.16°	0.27	4.47	11.83	2.01	0.58	0.019

Table 15: Mesh quality improvement for `rand2` with both swapping and smoothing

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	12°	18°	% Dihedral Angles > 162°	168°	174°
Lap: No constraint	.037°	179.85°	0.18	0.52	1.11	0.25	0.14	0.064
Lap: Maxmin angle	0.32°	178.72°	0.057	0.25	0.88	0.13	0.054	0.0086
Lap: Minmax angle	0.32°	178.72°	0.059	0.26	0.89	0.13	0.056	0.0086
Opt: Maxmin angle	9.81°	169.99°	0	0.026	0.24	0.034	0.0034	0
Opt: Minmax angle	$1.27 \cdot 10^{-6}$°	164.09°	1.16	2.46	4.31	0.0017	0	0
Opt: Maxmin cosine	0.0028°	177.27°	1.08	2.31	4.20	0.011	0.0057	0.0017
Opt: Minmax cosine	10.57°	170.64°	0	0.023	0.23	0.037	0.0029	0
Opt: Maxmin sine	9.72°	167.65°	0	0.017	0.26	0.016	0	0

Table 16: Effect of the number of optimization passes on mesh improvement (`rand2` with minmax angle smoothing)

Passes	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	12°	18°	% Dihedral Angles > 162°	168°	174°
0	0.32°	178.72°	0.11	0.63	1.99	0.26	0.093	0.021
1	2.80°	175.72°	0.027	0.23	0.95	0.12	0.034	0.0023
2	4.38°	172.85°	0.0068	0.12	0.59	0.075	0.010	0
3	6.61°	174.84°	0	0.072	0.41	0.054	0.0080	0.00057
4	7.81°	171.46°	0	0.048	0.31	0.044	0.0046	0
5	9.01°	169.86°	0	0.027	0.27	0.037	0.0028	0
6	9.81°	169.99°	0	0.026	0.24	0.034	0.0034	0

Table 17: Mesh quality improvement for combined Laplacian/optimization-based smoothing (mesh `rand2`)

Case	Min. Dihed.	Max. Dihed.	% Dihedral Angles < 6°	12°	18°	% Dihedral Angles > 162°	168°	174°	Time ms
Laplacian	0.322°	178.72°	0.057	0.25	0.88	0.13	0.054	0.0085	2.08
Optimization	9.72°	167.65°	0	0.017	0.26	0.016	0	0	21.2
Combined (30)	10.58°	168.31°	0	0.022	0.22	0.014	0.00057	0	18.5
Combined (15)	9.91°	167.83°	0	0.021	0.77	0.039	0	0	3.74
Combined (10)	8.99°	169.87°	0	0.12	0.86	0.062	0.0040	0	2.56
Combined (5)	5.00°	174.27°	0.017	0.22	0.87	0.10	0.021	0.00057	2.15
Floating	8.69°	168.93°	0	0.055	0.81	0.043	0.0029	0	2.53

Table 18: Verification of mesh improvement recommendations for `tfire` mesh

Rec.	Variation	Max. Dihed.	Min. Dihed.	% Dihedrals < 6°	% Dihedrals < 12°	% Dihedrals < 18°	% Dihedrals > 162°	% Dihedrals > 168°	% Dihedrals > 174°	Time (sec)
—	Baseline	4.91°	175.00°	.0018	.034	.11	.019	.0076	.0013	154.3
1	In-sphere swap	.00043°	180.00°	.0033	.059	.92	.015	.0058	.0012	121.9
2	BATR, No edge swap	3.18°	175.37°	.0097	.048	.13	.020	.010	.0018	99.1
	BATR, Angle, no edge	4.02°	175.87°	.0075	.046	.13	.018	.010	.0018	96.7
3	Angle swap	6.98°	171.41°	0	.022	.11	.015	.0044	0	142.9
	In-sphere + Angle	.0058°	179.99°	.0031	.025	.12	.013	.0039	.0009	179.3
	In-sphere + Sine	.00043°	180.00°	.0035	.012	.084	.0092	.0035	.0017	193.1
4	Optimize	4.66°	174.50°	.0018	.026	.061	.018	.0076	.0009	513.9
	Fixed thresh. 10°	4.04°	175.77°	.0040	.043	.12	.024	.010	.0013	154.1
	Fixed thresh. 15°	4.11°	175.72°	.0040	.029	.093	.018	.0080	.0013	156.9
	Lap: Maxmin angle	2.22°	178.22°	.0093	.055	.13	.032	.015	.0040	176.2
	Maxmin angle	5.30°	177.17°	.0013	.030	.10	.016	.0062	.0005	179.9
	Maxmin cosine	0.88°	176.32°	.023	.12	.41	.024	.012	.0049	268.2
	Three passes	4.76°	174.48°	.0044	.044	.12	.021	.010	.0022	136.9
	Five passes	5.31°	174.46°	.0013	.032	.11	.020	.0071	.0009	171.2
5	No BATR	4.16°	174.86°	.0062	.044	.13	.025	.011	.0013	143.5
	Multi-BATR	4.98°	174.37°	.0022	.044	.11	.019	.010	.0009	176.3