LOCAL OPTIMIZATION-BASED SIMPLICIAL MESH UNTANGLING AND IMPROVEMENT*

LORI A. FREITAG† AND PAUL PLASSMANN‡

Abstract. We present an optimization-based approach for mesh untangling that maximizes the minimum area or volume of simplicial elements in a local submesh. These functions are linear with respect to the free vertex position; thus the problem can be formulated as a linear program that is solved by using the computationally inexpensive simplex method. We prove that the function level sets are convex regardless of the position of the free vertex, and hence the local subproblem is guaranteed to converge. Maximizing the minimum area or volume of mesh elements, although well-suited for mesh untangling, is not ideal for mesh improvement, and its use often results in poor quality meshes. We therefore combine the mesh untangling technique with optimization-based mesh improvement techniques and expand previous results to show that a commonly used two-dimensional mesh quality criterion can be guaranteed to converge when starting with a valid mesh. Typical results showing the effectiveness of the combined untangling and smoothing techniques are given for both two- and three-dimensional simplicial meshes.

Keywords. Mesh Untangling, Mesh Improvement, Simplicial Mesh Quality, Mesh Smoothing

1. Introduction. Simplicial meshes often contain poorly shaped, distorted, or inverted elements that result in numerical difficulties during the solution of finite element and finite volume applications.[1] Several methods have been developed to improve element quality if the mesh is valid, that is, the mesh contains elements with positive area or volume. These techniques include local reconnection methods, such as edge or face swapping,[2, 3, 4] or node point adjustment methods such as mesh smoothing.[5, 6, 7] The most commonly used class of mesh smoothing techniques comprises local methods that operate on one vertex at a time to improve mesh quality in a neighborhood of that vertex. Some number of sweeps over the adjustable vertices are performed to achieve an overall improvement in the mesh. These techniques cannot, however, always be used when starting with a mesh with inverted elements. To address this problem, in this paper we present a local mesh "smoothing" technique designed specifically for mesh untangling. This method assumes that the mesh has valid connectivity but that the node point positions are such that some of the elements are inverted.

Local mesh smoothing techniques operate using data from the neighborhood of the grid point that is being adjusted. To illustrate this approach we show an example of such a submesh neighborhood in Figure 1.1. The submesh consists of the free vertex, v, eight incident elements, $t1, \ldots, t8$, and eight fixed vertices, $v1, \ldots, v8$. The shaded region in each submesh is the *feasible region*, which we define to be the set of possible locations for v for which all of the incident elements have a positive area. The local submesh on the left shows the free vertex in a position that is not ideal. Elements $t1$, $t2$, and $t7$ have poor quality, but all of the elements in this mesh are valid because the free vertex lies inside the feasible region. In the middle submesh, we show the same local submesh after a typical mesh smoothing operation. In local mesh smoothing techniques, the location of the free vertex is changed according to some rule or heuristic procedure based on information available at the adjacent grid points. Only the position of v is affected; adjacent vertex locations remain unchanged. The quality of elements $t1$, $t2$, and $t7$ has significantly improved by moving v toward the middle of the feasible region. The submesh on the right shows a tangled local submesh; the free

†Assistant Scientist, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. freitag@mcs.anl.gov.

‡Assistant Professor, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. plassman@cse.psu.edu. This work was supported by an Alfred P. Sloan research fellowship.

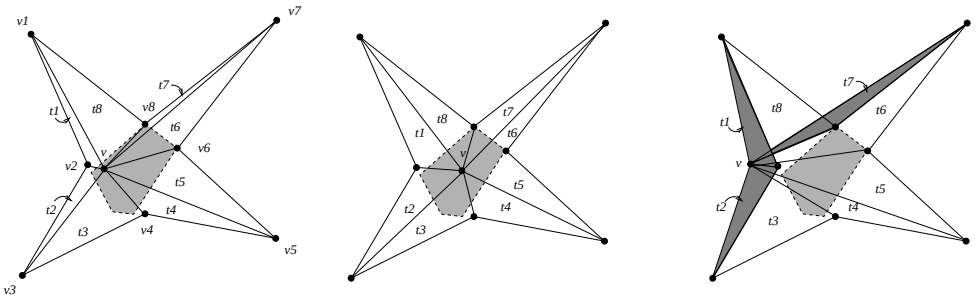

FIG. 1.1. *A local submesh consisting of a free vertex, v, to be moved and its incident vertices, $v1, \ldots, v8$, and elements, $t1 \cdots t8$ The feasible region for the free vertex location is shaded gray in all three submeshes. The submeshes show three possible locations for v. The first results in a valid but poor-quality mesh, the second in a higher-quality valid mesh, and the third in an invalid mesh with inverted elements.*

vertex v is outside the feasible region, and the shaded elements $t1$, $t2$, and $t7$ are inverted.

The most commonly used local mesh smoothing technique is Laplacian smoothing,[8, 9] which moves the free vertex to the geometric center of its incident vertices. Laplacian smoothing is computationally inexpensive but does not guarantee improvement in the element quality. In fact, it is possible to create inverted elements, and this method is therefore not guaranteed to correct an invalid mesh, even for the local subproblem. In contrast, optimization-based approaches to mesh smoothing avoid the creation of invalid elements and find the optimal location of each mesh vertex in the local submesh.[6, 7, 10, 11, 12, 13, 14] These approaches offer the advantage of guaranteed mesh improvement and validity; however, this guarantee comes at a much higher computational cost. Thus, a natural approach that has been shown to obtain high-quality elements at a low computational cost is to combine Laplacian and optimization-based smoothing techniques.[10, 15]

In previous papers, the authors have developed an optimization-based approach to mesh improvement that involves minimizing a nonsmooth, composite function on a local submesh using a technique analogous to steepest descent.[11, 15] This approach has been shown to be equivalent to generalized linear programming techniques,[16] and can be guaranteed to converge to an optimal solution given convex function level sets in the feasible region and a feasible starting point. In Figure 1.2, we show the level sets for three geometric mesh quality metrics that can be used to create a composite function: minimum angle in the local submesh, minimum sine of an angle, and scaled root mean square. The level sets are created by choosing a series of candidate locations for the free vertex both inside and outside the feasible region (three such candidate locations are shown in Figure 1.1) and evaluating the composite function at those points. The contours of the resulting function point set are plotted by using Matlab. Note that these function level sets are nonconvex if the free vertex lies outside of the feasible region. Thus, if the mesh contains inverted or tangled elements, optimization approaches using these metrics cannot be guaranteed to converge and, in fact, often diverged in preliminary tests using them for mesh untangling.

To address this problem, we have developed an optimization-based approach to mesh untangling based on maximizing the minimum area or volume of mesh elements in a local submesh. The formulation and solution are presented in Section 2. We prove that the function level sets are convex regardless of the position of the free vertex, and hence the local subproblem is guaranteed to converge. Maximizing the minimum area or volume of mesh elements, although well suited for mesh untangling, is not ideal for mesh improvement because a small, but perfectly shaped element is likely to be distorted in an effort to maximize its area. We therefore combine this technique for mesh untangling with local mesh improvement methods in a two-stage solution process. We include a brief description of the improvement method for completeness in Section 3, and expand previous results to show that the level sets for the two-dimensional mesh quality criterion minimum sine of an angle are

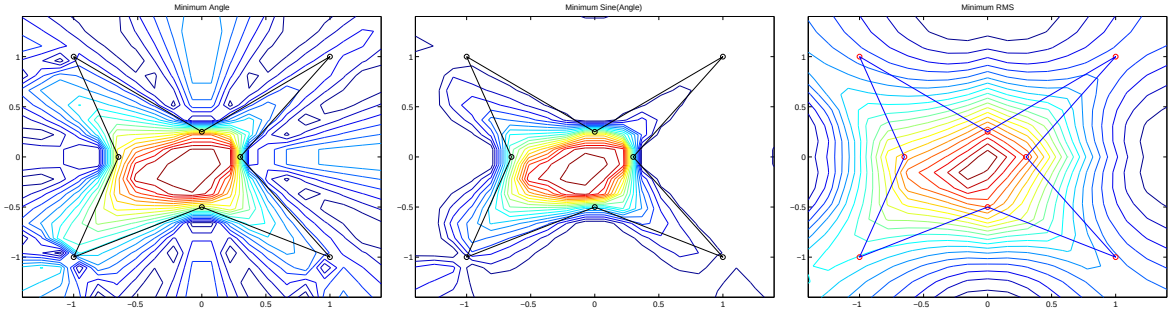

FIG. 1.2. *Level sets for the minimum angle, minimum sine of the angle, and minimum root mean square quality metrics. Each of these metrics is nonconvex outside of the feasible region.*

convex in the feasible region. Typical results showing the effectiveness of the combined untangling and smoothing techniques for both two- and three-dimensional meshes are given in Section 4.

2. Optimization-based Mesh Untangling. Local mesh untangling techniques are formulated in terms of the grid point to be adjusted, the *free vertex*, v, and that grid point's adjacent vertices, $adj(v) = \{u \mid \text{an edge exists between } v \text{ and } u\}$. Suppose $\mathbf{x}$ is the position of the free vertex; then the general form of a local untangling algorithm is given by

$$\mathbf{x}_{new} = \texttt{Untangle}(\mathbf{x}, adj(v), \text{conn}(v)),$$

where $\mathbf{x}_{new}$ is the proposed new position of v and $\text{conn}(v)$ is the connectivity information of the elements adjacent to v.

Ideally, $\mathbf{x}_{new}$ improves the local submesh in such a way that it is untangled or can be untangled in a succeeding sweep through the mesh. In this section we describe a method for performing the node point adjustment based on maximizing the minimum area or volume of a simplex contained in the local submesh. This method is computationally inexpensive, and we prove that convergence of the local subproblem is guaranteed by showing that the level sets are convex regardless of the position of the free vertex.

2.1. Formulation. The function that gives the minimum area or volume of a simplex in a local submesh is

$$f(\mathbf{x}) = \min_{1 \le i \le n} A_i(\mathbf{x}), \tag{2.1}$$

where n is the number of simplices in the local submesh, A_i is the area (volume) of simplex t_i, and $\mathbf{x}$ is the position of the free vertex. In two dimensions, if triangle t_i is defined by the free vertex position, $\mathbf{x}$, and the positions of two other vertices, $\mathbf{x}_i$ and $\mathbf{x}_j$, then A_i can be expressed as a function of the Jacobian of the element[14]

$$A_i = \frac{1}{2}\det(\mathbf{x}_i - \mathbf{x},\ \mathbf{x}_j - \mathbf{x}) = a_{x\,i}x + a_{y\,i}y + c_i, \tag{2.2}$$

where

$$a_{x\,i} = y_i - y_j, \qquad a_{y\,i} = x_j - x_i, \qquad c_i = x_i y_j - x_j y_i.$$

Similarly, in three dimensions, if tetrahedron t_i is defined by the free vertex position, $\mathbf{x}$, and the positions of three other vertices, $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$, then A_i is given by

$$A_i = \frac{1}{6}\det(\mathbf{x}_i - \mathbf{x},\ \mathbf{x}_j - \mathbf{x},\ \mathbf{x}_k - \mathbf{x}) = a_{x\,i}x + a_{y\,i}y + a_{z\,i}z + c_i, \tag{2.3}$$

where

$$a_{x\,i} = -\det \begin{bmatrix} 1 & 1 & 1 \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{bmatrix}, \qquad a_{y\,i} = -\det \begin{bmatrix} x_i & x_j & x_k \\ 1 & 1 & 1 \\ z_i & z_j & z_k \end{bmatrix}, \qquad a_{z\,i} = -\det \begin{bmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ 1 & 1 & 1 \end{bmatrix},$$

$$c_i = \det \begin{bmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{bmatrix}.$$

In both two and three dimensions, A_i is a linear function of the free vertex position, $\mathbf{x}$. We can use this fact to pose the solution of the optimization problem

$$\max \min_{1 \le i \le n} A_i(\mathbf{x}) \tag{2.4}$$

as a linear programming problem. To do this, we first construct the dual problem. Let d be the spatial dimension of the problem and n be the number of incident elements. Define the $(d+1) \times n$ matrix $\mathcal{A}$ to be the matrix whose ith column is $\left[a_{x\,i},\ a_{y\,i},\ 1\right]^T$ for $d = 2$ and $\left[a_{x\,i},\ a_{y\,i},\ a_{z\,i},\ 1\right]^T$ for $d = 3$ and π to be the $(d+1)$-vector containing the spatial coordinates of the free vertex in the first d components and the current estimate of the minimum area (volume) in the last component. Then, by definition of $\mathcal{A}$ and π,

$$\mathcal{A}^T \pi = \mathbf{c} - \mathbf{s},$$

where $\mathbf{c}$ is the n-vector containing the values of c_i defined above, and $\mathbf{s}$ is an n-vector of slack variables where the ith component, s_i, gives the difference between the area (volume) of simplex t_i and the current estimate of the minimum area (volume). Thus, the dual of the linear programming problem is

$$\begin{gathered} \max \mathbf{b}^T \pi \\ \text{subject to } \mathcal{A}^T \pi + \mathbf{s} = \mathbf{c}, \quad \mathbf{s} \ge \mathbf{0}, \end{gathered}$$

where $\mathbf{b}$ is a $(d+1)$-vector whose first d components are zero and whose last component is one, so that $\mathbf{b}^T \pi$ gives the current minimum simplex area (volume).

The primal formulation of this linear program can be written in standard form as

$$\min \mathbf{c}^T \mathbf{y} \tag{2.5}$$

$$\text{subject to } \mathcal{A}\mathbf{y} = \mathbf{b}, \quad \mathbf{y} > \mathbf{0}, \tag{2.6}$$

where $\mathbf{y}$ is the primal solution vector.

The linear program has been solved when $s_i \ge 0,\ i = 1 \cdots n$, that is, when all of the elements have an area (volume) greater than or equal to the current minimum value, and the complementarity condition $\mathbf{y}^T \mathbf{s} = 0$ has been satisfied.

2.2. Phase One Solution. The linear programming problem defined by equations (2.5)–(2.6) can be solved by using the simplex method.[17] To begin the solution process, we first solve a phase one problem to find an initial feasible point, $\mathbf{y}_0$, that satisfies $\mathcal{A}\mathbf{y}_0 = \mathbf{b}$. Given the special form of $\mathbf{b}$, this is equivalent to finding $d+1$ positive components of $\mathbf{y}_0$ (the rest are set equal to zero) such that $\sum_{i=0}^{n} y_i = 1$ and the linear combination of the first d corresponding columns of $\mathcal{A}$ sum to zero.

The phase one problem can also be formulated as a linear programming problem. A solution to the phase one problem exists if the subproblem is well-posed and is not degenerate. Let $\mathcal{F}$ be

the $d \times n$ principal submatrix of $\mathcal{A}$, and assume that there exist d columns of $\mathcal{F}$ that span $\mathcal{R}^d$. If this is not the case, the local subproblem is degenerate, and the vertices all lie on in a lower-dimensional subspace. In this case, the optimal solution is to place the free vertex anywhere in this subspace, resulting in zero volumes for all the elements, and the linear programming approach to mesh untangling is not used to solve this subproblem in this sweep through the mesh.

We also assume that none of the columns of $\mathcal{F}$ are zero. If one or more of the columns of $\mathcal{F}$ are zero, then at least two of the incident vertices are co-located at the same point in space, resulting in a simplex of zero area (volume) regardless of the position of the free vertex. If this situation occurs, one of the co-located vertices is removed from the local submesh and the untangling method is restarted with the reduced incident vertex set. Note that the co-located vertex is not removed from the global mesh problem, just from the current local submesh.

If the free vertex is an interior vertex of a valid triangulation and the corresponding subproblem is not degenerate, there must be $d+1$ vectors (corresponding to the normals to the faces opposite to the free vertex) that can be used to solve the phase one problem. In this case, we say the subproblem is *well-posed*.

If the problem is not degenerate, d linearly-independent columns of $\mathcal{F}$ are selected as the initial active set for the phase one linear program. Without loss of generality, assume that one vector in the active set is the last column of $\mathcal{F}$, $\mathcal{F}_n$. Then the phase one solution is formulated as follows

$$\min \hat{\mathbf{c}}^T \hat{\mathbf{y}} \tag{2.7}$$

$$\text{subject to } \hat{\mathcal{A}}\hat{\mathbf{y}} = \hat{\mathbf{b}}, \quad \hat{\mathbf{y}} > \mathbf{0} \tag{2.8}$$

where

$$\hat{\mathcal{A}}(i,:) = \left[\mathcal{F}_{i,0}, \cdots, \mathcal{F}_{i,n-1}, -\mathcal{F}_{i,n} - \sum_{j=1}^{n-1} \mathcal{F}_{i,j} * \hat{y}_j \right] \tag{2.9}$$

$$\hat{b}_i = -\mathcal{F}_{i,n}, \text{ and}$$

$$\hat{\mathbf{c}} = [0, \cdots, 0, 1].$$

The initial guess for the phase one solution will be the vector $\hat{\mathbf{y}}_0$ whose components are all zero except for those corresponding to the active set which are equal to one. This initial point is clearly a feasible point.

We use the simplex method to find an iterate $\hat{\mathbf{y}}_k$ of equations (2.7)–(2.8) that satisfies the constraints of the original formulation for mesh untangling given in equations (2.5)–(2.6), which occurs when

$$\mathcal{F}\hat{\mathbf{y}}_k + \mathcal{F}_n = \mathbf{0}. \tag{2.10}$$

We note that an iterate that does not minimize $\hat{\mathbf{c}}^T \hat{\mathbf{y}}$ may satisfy the criterion given in (2.10), and it may, therefore, be necessary to only partially solve the phase one linear program.

2.3. Convergence of the Local Subproblem. To guarantee convergence of the local optimization problem, the level sets of the composite function given in equation (2.1) must be convex and closed. Typical level sets for this function in two dimensions are shown in Figure 2.1 for three different local submeshes, including one that has fixed edges that are tangled. In each case, the level sets appear to be convex, and in this section we prove that they are convex in both two and three dimensions in the entire domain if the initial local submesh is not degenerate and the subproblem is well-posed.

In two dimensions, this function can be written as

$$f(\mathbf{x}) = \min_{1 \le i \le n} A_i(\mathbf{x}) = \min_{1 \le i \le n} \frac{1}{2} b_i h_{\perp i}(\mathbf{x}),$$

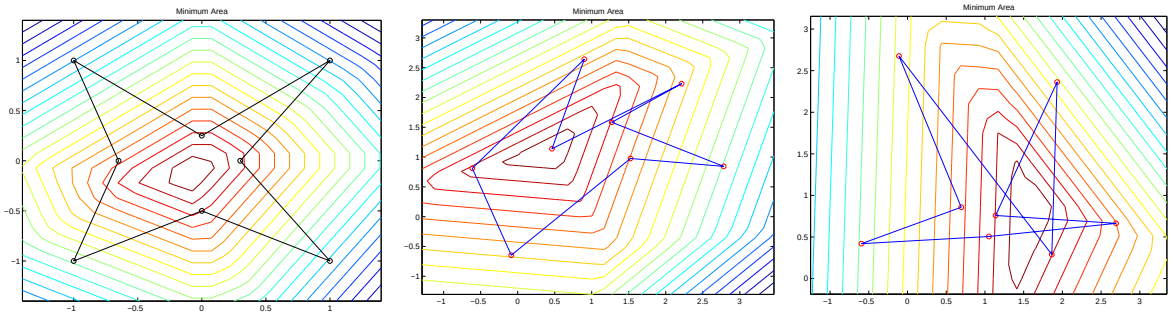

FIG. 2.1. *Level sets for the minimum element area function for three local submeshes, including one that has fixed edges that are tangled*

where b_i is the base of simplex t_i, and $h_{\perp i}(\mathbf{x})$ is the perpendicular distance between $\mathbf{x}$ and b_i. One can think of a typical level set boundary for each triangle to be a line parallel to the base of the triangle. This boundary is illustrated in Figure 2.2 as a dashed line for a triangle with base vertices s and t and area equal to $\frac{1}{2}b_i h_{\perp i}(\mathbf{x}) = c$. Thus, all triangles with base vertices s and t and area greater than c have a third vertex that lies in a half-plane as shown by the shaded area in Figure 2.2; this region is convex. The level set boundary for a local submesh is the intersection of the half planes defined by the individual triangles in that local submesh. As the intersection of convex regions is convex, these levels sets are convex.

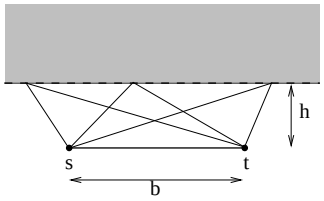

FIG. 2.2. *The boundary of a typical level set for the function* $f(\mathbf{x}) = \frac{1}{2}bh$ *is shown by the dashed line. All triangles with base vertices* s *and* t *and third vertex on the dashed line have the same area.*

One can formally show that the level sets are convex in both two and three dimensions.

THEOREM 2.1. *The level sets of the function*

$$f(\mathbf{x}) = \min_{1 \le i \le n} A_i(\mathbf{x})$$

are convex where A_i *is defined by equations (2.2) and (2.3) in two and three dimensions, respectively.*

Proof. Note that component functions $A_i(\mathbf{x})$ as given in equations (2.2) and (2.3) are linear functions of the position $\mathbf{x}$ of the free vertex. Thus, these component functions are concave. We have that the minimum of concave functions is also concave, thus $f(\mathbf{x})$ is concave.

The level sets for any concave function are convex.[18] For any two points, $\mathbf{x}_1$ and $\mathbf{x}_2$, on the boundary of the set $S(\mu) = \{\mathbf{x} \,|\, f(\mathbf{x}) \ge \mu\}$ we have that for $\alpha \in [0, 1]$,

$$\begin{aligned} f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) &\ge \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2) \\ &= \mu. \end{aligned}$$

Thus, the set $S(\mu)$ is convex and, therefore, the level sets of f are convex. ∎

As a simple corollary to this theorem we have the following fact.

LEMMA 2.2. *The feasible region for each local submesh is convex.*

Proof. The feasible region corresponds to the set $S(0) = \{\mathbf{x} \mid f(\mathbf{x}) \geq 0\}$. By Theorem 2.1, this region is convex. □

We assume that the local subproblems are not degenerate and are well-posed so that $f(\mathbf{x})$ (the minimum area or volume of an element in the submesh) is bounded. As the level sets are convex, any local maximum of $f(\mathbf{x})$ is a global maximum. Thus, any optimization algorithm guaranteed to find a local maximum (such as the linear programming approach presented above) is guaranteed to determine the global maximum for the local subproblem. Because the level sets for $f(\mathbf{x})$ are convex on the entire domain, this approach converges to the global maximum from any starting point for the free vertex.

3. Local Node Point Smoothing Techniques. The mesh smoothing method is formulated in the same manner as the optimization-based untangling technique; that is, given a free vertex, v, and its adjacent vertices and elements, the new position of the free vertex, $\mathbf{x}_{new}$, is given by the general operation

$$\mathbf{x}_{new} = \texttt{Smooth}(\mathbf{x}, adj(v), \text{conn}(v)).$$

Ideally, the new location of the free vertex will improve the mesh according to some measure of mesh quality such as dihedral angle or element aspect ratio. The action of the function `Smooth` is determined by the particular algorithm chosen, and in this section we describe four methods for performing the node point adjustment.

3.1. Laplacian and "Smart" Laplacian Smoothing. For Laplacian smoothing, the smoothing operator moves the free vertex to the geometric center of the adjacent grid points. No effort is made to ensure that mesh quality is improved, and poor quality or even invalid elements can result from the use of this technique. For an example in which Laplacian smoothing creates an invalid mesh, consider the leftmost submesh in Figure 3.1. The center submesh the shows the results of Laplacian smoothing; the free vertex position has moved outside the feasible region, and elements $t2$ and $t3$ have become inverted. However, the technique is computationally inexpensive, easy to implement, and therefore commonly used.

A simple variant of Laplacian smoothing, which we call "smart" Laplacian smoothing, relocates the free vertex to the geometric center of the adjacent grid points only if the quality of the local mesh is improved according to some mesh quality measure.

3.2. Optimization-based Smoothing. Like optimization-based mesh untangling, the optimization approach to mesh improvement finds the position $\mathbf{x}^*$ that maximizes the composite function

$$f(\mathbf{x}) = \min_{1 \leq i \leq n} q_i(\mathbf{x}), \tag{3.1}$$

where $q_i(\mathbf{x})$ is a mesh quality metric such as minimum angle, sine of the minimum angle, or aspect ratio of an element. Typically, each $q_i(\mathbf{x})$ is a nonlinear, smooth, and continuously differentiable function, and multiple functions can obtain the minimum value of the composite function. Hence, the composite function $f(\mathbf{x})$ has discontinuous partial derivatives where the active set changes from one set of functions to another set. Let the minimum value of the functions evaluated at $\mathbf{x}$ be called the *active value*, and the set of functions that obtain that value, the *active set*, be denoted by $\mathcal{S}(\mathbf{x})$.

We solve this nonsmooth optimization problem using an analogue of the steepest descent method for smooth functions. The search direction, $\mathbf{s}$, at each step is the steepest descent direction derived from all possible convex linear combinations of the gradients in $\mathcal{S}(\mathbf{x})$. This direction is computed by solving the quadratic programming problem

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

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

for the β_i. The line search subproblem along $\mathbf{s}$ is solved by predicting the points at which the active set $\mathcal{S}$ will change. These points are found by computing the intersection of the projection of a current active function in the search direction with the linear approximation of each $q_i(\mathbf{x})$ given by the first-order Taylor series approximation. 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 achieved by moving v 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-based smoothing algorithm is described in more detail elsewhere.[11, 15] We note that similar local optimization-based smoothing methods have been proposed for a variety of optimization procedures and mesh quality measures.[12, 19, 13, 14]

To illustrate the benefits of optimization-based smoothing compared with Laplacian smoothing, we consider the initial local submesh drawn in the leftmost figure of Figure 3.1. Recall that the center submesh shows the results of Laplacian smoothing, which created two inverted elements. In the rightmost figure we show the results of optimization-based smoothing. The local submesh has significantly improved quality, and all of the elements remain valid.

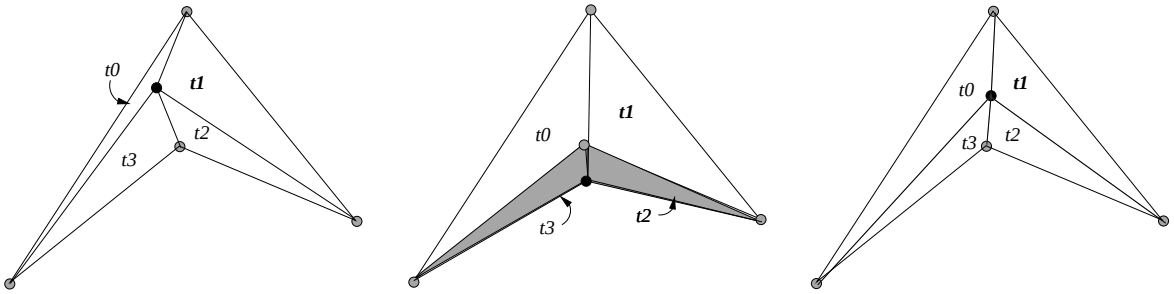

FIG. 3.1. *A local submesh that shows that Laplacian smoothing can sometimes fail. The original local submesh is shown in the leftmost figure. The center figure shows the results of Laplacian smoothing, which is a tangled mesh. In the rightmost figure we show the results of optimization-based smoothing using the combined approach.*

Experimental results have demonstrated the effectiveness of the optimization-based method compared with Laplacian smoothing for two- and three-dimensional simplicial meshes.[11, 20] In each case the optimization-based method with the quality metric minimum sine of an angle was effective at eliminating extremal angles from the mesh whereas the Laplacian smoother is often unable to significantly improve the most severely distorted elements. The corresponding increase in computational cost is approximately a factor of four in two dimensions and a factor of ten in three dimensions. These results also showed that three sweeps of the mesh were usually sufficient to improve the mesh quality; additional sweeps offered minimal incremental improvement.

3.3. Combining Laplacian and Optimization-based Smoothing. A number of approaches that combine Laplacian and optimization-based smoothing can be used to improve the mesh as effectively as optimization-based smoothing used alone at a fraction of the cost.[15] The approach used for the examples in Section 4 of this paper is to try smart Laplacian smoothing as the first step for every subproblem. If the active value in the local mesh after this step exceeds a user-defined threshold value (in this case, 30° in two dimensions and 15° in three dimensions), the local algorithm terminates; otherwise, optimization-based smoothing is performed. We note that other approaches exist for combining Laplacian and optimization-based smoothing.[15, 13]

3.4. Convergence of the Local Subproblem. We now show that the level sets for the composite function

$$f(\mathbf{x}) = \min_{1 \le i \le n} \sin\theta_i(\mathbf{x}) \tag{3.2}$$

are convex in the feasible region of a two-dimensional local submesh. Thus, the convergence of the local submesh problem can be theoretically guaranteed. In practice, 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 are satisfied, indicating that we have found a local maximum $\mathbf{x}^*$.[21]

LEMMA 3.1. *Let $\theta_i(\mathbf{x})$ be an element angle such that $\sin\theta_i(\mathbf{x}) = f(\mathbf{x})$ for any $\mathbf{x}$ in the strictly feasible region. Then we have that $\theta_i(\mathbf{x}) \le \frac{\pi}{2}$.*

Proof. We prove this lemma by contradiction. Let $\theta_i(\mathbf{x}) > \frac{\pi}{2}$ such that $\sin\theta_i(\mathbf{x}) = f(\mathbf{x})$, that is $\theta_i(\mathbf{x})$ is in the active set at $\mathbf{x}$. Let θ_s and θ_t be the other two angles in the element containing θ_i. These three angles sum to π, thus both θ_s and θ_t must be acute and, because $\mathbf{x}$ is strictly feasible, nonzero. We have that

$$\begin{aligned} \sin(\theta_i) &= \sin(\pi - \theta_s - \theta_t) \\ &= \sin(\theta_s + \theta_t) \\ &> \sin(\theta_s). \end{aligned}$$

This is contradicts our assumption that $\sin\theta_i(\mathbf{x}) = f(\mathbf{x})$, thus we have that $\theta_i(\mathbf{x}) \le \frac{\pi}{2}$. □

THEOREM 3.2. *The level sets, S, of the function $f(\mathbf{x}) = \min_{1 \le i \le n} \sin\theta_i(\mathbf{x})$ are convex in the feasible region of a two-dimensional local submesh.*

Proof. By Lemma 3.1 we have that any angle $\theta_i(\mathbf{x})$ with $\sin\theta_i(\mathbf{x}) = f(\mathbf{x})$ for $\mathbf{x}$ in the feasible region must satisfy $0 \le \theta_i(\mathbf{x}) \le \frac{\pi}{2}$. As sine is monotonic for angles in this range, it is sufficient to show that the levels sets of the function

$$g(\mathbf{x}) = \min_{1 \le i \le n} \theta_i(\mathbf{x})$$

are convex. This fact has been shown previously,[16, 22] and we briefly review the argument here.

First consider angles $\theta_j(\mathbf{x})$ at triangle vertices other than the free vertex, call these angles external angles. Let $S_j(\theta)$ be the set of points in the feasible region, denoted $\mathcal{R}_\mathcal{F}$, which form an angle $\theta \le \theta_j(\mathbf{x})$ with the external triangle edges, that is, let $S_j(\theta) = \{\mathbf{x} \in \mathcal{R}_\mathcal{F} \,|\, \theta \le \theta_j(\mathbf{x})\}$ for $0 \le \theta \le \frac{\pi}{2}$. For example, in the left submesh in Figure 3.2, we show the set $S_j(\theta)$ for the angle associated with vertex s and formed with the external edge (s,t). This region is convex.

Internal angles are formed by the free vertex, v, and the line segments connecting v to the other two vertices of each external triangle edge. Recall the geometric fact that the set of points where this angle is a constant is the circle containing these three vertices where v can be any point on the boundary of the circle. Thus, the set $S_j(\theta)$ for an internal angle is this circle intersected with the feasible region. An example of $S_j(\theta)$ is illustrated by the shaded region the right submesh in Figure 3.2 for free vertex v and the external triangle edge vertices r and s. This set is also convex.

The set $S(\theta) = \{\mathbf{x} \,|\, \theta \le g(\mathbf{x})\}$ for $0 \le \theta$ is the intersection of the sets $S_j(\theta)$. As the intersection of convex sets is convex, we have that $S(\theta)$ is convex. Thus, the level sets for $g(\mathbf{x})$ and, therefore, the level sets for $f(\mathbf{x})$ are convex in the feasible region. □

In Figure 3.3 we illustrate one of these level sets for a subproblem example. The two points $\mathbf{x}$ and $\mathbf{y}$ are on the level set boundary because of the wedge-shaped feasible regions corresponding to the two exterior angles originating from the vertex t. Note that the level set is convex and contained in the interior of the feasible region.

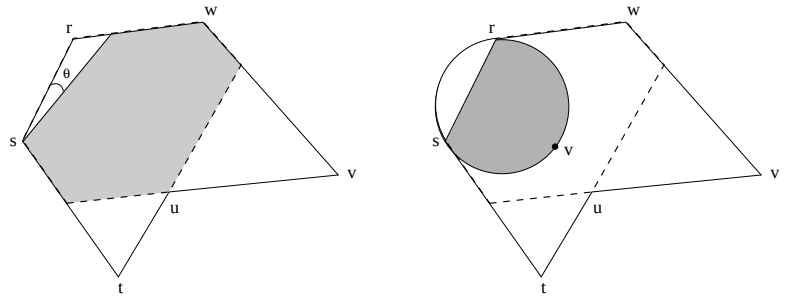

FIG. 3.2. *The level sets for external and internal angles in the local submesh. In each figure, the dashed line represents the feasible region. In the left figure, the shaded region shows the set $S_j(\theta)$ for the exterior angle associated with vertex s and the edge (s,t). In the right figure, the shaded region shows the set $S_j(\theta)$ of the interior angle associated with triangle vrs.*

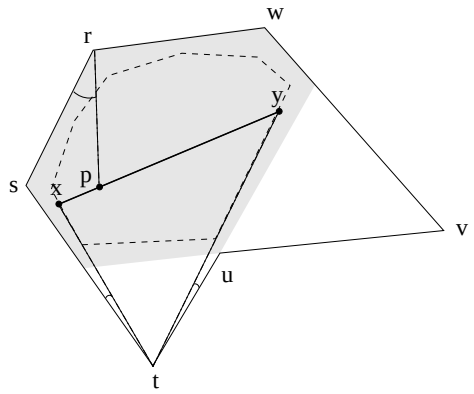

FIG. 3.3. *A local submesh in which the feasible region is shaded gray. The dashed line represents a typical level set, the boundary of $S(\theta)$, for the function $f(\mathbf{x}) = \min_{1\le i\le n} \sin\theta_i(\mathbf{x})$. The points $\mathbf{x}$ and $\mathbf{y}$ are two arbitrary points on the boundary of S. The level set is convex if $\mathbf{p} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in S$ for $0 \le \alpha \le 1$.*

4. Typical Results for Mesh Untangling and Improvement. To show typical results for the mesh untangling problem, we start with a two-dimensional Delaunay mesh created using the Triangle package[23] and a three-dimensional tetrahedral mesh created using the GRUMMP package.[24] We perturb a percentage of the nodes in these meshes a given distance to create new meshes that have valid connectivity, but invalid elements with negative area. In the discussion that follows, let P be the percentage of nodes perturbed and h be the average edge length in the mesh.

The initial two-dimensional mesh, which is shown in Figure 4.1, has 333 elements and an initial minimum angle of 22°. We perturb 10 percent of the nodes in this mesh a distance, h, to create a tangled mesh containing 28 invalid elements; a portion of this tangled mesh is shown in the leftmost figure of Figure 4.2. The perturbed mesh is untangled by using the linear programming approach described in Section 2. The untangling process stops when all of the elements are valid or 40 sweeps through the mesh have been performed. In this case, 2 sweeps through the mesh were required to eliminate all of the invalid elements. The second mesh in Figure 4.2 shows the untangled mesh. Although this mesh is valid, the untangling procedure results in meshes of extremely poor quality; minimum angles of 10^{-3} degrees are typical. In this case the minimum angle following mesh untangling is .0856°. We therefore follow the mesh untangling procedure with three and ten passes of optimization-based smoothing using the combined approach described in Section 3 and the quality metric maximize the minimum sine. The results are shown in the last two meshes in Figure

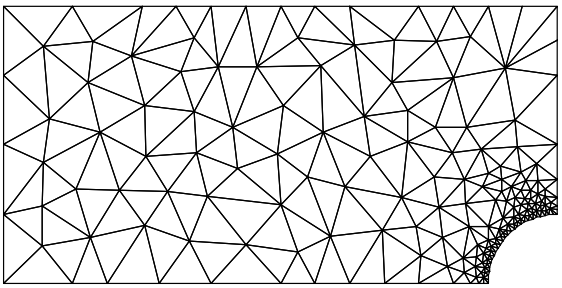

FIG. 4.1. *The initial mesh created using the Triangle package*

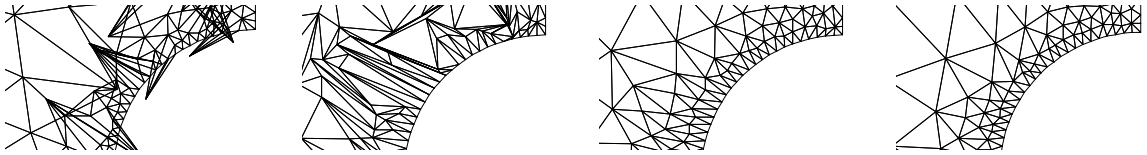

FIG. 4.2. *Typical results for mesh untangling using the linear programming approach. The mesh on the left is the original, tangled mesh; the mesh in the middle is the same mesh after untangling; the two meshes on the right are the same mesh after three and ten passes of mesh smoothing, respectively.*

4.2. The meshes have a minimum angle of 17.6° and 29.9°, respectively.

The initial three-dimensional mesh is shown in Figure 4.3; this mesh has 11,098 elements and an initial minimum dihedral angle of .657°. We again perturb this mesh in such a way that 10 percent of the nodes are moved a distance h from their original position, creating a mesh with 910 invalid elements. In this case, four sweeps through the mesh using the untangling procedure were required to create a valid mesh with a minimum dihedral angle of .0372°. The untangling procedure is again followed by three passes of mesh smoothing using the combined approach and a mesh quality metric of minimum sine of the dihedral angle, resulting in a final mesh with a minimum dihedral angle of 4.34°. In this case, the mesh fails to improve after three passes, and ten passes of mesh smoothing are not performed.

To evaluate the performance of the mesh untangling procedure as a function of the number of invalid elements and the amount of node perturbation, we created two series of perturbed meshes for each of the initial meshes. The perturbation direction for a particular node is randomly chosen, but for each series of meshes the same nodes are perturbed in the same direction. In the first series, the magnitude of the perturbation is fixed to be the average element edge length, h, and additional nodes are perturbed in each successive mesh in the series. In the second series of meshes, the number of nodes that are perturbed is constant, but the amount of the perturbation is increased in each successive mesh.

In Tables 4.1 and 4.2 we present untangling and improvement results for the two- and three-dimensional mesh series, respectively. In these tables P is the percentage of perturbed nodes, D is the distance they are perturbed, N is the resulting number of invalid elements, S_U is the number of sweeps required to untangle the mesh, Min. θ_U is the minimum (dihedral) angle in the mesh following untangling, and Min. θ_S is the minimum dihedral angle after mesh smoothing.

These results clearly show that the amount a grid point is perturbed significantly increases the number of untangling passes required to create a valid mesh and decreases the effectiveness of three passes of mesh smoothing. We note that if 10 passes of mesh smoothing were used, results of $\theta_S = 17.6°$ and $\theta_S = 4.34°$ were achieved for the last two- and three-dimensional test cases, respectively. The cost of mesh untangling per grid point is approximately half the cost of mesh

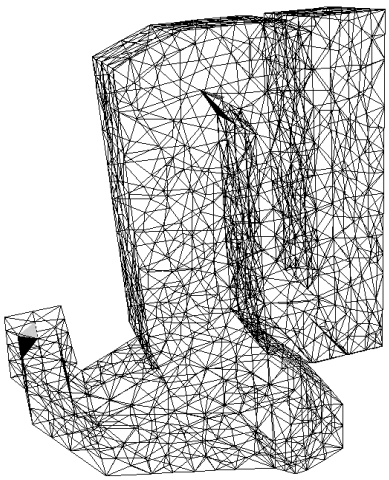

FIG. 4.3. *The initial three-dimensional tetrahedral mesh based on the geometry of a tire incinerator*

TABLE 4.1
Untangling results for a series of 2D meshes

			Untangling				Smoothing		
P	D	N	S_U	Total Time (s)	Time/ Call (s)	Min. θ_U	Total Time (s)	Time/ Call (s)	Min. θ_S
5	h	15	3	1.36e-01	3.36e-04	4.46e-01	1.58e-01	3.90e-04	14.5
10	h	28	2	6.99e-02	2.59e-04	8.56e-02	1.61e-01	3.97e-04	17.6
25	h	51	3	1.04e-01	2.56e-04	5.70e-03	2.21e-01	5.46e-04	14.9
50	h	93	4	1.35e-01	2.50e-04	6.20e-01	2.15e-01	5.32e-04	19.1
25	h	51	3	1.04e-01	2.56e-04	5.70e-03	2.21e-01	5.46e-04	14.9
25	$2h$	73	4	1.33e-01	2.46e-04	2.10e-01	2.30e-01	5.68e-04	12.3
25	$4h$	92	5	1.61e-01	2.39e-04	6.87e-02	2.78e-01	6.86e-04	8.86
25	$8h$	95	10	3.39e-01	2.51e-04	2.33e-01	2.79e-01	6.89e-04	2.52

smoothing.

To further reduce the costs of the mesh untangling procedure, we experimented with combining the optimization-based approach with Laplacian smoothing. We considered four different approaches to mesh untangling:

1. Laplacian smoothing used alone (the Laplacian-smooth or LS method),
2. optimization-based untangling used alone (untangle-smooth or US method),
3. Laplacian smoothing on each local submesh, followed immediately by optimization-based untangling if the local submesh is still invalid (the combined-smooth or CS method), and
4. a few sweeps through the mesh using Laplacian smoothing only followed by sweeps through the mesh using optimization-based untangling only (the Laplacian-untangle-smooth or LUS method).

In each case, mesh untangling is followed by three passes of mesh smoothing to improve the quality of the final mesh.

In two dimensions, the LS method successfully untangled each test case. We therefore recom-

TABLE 4.2
Untangling results for a series of 3D meshes

			Untangling				Smoothing		
P	D	N	S_U	Total Time (s)	Time/ Call (s)	Min. θ_U	Total Time (s)	Time/ Call (s)	Min. θ_S
5	h	447	3	8.94e+00	2.25e-03	3.61e-03	1.65e+01	4.17e-03	4.34
10	h	910	3	9.22e+00	2.32e-03	6.32e-03	1.75e+01	4.41e-03	4.34
25	h	2030	5	1.48e+01	2.25e-03	1.42e-03	2.03e+01	5.11e-03	4.34
50	h	3277	7	2.08e+01	2.24e-03	3.92e-01	2.14e+01	5.41e-03	4.34
25	h	2030	5	1.48e+01	2.25e-03	1.42e-03	2.03e+01	5.11e-03	4.34
25	$2h$	3218	9	2.74e+01	2.23e-03	0.00e+00	2.49e+01	6.27e-03	4.34
25	$4h$	3667	9	2.67e+01	2.24e-03	1.04e-05	2.57e+01	6.48e-03	4.34
25	$8h$	3891	29	8.77e+01	2.29e-03	0.00e+00	3.23e+01	8.13e-03	1.74

mend first attempting to untangle a mesh using Laplacian smoothing only. In three dimensions, however, the LS method failed to untangle any of the test cases. In Table 4.3 we give results for each of the three other techniques. For each technique, we give the number of sweeps required to untangle the mesh, S_U, the time required to untangle the mesh in seconds, and the average minimum element dihedral angle in the mesh following untangling, Min. θ_U. For the LUS method, we used three passes of Laplacian smoothing. The number of sweeps needed for mesh untangling is reported as 3–X for that technique, where X is the number of optimization-based sweeps.

TABLE 4.3
Untangling results for three different techniques on the three-dimensional mesh series

			U-S Method			C-S Method			L-U-S Method		
P	D	N	S_U	Untangle Time (s)	Min. θ_U	S_U	Untangle Time (s)	Min. θ_U	S_U	Untangle Time (s)	Min. θ_U
5	h	447	3	8.94	28.3	3	1.28	43.8	3–1	2.92	36.1
10	h	910	3	9.22	27.3	4	2.48	44.4	3–1	2.93	36.1
25	h	2030	5	14.8	26.3	5	4.54	44.2	3–1	2.92	44.7
50	h	3277	7	20.8	26.4	12	7.02	44.3	3–1	2.93	36.1
25	$2h$	3218	9	27.4	25.4	12	8.02	44.2	3–1	2.93	36.1
25	$4h$	3667	9	26.7	24.9	28	12.1	44.4	3–1	2.93	36.1
25	$8h$	3891	29	87.7	22.1	–	–	–	3–1	3.25	35.9

Using approaches that combine Laplacian smoothing with optimization-based untangling result in higher quality meshes at lower computational cost than optimization-based untangling used alone. The LUS method is the recommended method. It requires only one sweep of optimization-based untangling for each case, and the total time to untangle the mesh is constant and smaller than the US method used alone for all cases and smaller than the CS method for all but two cases. In addition, the quality of the final untangled mesh as measured by the average minimum dihedral angle falls between the US and CS method results. In general, the CS method required more iterations to converge than the US method, but each iteration was much less expensive, with the average time to process a local submesh being 3.75^{-4} seconds compared with 2.32^{-3} seconds. In one case, however, the CS method failed to converge in less than 80 iterations. In Figure 4.4 we show the convergence history as a function of both the number of invalid elements in the mesh and the minimum tetrahedral volume for each of the three techniques for the $P = 25$, $D = 8h$ case. We

use linear-log plots so that the details of the final iterations can be clearly seen, and therefore must plot the log of the negative minimum tetrahedral volume in the right figure. Because Laplacian smoothing is not guaranteed to improve a local submesh, the CS method falls into a cyclic pattern after 60 iterations, and the minimum tetrahedral volume fails to increase after that point.

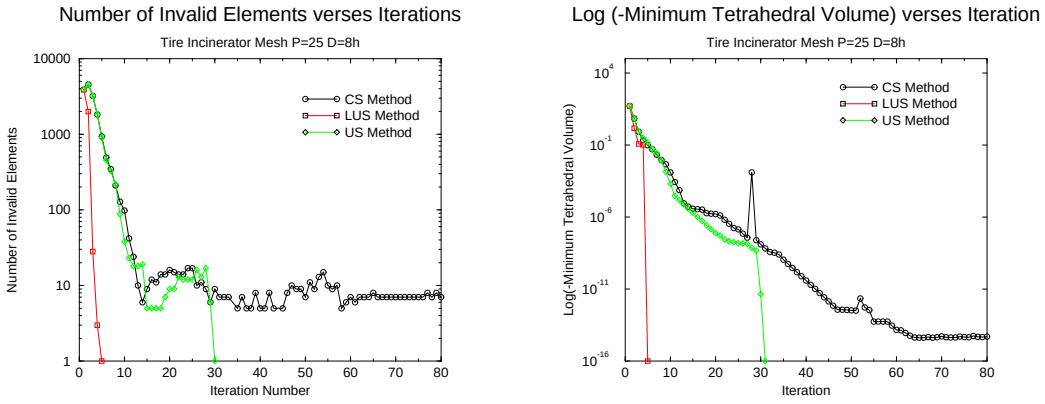

FIG. 4.4. *The number of invalid elements and minimum tetrahedral volume (displayed as log(-volume)) for the three techniques CS, LUS, and US.*

5. Conclusions. In this paper we presented an optimization-based approach to simplicial mesh untangling that uses linear programming techniques to maximize the minimum area or volume in a local submesh. As with optimization-based smoothing techniques, this method should be used in combination with less expensive methods for mesh smoothing. In particular, our experience shows that Laplacian smoothing is often effective even though it offers no guarantee of mesh improvement. Thus, high-quality, valid meshes can be achieved at low computational cost by combining Laplacian smoothing, or its variants, with a strategic use of optimization-based techniques. Finally, we proved that the linear programming approach is guaranteed to converge for local simplicial submeshes, but the global convergence of both optimization-based mesh improvement and optimization-based mesh untangling is still an open question for general domains and mixed or hexahedral element types.

Acknowledgments. We acknowledge the CUBIT research group at Sandia National Laboratory for providing both a stimulating work environment for the first author's sabbatical visit and also the motivation for pursuing this line of research. In particular, we thank Patrick Knupp of Sandia National Laboratories for his many insightful ideas and suggestions during the course of this research. His related work on optimization-based smoothing and untangling using Jacobian-based quality measures was critical in the overall formulation of the mesh untangling procedure presented in this paper.

REFERENCES

[1] Lori Freitag and Carl Ollivier-Gooch. A cost/benefit analysis of simplicial mesh improvement as measured by solution efficiency. Preprint ANL/MCS-P722-0598, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1998. To appear in International Journal of Computational Geometry.

[2] 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.

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

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

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] R. E. Bank and R. K. Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, June 1997.

[13] Matthew L. Staten Scott A. Canann, Joseph R. Tristano. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. In *Proceedings of the 7th International Meshing Roundtable*, pages 479–494. Sandia National Laboratories, 1998.

[14] Patrick Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part 1 - A framework for surface mesh optimization. Technical Report SAND 99-0110J, Sandia National Laboratories, 1999.

[15] Lori Freitag. On combining Laplacian and optimization-based smoothing techniques. In *Trends in Unstructured Mesh Generation*, volume AMD-Vol. 220, pages 37–44. ASME Applied Mechanics Division, 1997.

[16] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. In *8th ACM-SIAM Symp. on Discrete Algorithms*, pages 528–537, 1997.

[17] P. Gill, W. Murray, and Margaret Wright. *Practical Optimization*. Academic Press, 1981.

[18] R. Fletcher. *Practical Methods of Optimization (Second Edition)*. John Wiley and Sons, 1987.

[19] Mark Shephard and Marcel Georges. Automatic three-dimensional mesh generation by the finite octree technique. Technical Report SCOREC Report No. 1-1991, Scientific Computation Research Center, Rensselaer Polytechnic Institute, 1991.

[20] Lori Freitag and Carl Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal of Numerical Methods in Engineering*, 40:3979–4002, 1997.

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

[22] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. Technical report, Freie Univ. Berlin, Fachb. Mathematik, August 1992.

[23] Jonathan Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *Proceedings of the First Workshop on Applied Computational Geometry*, pages 124–133, Philadelphia, Pennsylvania, May 1996. ACM.

[24] Carl Ollivier-Gooch. Beta Release of the GRUMMP software, http://tetra.mech.ubc.ca/GRUMMP/, 1998.