UNTANGLING MAPPED QUADRILATERAL MESHES WITH CONCAVE BOUNDARIES[1]

Lori A. Freitag[†] & Paul E. Plassmann[*]

[†]*Mathematics and Computer Science Division*
Argonne National Laboratory, Argonne, IL 60439
freitag@mcs.anl.gov.

[*]*Department of Computer Science and Engineering*
The Pennsylvania State University, University Park, PA 16802
plassman@cse.psu.edu

ABSTRACT

The generation of a valid computational mesh is an essential step in the solution of many complex scientific and engineering applications. In this paper we present a new, robust algorithm, and several variants, for untangling invalid quadrilateral meshes. The primary computational aspect of the algorithm is the solution of a sequence of local linear programs, one for each interior mesh vertex. We show that the optimal solution to these local subproblems can be guaranteed and found efficiently. We present experimental results showing the effectiveness of this approach for problems where invalid, or negative area, elements can arise near highly concave domain boundaries or boundaries with sharp corners.

Keywords: Quadrilateral Mesh Untangling, Mesh Improvement, Mesh Quality, Mesh Smoothing

1. INTRODUCTION

The generation of a valid, high quality computational mesh is an essential, although often difficult, step in the numerical solution of partial differential equations on complex problem domains. Much work has been done on the development and implementation of algorithms to improve the quality of a mesh through topological changes, such as edge or face flipping [7, 14, 15], alone or in combination with geometric changes, such as vertex smoothing [2, 5, 19]. These approaches usually demand that the initial mesh be valid; however, recent work has begun to consider the problem of recovering a valid mesh from a topologically correct mesh that contains inverted, or neg-

[1]THE WORK OF THE FIRST AUTHOR WAS SUPPORTED BY THE UNIVERSITY OF CHICAGO AS OPERATOR OF ARGONNE NATIONAL LABORATORY ("ARGONNE") UNDER CONTRACT NO. W-31-109-ENG-38 WITH THE U.S. DEPARTMENT OF ENERGY. THE U.S. GOVERNMENT RETAINS FOR ITSELF, AND OTHERS ACTING ON ITS BEHALF, A PAID-UP, NONEXCLUSIVE, IRREVOCABLE WORLDWIDE LICENSE IN SAID ARTICLE TO REPRODUCE, PREPARE DERIVATIVE WORKS, DISTRIBUTE COPIES TO THE PUBLIC, AND PERFORM PUBLICLY AND DISPLAY PUBLICLY, BY OR ON BEHALF OF THE GOVERNMENT. THE WORK BY THE SECOND AUTHOR WAS SUPPORTED BY NSF GRANT ACI-9908057, DOE GRANT DG-FG02-99ER25373, AND THE ALFRED P. SLOAN FOUNDATION.

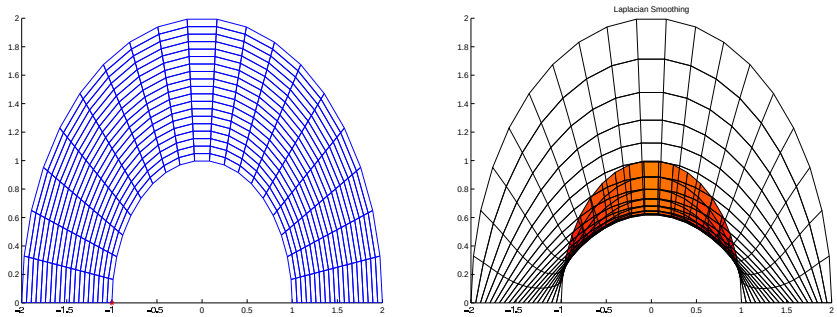

Figure 1. Example showing Laplacian smoothing creating an invalid quadrilateral mesh

ative area, elements [4, 12, 17]. Such elements can arise either during the initial generation of a mesh or during the subsequent modification of a valid mesh through dynamic mesh techniques such as Lagrangian (moving mesh) discretization techniques or adaptive mesh refinement. In this paper, we extend a method developed for untangling simplicial meshes [12] to quadrilateral meshes and discuss how this new approach can be extended for use with hexahedral meshes. We begin by reviewing the local mesh smoothing approaches, both heuristic and optimization-based, that motivate our approach.

Local mesh smoothing methods operate on one mesh vertex at a time to improve mesh quality in a neighborhood of that vertex. Each adjustable, or *free*, vertex is geometrically repositioned according to information at the incident vertices or elements—ideally the vertex's new position will improve the quality of the mesh according to some metric such as aspect ratio or element condition number [16]. To achieve an overall improvement in mesh quality, some number of sweeps over the free vertices of the mesh are performed.

The simplest, most computationally inexpensive, and commonly used local mesh smoothing technique is Laplacian smoothing. With this method the free vertex is moved to the geometric center of its incident vertices [9, 18]. A distinct disadvantage with this method is that it is not guaranteed to improve element quality. In fact, it is possible to create inverted elements as is illustrated in Figure 1. The figure on the right shows the results of one hundred passes of Laplacian smoothing applied to the initial mesh shown on the left. The shaded quadrilateral elements near the concave boundary have been pulled outside of the computational domain and inverted.

In contrast, optimization-based approaches to mesh smoothing aim to place the free vertex in a location that optimizes some measure of mesh quality. A number of approaches have been considered; each is differentiated by the particular choices of mesh quality metric and optimization technique. For example, several researchers have used mesh quality metrics that measure element or node quality *a priori* using geometric criteria, e.g., [6, 16], and a few have used *a posteriori* metrics that incorporate information from the application solution, e.g., [3]. The optimization techniques used have targeted improvement of either the average element quality, e.g., [16] or the extremal element quality, e.g., [12, 20]. Convergence of the local subproblem to the optimal solution can often be theoretically guaranteed [1, 12]; thus, these approaches are more robust than Laplacian smoothing but have a greater computational cost. To reduce the overall computational cost of these methods without sacrificing robustness, a number of approaches that combine Laplacian and optimization-based smoothing [10, 20, 21] or that combine variants of optimization-based approaches [11] have been developed.

The convergence results for these optimization-based approaches typically hold only when starting with a valid initial mesh. If elements in the local submesh are inverted, the level sets of the objective function are often no longer convex [12], and the methods fail. To address this situation, recent work has been done to develop optimization-based untangling approaches for both simplicial [11, 12] and quadrilateral and hexahedral [17, 4] meshes.

In this paper, we extend the mesh untangling approach presented in [12] to quadrilateral meshes. We review the solution techniques used for simplicial meshes in §2 and show how they can be applied to quadrilateral meshes in §3. We develop several variants of the algorithm and discuss the convergence guarantees and solution uniqueness in each case. In §4, we demonstrate the effectiveness of our approach in untangling inverted elements near concave boundaries. We close by offering concluding remarks in §5.

2. SIMPLICIAL MESH UNTANGLING

To correct inverted triangles and tetrahedra, we recently developed an optimization based approach to simplicial mesh untangling [12]. An untangled simplicial mesh is a mesh whose elements all have positive area, for the case of triangles, or volume, for the tetrahedral case. As with the smoothing algorithms, the untangling technique solves a local optimization problem for each interior vertex by moving the geometric position of this vertex to maximize a quality metric. Given the position of vertex v, $\mathbf{x}$, the quality metric used for untangling is based on the areas (or volumes) of the elements T_i that contain this vertex. We define the quality metric for each element, as a function of the vertex position, to be

$$Q_{T_i,v}(\mathbf{x}) = A(T_i), \tag{1}$$

where $A(T_i)$ is the area (or volume) of element T_i. If the number of elements adjacent to the vertex is n, the objective function to be maximized is

$$f_Q(\mathbf{x}) = \min_{1 \le i \le n} Q_{T_i,v}(\mathbf{x}). \tag{2}$$

We note that given the positions of the other vertices in the element, T, the area $A(T)$ can be expressed as a function of the Jacobian of the element [16].

In both two and three dimensions, $A(T_i)$ is a linear function of the free vertex position $\mathbf{x}$. We use this fact to pose the optimization problem as a linear programming problem which we solve using the computationally efficient and robust simplex method [13]. In practice the optimization problem is solved in two steps. First, a phase one linear programming problem is solved to determine a feasible point. Second, the actual optimization problem is solved using the feasible point as the initial point. The linear program has been solved when all of the elements have an area (volume) greater than or equal to the current minimum value and the complementarity condition has been satisfied. More details on this method can be found in [12].

These problems can always be solved if the local subproblem is well-posed. We define a well-posed local subproblem to be one in which 1) the incident vertices (i.e., all vertices that are adjacent to v in the mesh) do not all lie in a lower-dimensional subspace and 2) no two incident vertices are co-located at the same point in space. When the subproblem is not well-posed, we deal with these special cases in the following ways. If the vertices all lie in a lower-dimensional subspace, the optimal solution is to place the free vertex anywhere in this subspace, resulting in zero volumes for all the elements. In this case, the linear programming approach is not used to solve the subproblem in this sweep through the mesh. If the vertices are co-located in space, there is 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. The co-located vertex is not removed from the global mesh problem, just from the current local submesh.

To guarantee convergence of a well-posed local optimization problem, we proved that the level sets of the objective function given in equation (2) are convex and closed on the entire domain in both two and three dimensions [12]. Because 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 described above) is guaranteed to determine the global maximum for the local subproblem. This approach therefore converges to the global maximum of the local subproblem from any starting point for the free vertex.

In [12], we presented results for this mesh untangling algorithm for both two- and three-dimensional simplicial meshes. The algorithm was often able to create valid meshes in fewer than 10 sweeps through the mesh. Although the meshes were valid, the untangling procedure results in meshes of extremely poor quality—minimum angles of 10^{-3} degrees were typical. Therefore, we followed the mesh untangling procedure with three passes of combined Laplacian/optimization-based smoothing [10] to improve element quality. Our results showed severely tangled meshes required significantly more sweeps to untangle the mesh and decreased the effectiveness of the mesh smoothing passes. To reduce the costs of the mesh untangling procedure, we experimented with combining the optimization-based approach with Laplacian smoothing and found that these approaches resulted in higher quality meshes at lower computational cost than optimization-based untangling used alone. The best performing method used three sweeps of Laplacian smoothing followed by the optimization-based untangling technique. Based on these results, in the following section we consider the problem of extending this approach to the problem of untangling quadrilateral meshes.

In [12], we note that although the solution to each local subproblem is unique, the vertex positions following a sweep depend on the order vertices are processed and are not unique.

3. QUADRILATERAL MESH UNTANGLING

The criteria for a valid, untangled quadrilateral element is that the angles formed by the two edges incident to each quadrilateral vertex be less than 180 degrees. This criteria is equivalent to the quadrilateral element being convex. We do not allow quadrilateral elements with positive area but vertex angles greater than 180 degrees because, for linear finite-element basis functions, the Jacobian mapping of such an element to the canonical quadrilateral will be singular within the element. For a quadrilateral mesh to be valid, or untangled, each element in the mesh must be untangled.

We assume that any initial mesh has a valid boundary. That is, quadrilateral element boundary vertices (with both incident edges on domain boundaries) do not form an angle greater than 180 degrees. The algorithms we consider do not modify boundary vertices, only the geometric locations of the interior mesh vertices.

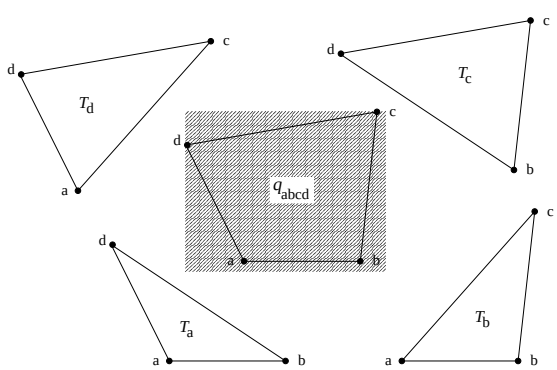

Figure 2. A valid quadrilateral element decomposed into four valid triangular elements

3.1 A triangle decomposition for quadrilaterals

The algorithms we present in this section are based on a construction that decomposes a quadrilateral element into triangles. For example, in Figure 2 we show how the quadrilateral q_{abcd} can be decomposed into four triangular elements T_{dab}, T_{abc}, T_{bcd}, and T_{cda}. We associate each of these triangles with the quadrilateral vertex that has two of the triangle's edges incident to it. Thus, we label the four triangles above as T_a, T_b, T_c, and T_d respectively.

The key observation is that the quadrilateral element is valid, or untangled, if and only if each triangle in this decomposition is valid. Thus, we can employ the same area metric used for untangling a simplicial mesh for quadrilaterals by appling it to each triangle in this decomposition. Note that this approach is easily extended to hexahedral elements—the hexahedral decomposition is to associate a tetrahedron with each of the eight vertices of the hexahedron.

Given an invalid quadrilateral mesh, the approach we employ is the same as that used with simplicial meshes. We sweep over the set of interior vertices, solving a local optimization problem based on the geometric location of each vertex. We note that changing the geometric location of one vertex of a quadrilateral changes only three of the triangles in the triangle decomposition detailed above—the area of the triangle associated with the vertex opposite the free vertex will not change as none of its three vertices are moved. Only the triangles associated with the free vertex and its two adjacent vertices in the quadrilateral are affected. For example, if vertex a in Figure 2 is the free vertex, triangles T_a, T_b, and T_d will be affected by a change in its location; T_c will remain unchanged. Given the quadrilateral q and a vertex in the quadrilateral v, we denote this three-vertex set as $V_q^3(v) = \{v, \mathrm{adj}_q(v)\}$.

3.2 Quality metrics for untangling quadrilaterals

Based on the triangle decomposition and the vertex sets defined above, we consider two quality metrics: the *three-triangle* metric and the *interior-triangle* metric. Given a vertex v in quadrilateral q and the areas, $A(T_{v'})$, of the triangles associated with any quadrilateral vertex, v', we define these metrics as follows

three-triangle metric (3T): This metric is given by the minimum area of the three triangles affected by the position $\mathbf{x}$ of the vertex v: $Q^3_{q,v}(\mathbf{x}) = \min_{v' \in V_q^3(v)} A(T_{v'})$.

interior-triangle metric (IT): This metric considers only the area of the interior triangle T_v: $Q^I_{q,v}(\mathbf{x}) = A(T_v)$.

We note an additional motivation for the use of these triangle areas as a metric for the quadrilateral. For the standard linear, isoparametric

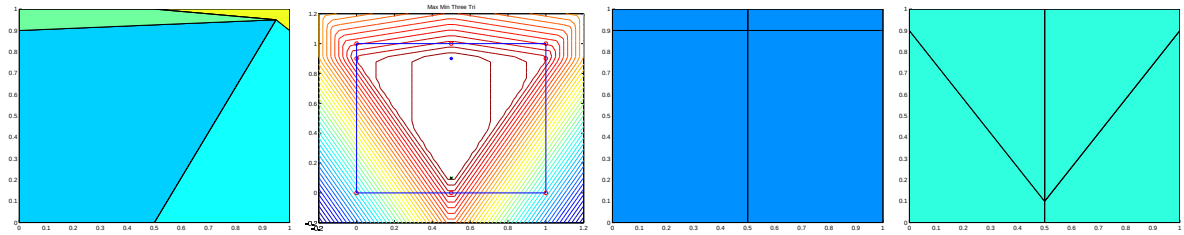

Figure 3. Example showing the nonuniqueness of the three-triangle method

finite element basis functions for the quadrilateral, we have that the determinant of the Jacobian (the mapping to the canonical quadrilateral element) evaluated at each element vertex is equal to the area of the triangle centered at that vertex. Clearly, in the evaluation of the finite element, is is important that the Jacobian remain positive, and if all these triangle areas are all positive then the Jacobian will be positive on the entire element.

Given one of these metrics, $Q_{q,v}$, the function that we wish to maximize for each interior vertex v contained by the n quadrilaterals $q_1, q_2, \ldots, q_n$ is

$$f_Q(\mathbf{x}) = \min_{1 \le i \le n} Q_{q_i,v}(\mathbf{x}). \tag{3}$$

Each of the triangle area functions is a linear function of $\mathbf{x}$. Therefore, as with the case of the metric for a simplicial element [12], this local subproblem can be posed as a linear programming problem for both the three-triangle and interior-triangle metrics. We again employ the simplex method for a robust means to solve these local optimization problems—the simplex method guarantees us a optimal value for the quality metric for a well-posed problem. Based on this approach for solving the local subproblems, the basic untangling algorithm for a quadrilateral mesh is the algorithm given in Figure 4.

While the mesh is invalid **do**
  For each interior vertex v **do**
    Solve the linear program for x^*
      which optimizes $f_Q(\mathbf{x})$;
    $x_v \leftarrow x^*$;
  enddo
enddo

Figure 4. A basic untangling algorithm

3.3 A tradeoff between the two quadrilateral metrics

Given a valid boundary, the three-triangle metric is positive for each interior vertex if and only if the quadrilateral mesh is valid. However, an untangling algorithm based solely on this metric has several drawbacks. The three-triangle method is not guaranteed to produce a unique vertex position given apparently well-posed initial quadrilaterals. In addition, the experimental results described in the next section demonstrate that the meshes generated using solely this metric are of poor quality.

First, we give an example demonstrating that the three-triangle method is not guaranteed to produce unique optimal vertex position. Note, however, that even though it is not unique, the optimal value for the metric is well defined from the linear programming formulation. Consider the example given in Figure 3. The initial mesh and level sets for the three-triangle objective function are shown in the two leftmost figures. The level sets, although convex everywhere, show a region where equivalent solutions can exist. Two such solutions are shown in the rightmost two figures; in both cases, the final minimum triangle area is .025 units. In this example, the two solutions are obtained by simply reversing the input order of the elements to the simplex method.

Our results show that the interior-triangle method generally works well in practice; however, it is not theoretically guaranteed to untangle a local submesh. Consider the simple example in Figure 5. The initial submesh contains an inverted, "bow-tie" quadrilateral with a non-convex angle. The solution of the interior-triangle method is shown in the central submesh. The vertex location that maximizes the minimum areas in the interior-triangle method falls outside the feasible region, and the quadrilateral with the non-convex angle remains an inverted "bow-tie" element. The solution given by the three-triangle method is shown in

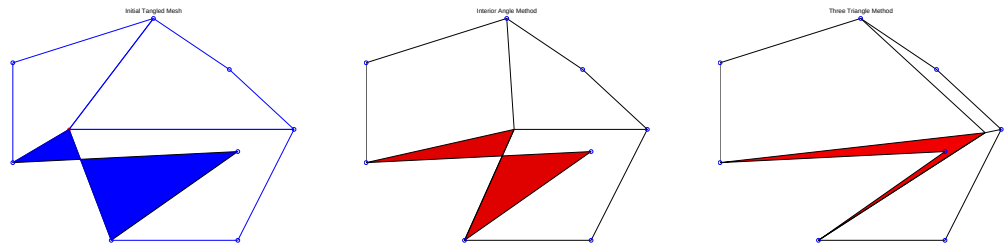

Figure 5. Example showing the tendency of the three-triangle method to create ill-conditioned meshes and the failure of the interior-triangle method to create a valid mesh

the rightmost submesh. The three angles affected by the vertex position in the invalid element are all now valid—the remaining invalid angle would have to fixed via the solution of the subproblems associated with the other three vertices. Note that the placement of the free vertex in this case is very close to one of the incident vertices which results in the remaining quadrilaterals being valid but of exceptionally poor quality.

In general, we have found that the three-triangle method tends to produce valid meshes with poor quality elements. Such elements can lead to ill-conditioning in the simplex method which in turn creates numerical difficulties in the solution process. If the vertices become co-located, they are randomly perturbed in an attempt to form a non degenerate problem. If the simplex method still fails to solve the problem, the free vertex location is unchanged in this sweep through the mesh.

3.4 A hybrid approach that combines the two metrics

It is evident that an algorithm based only on the three-triangle or interior-triangle metric is not guaranteed to be effective. As a result, we have developed a hybrid approach to try to take advantage of the relative strengths of the two metrics. As the interior-triangle metric appears to ultimately produce better quality meshes, we compute an initial step based on this metric. This step does not ensure that the modified quadrilaterals will be valid. Therefore, we use a hybrid method, denoted $IT - LS$, that starts with the step generated by the interior-triangle metric and, if necessary, performs a simple back-tracking procedure to find a point such that the three-triangle metric is positive. Specifically, if $\mathbf{x}^*$ is the new position given by the solution of the interior-triangle metric and the starting position of the vertex is $\mathbf{x}$, the hybrid step $\mathbf{x}_H$ is specified by a constant $\alpha, 0 \leq \alpha \leq 1$, by

$$\mathbf{x}_H = \mathbf{x} + \alpha(\mathbf{x}^* - \mathbf{x}). \tag{4}$$

The step length α is chosen using a line search procedure based on bisection that determines a point for which the the three-triangle quality metric ensures a valid mesh. Note that the search direction given by the interior-triangle method is not necessarily a descent direction with respect to the three-triangle metric. If it is not a descent direction, we return the vertex to its original location and do not update it during the current sweep through the mesh.

In the experimental section that follows, we use this hybrid step as the basis of two combined algorithms.

Combined algorithm 1 (CA1): If the three-triangle quality metric is negative for the vertex v, use the step computed from this metric. Otherwise, use the hybrid step.

Combined algorithm 2 (CA2): Always use the hybrid step; however, if the mesh is not improved using the hybrid step, return the vertex to its original location and use the step computed by the three-triangle metric.

Both of these combined algorithms ultimately rely on the three-triangle quality metric; therefore, the overall local mesh quality cannot decrease with the selected step. Thus, each local subproblem is guaranteed to converge when these combined methods are used.

4. NUMERICAL EXPERIMENTS

To analyze the effectiveness of our optimization-based untangling techniques, we devised a suite of

test cases based on the arc mesh shown in Figure 1. Although it is a simple geometry, the concave boundary in the arc and the regular structure of the mesh results in a difficult test case for mesh untangling routines [17]. Furthermore, experiments with both structured and unstructured mesh generation schemes in CUBIT [8] showed that test cases containing mapped, logically regular meshes are more difficult to untangle than unstructured meshes. We therefore focus our numerical experiments on this type of mesh.

Our experiments consist of three types of problems: randomly perturbed meshes, smoothly folded meshes such as the mesh shown in the right image of Figure 1, and meshes in geometries with sharp corners. In each case we design a suite of test cases consisting of increasingly difficult problems to highlight different aspects of the optimization algorithms. In all cases, we report the number of sweeps through the mesh as well as the computational time required required to untangle the mesh on a Linux i686 processor.

We use the following notation to denote each of the various algorithms studied.

- *Lap*: Laplacian smoothing
- *3T*: the three-triangle metric only
- *IT*: the interior-triangle metric only
- *IT-LS*: the hybrid step that uses the interior-triangle metric with the line search procedure
- *CA1*: combined algorithm 1 from Section 3
- *CA2*: combined algorithm 2 from Section 3
- *L(3)/CA2*: three passes of Laplacian smoothing followed by combined algorithm 2
- *L(10)/CA2*: ten passes of Laplacian smoothing followed by combined algorithm 2

4.1 Untangling Randomly Perturbed Interior Nodes

In our first set of experiments, we randomly perturb all interior vertices of the arc mesh shown in Figure 1 by a distance of H=1, 2, 4, and 8 times the average element edge length. The resulting initial meshes contain 256, 288, 301, and 312 inverted elements, respectively. In Table 1, we give results reporting the number of sweeps, S_U, the cost per vertex in seconds, τ_v, and the total time in seconds, τ_t, required to untangle the mesh for each of the algorithms described above. If the algorithm is unable to untangle the mesh in 500 sweeps through the mesh, we denote this by a line in Table 1. For six of the techniques, we show typical resulting meshes for the $H = 4$ case in Figure 6. We note that the untangling algorithms are designed to create *valid* quadrilateral meshes and have no motivation to create *high-quality* valid meshes. Further improvements in each mesh could be obtained by following the untangling procedure with one or more passes of optimization-based mesh improvement algorithms [20].

For all approaches, it is clear that as H increases, creating valid elements becomes more difficult and generally requires more sweeps through the mesh. Laplacian smoothing is able to untangle the mesh only if H is small and the overlap is not severe. For the $H = 2$, 4 and 8 cases, Laplacian smoothing fails to untangle the mesh. In these cases, elements are drawn below the concave boundary before the main body of the mesh can be successfully untangled. The most expensive approaches are those that depend heavily on the three-triangle metric: in particular 3T and CA1. The ill-conditioning caused by colocated vertices (see the leftmost mesh in the top row in Figure 6) causes slow and sporadic convergence as H increases. The interior-triangle metric, although it is not theoretically guaranteed to untangle the mesh is successful in all cases and results in fairly good quality meshes as is evidenced by the middle mesh in the top row of Figure 6. Adding the line search procedure to the interior-triangle metric significantly improves its performance for the most severely perturbed case. Furthermore, it does not impact the quality of the resulting mesh as shown in the rightmost mesh of the top row in Figure 6.

As expected Laplacian smoothing is the most computationally efficient method and is a factor of 6 and 8.5 faster on a cost per vertex basis than the IT and 3T methods, respectively. The 3T method is more expensive than the IT method due to the larger linear program solved at each vertex. The IT-LS cost per vertex falls between the IT and 3T metrics; it solves the smaller linear programming problem, but has additional costs associated with the line search procedure. The total cost, τ_t, is directly proportional to the number of iterations required to untangle the mesh, and the IT and IT-LS methods outperform the 3T method in all cases.

Both of the algorithms that strategically combine the 3T and IT-LS methods, CA1 and CA2, successfully untangle all test cases. CA1, which uses the 3T method more liberally, is subject to the same ill-conditioning problems as 3T used alone as

Table 1. The number of sweeps, S_U, cost per vertex, τ_v, and total cost, τ_t, to untangle the perturbed meshes

Method	$H=1$			$H=2$			$H=4$			$H=8$		
	S_U	τ_v (s)	τ_t (s)	S_U	τ_v (s)	τ_t (s)	S_U	τ_v (s)	τ_t (s)	S_U	τ_v (s)	τ_t (s)
Lap	3	1.13e-5	.011	–	–	–	–	–	–	–	–	–
3T	29	9.70e-5	.911	26	9.66e-5	.814	178	9.84e-5	5.67	109	9.83e-5	3.47
IT	3	7.05e-5	.068	8	7.08e-5	.184	14	7.12e-5	.323	29	7.17e-5	.674
IT-LS	5	7.91e-5	.128	6	7.82e-5	.152	15	7.81e-5	.379	16	7.86e-5	.407
CA1	18	8.11e-5	.480	26	8.17e-5	.698	38	8.21e-5	1.03	79	8.18e-5	2.13
CA2	6	7.94e-5	.176	8	7.95e-5	.237	14	8.04e-5	.421	18	8.12e-5	.543
L(3)/CA2	3	1.12e-5	.011	3+3	4.59e-5	.089	3+6	5.85e-5	.171	3+19	7.39e-5	.527
L(10)/CA2	3	1.15e-5	.011	10+2	2.32e-5	.091	10+3	2.72e-5	.115	10+6	3.70e-5	.193

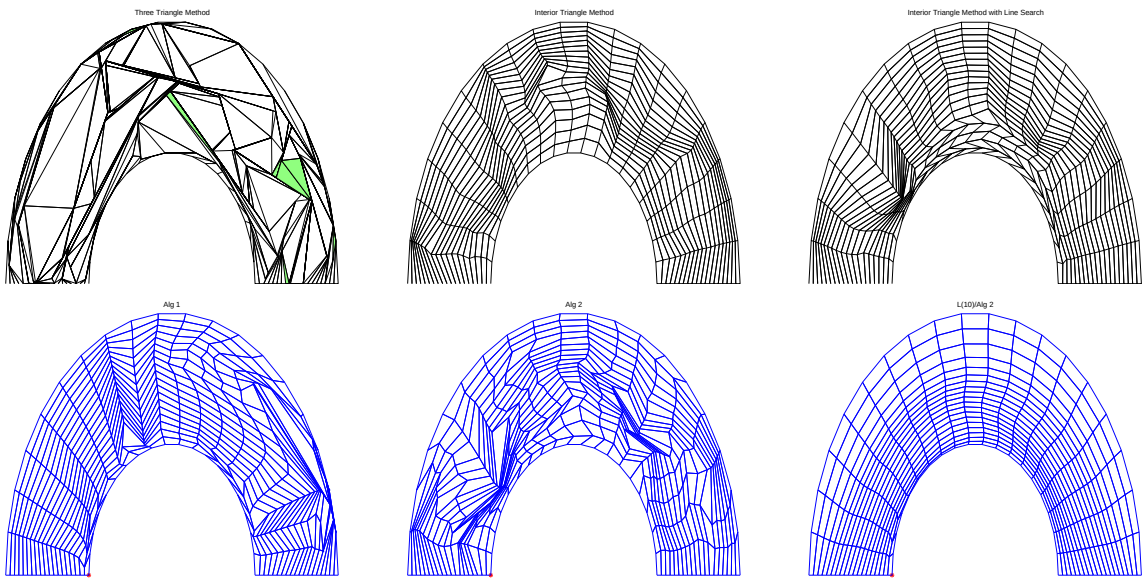

Figure 6. Untangled meshes for the $H=4$ case: top row left to right: 3T, IT, and IT-LS, bottom row left to right: CA1, CA2, and L(10)/CA2

is evident by the relatively high number of sweeps required to untangle the test cases. CA2, which uses the 3T method sparingly, is more consistent than CA1 and performs similarly to the IT-LS method used alone. The untangled meshes for CA1 and CA2 are shown in the two leftmost images in the bottom row of Figure 6. The tendency of the 3T method to colocate vertices is clearly seen in both cases. The cost per vertex in each of these methods falls between the IT-LS and 3T methods, with the heavier use of the 3T method evident in the slightly higher cost of the CA1 algorithm.

In the last two rows of Table 1, we show the results of preceding CA2 with three and ten sweeps of Laplacian smoothing, respectively. In all cases, Laplacian smoothing significantly reduces the number of optimization iterations, and consequently both the cost per vertex and total cost required to untangle the mesh. In particular, τ_v for the L(10)/CA2 method for the $H=4$ and $H=8$ cases is only 2.5 and 3 times more expensive, respectively, than Laplacian smoothing used alone. The mesh untangled with the L(10)/CA2 method is shown in the rightmost figure in the bottom row of Figure 6. This mesh clearly has the best quality, and Laplacian smoothing, although it can not be used alone, is highly recommended as a preconditioner for the optimization-based untangling methods.

4.2 Untangling a Smoothly Folded Mesh

Meshes that have been smoothly folded by applying Laplacian smoothing near convex boundaries or sharp corners are particularly challenging test cases for mesh untangling algorithms. In these cases, the inverted elements neighbor each other and the algorithm must work in an iterative process to pull the inverted elements back inside the concave boundary. For example, using Laplacian smoothing on the arc mesh shown in Figure 1 until the mesh has converged and no node movement is observed results in a smoothly folded mesh containing 103 inverted elements. We apply each of the five optimization algorithms to this test case and report the results in Table 2. As before we give the number of sweeps required to untangle the mesh, S_U, number of calls to the untangle routine, C_U, and total cost, τ_t, to untangle the folded mesh. We do not give the cost per call, τ_v as those values are essentially the same as those given in Table 1.

Table 2. The number of sweeps, S_U, number of calls to the untangle routine, C_U, and total cost, τ_t, to untangle the folded arc mesh

Method	S_U	C_U	τ_t (s)
3T	114	36936	3.43
IT	48	15552	1.10
IT-LS	69	22356	1.74
CA1	85	27540	2.23
CA2	67	23995	1.90

Each optimization algorithm can successfully untangle the folded mesh, although the number of sweeps required is typically significantly higher than was required for the randomly perturbed meshes. In particular, even for the most severely perturbed case which initially contains three times more inverted quadrilaterals, more iterations were required to untangle the folded mesh. Another interesting aspect of this test case is that in all cases the number of inverted elements is slightly increased during the initial untangling passes as more tangled elements are created near the corners of the folded region. Thus, allowing only monotonically increasing improvement in the elements during untangling would fail in this test case.

It is important to have a means to assess the limits of the untangling algorithms. To quantify this assessment, we have designed a test problem generator that creates problems that are increasingly difficult to untangle as a function of a parameter θ_R. The problems are formed by starting with a regular quadrilateral mesh for an annulus with inner radius 1.0 and outer radius 2.0. We then rotate the outer boundary by an angle θ_R. This rotation forces the quadrilaterals to become increasing skewed until, after a critical angle, it is impossible for all the quadrilaterals to be convex. This critical angle depends on the number of elements used in the radial and axial directions. An example of the initial mesh for $\theta_R = 45°$ is shown in the left of Figure 7.

Using these initial meshes, we solved two sequences of test problems using the combined algorithm CA2. The initial mesh in each sequence was generated by increasing the angle of rotation from 0° until it was no longer possible to untangle the mesh in less than 100 untangling passes. For each angle of rotation, we perturbed the mesh in two different ways prior to employing the untangling algorithm. For the first sequence, a random perturbation of length scale $H = 1$ was used followed by 10 passes of Laplacian smoothing. This results in a smoothly folded mesh, a typical example of which is shown in the center of Figure 7 for $\theta_R = 45°$. For the second sequence, a random perturbation of the vertices with a length scale $H = 1$ was used without any following Laplacian smoothing. An example of the resulting untangled meshes is shown in the right of Figure 7.

The results for these two test problem sequences are shown in Figure 8. The upper graph showns the number untangling passes required to untangle the perturbed meshes as a function of the angle of rotation θ_R. In the lower graph, the number of inverted elements in the initial perturbed meshes is shown. As our previously discussed results have indicated, the meshes that have been Laplacian smoothed provide consistently more difficult meshes to untangle, although the initial number of inverted elements is much fewer. What is surprising about these results is the fact that it is possible to untangle the annulus mesh with outer boundary rotated by an angle of greater than 120°. It is not apparent how to generate a mesh with this constraint with convex quadrilaterals a priori, without using the untangling algorithm.

4.3 Untangling Meshes in Geometry with Sharp Corners

To illustrate the effectiveness of the various methods in a geometry with sharp corners, our final test case sequence uses a mesh generated for the arc geometry from Figure 1 intersecting with a trapezoid to create sharp corners at the points of intersec-

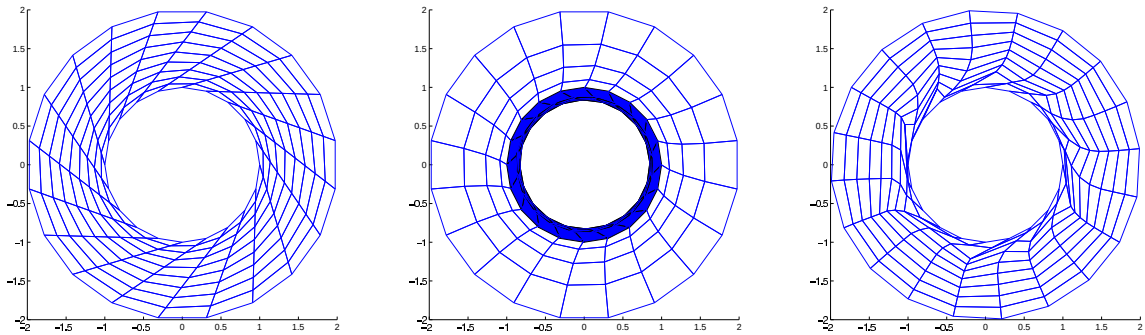

Figure 7. The initial rotated annulus mesh, the mesh with nodes perturbed with length scale $H = 1$ followed by ten passes of Laplacian smoothing, and the mesh untangled by the CA2 algorithm

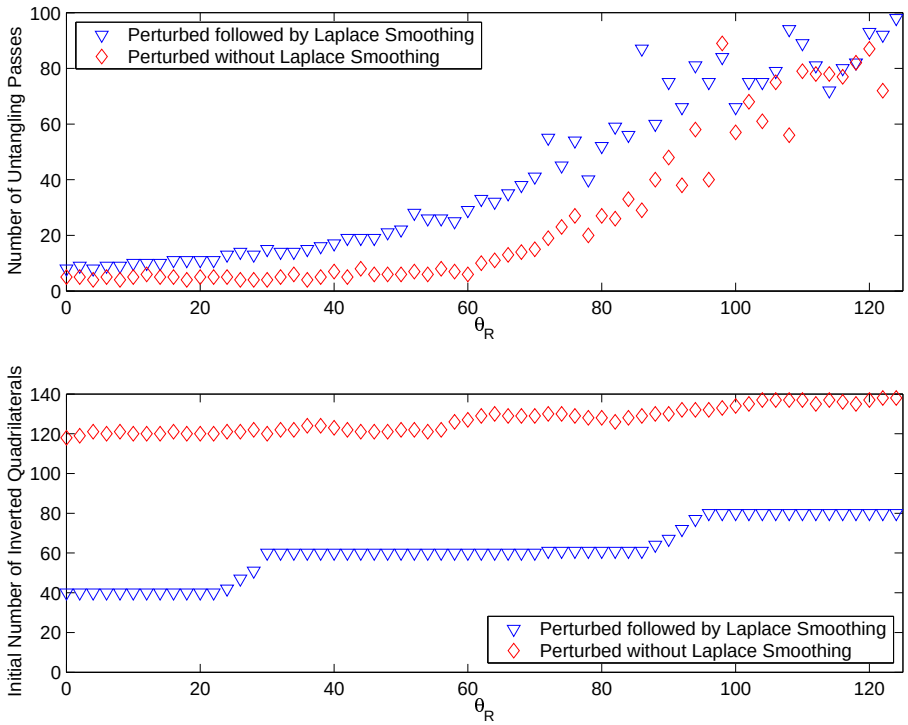

Figure 8. The number of smoothing passes required to untangle the rotated mesh using the CA2 algorithm as the angle of rotation increases (top); the number of initially inverted quadrilateral elements (bottom)

tion. This geometry and the initial mapped mesh are shown in the leftmost image of Figure 9. Using this mesh as our starting point, we create three test cases. First, we randomly perturb the mesh nodes a distance of $H = 1$ and apply the five algorithms directly to the perturbed mesh. Second, we randomly perturb the mesh as before and apply ten passes of Laplacian smoothing which creates 11 nonconvex quadrilaterals. Third, we randomly perturb the mesh as before and apply Laplacian smoothing until convergence which results in 14 nonconvex quadrilaterals located near the intersection corners; this mesh is shown in the center of Figure 9.

The results of this series of tests are given in Table 3. For this geometry only the 3T algorithm and the two combined algorithms, CA1 and CA2, can successfully untangle the mesh. The IT and IT-LS methods failed in all cases because these algorithms are unable to create a valid mesh near the sharp corners of the geometry as was illustrated in Figure 5. As before, preceding the optimization-based algorithms with a small number of Laplacian smoothing sweeps significantly reduces both the number of sweeps through the mesh and the

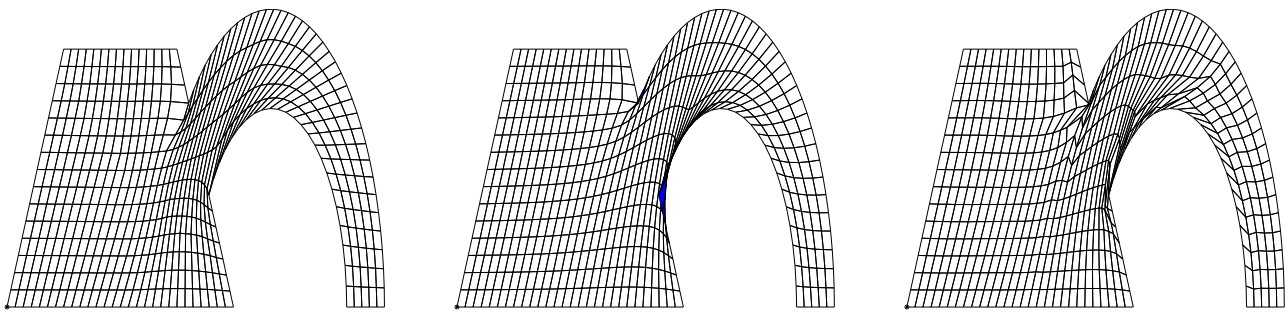

Figure 9. Example showing the arc geometry intersecting a trapezoid (left), the result of using Laplacian smoothing to convergence (middle), and the L(10)/CA2 untangling algorithm (right)

Table 3. The number of sweeps, S_U, number of calls to the untangle routine, C_U, and total cost, τ_t, to untangle the mesh with sharp corners

Method	S_U	C_U	τ_t (s)
H=1, No Lap			
3T	32	13920	1.39
CA1	14	6090	.515
CA2	5	2383	.200
H=1, Lap(10)			
3T	7	3045	.335
CA1	3	1305	.104
CA2	4	1781	.193
H=1, Lap Converged			
3T	6	2610	.235
CA1	29	12615	1.02
CA2	6	2678	.215

total time required to untangle the mesh compared to the optimization-based techniques used alone. The results of using the L(10)/CA2 algorithm to untangle the mesh are shown in the rightmost image of Figure 9. However, using Laplacian smoothing until convergence did not result in a further improvement in time required to untangle the mesh, and the CA1 algorithm was significantly penalized by the additional nonconvex quadrilaterals

5. CONCLUSIONS

When faced with invalid quadrilateral elements, algorithms that are designed for element quality optimization are not appropriate because the underlying objective functions are not convex. This problem was solved for simplicial element meshes by developing a metric for the local vertex subproblems that is convex for invalid elements and can be efficiently solved as a linear program. In this paper, we have extended this approach to quadrilateral meshes by introducing a triangular decomposition of quadrilateral elements into overlapping triangles that allows the use of similar, area-based metrics. We have introduced two metrics, a three-triangle and an interior-triangle metric, that have the convexity property. The resulting vertex subproblems based on these metrics can again be solved as a linear program. Thus, the solution to a well-posed subproblem is guaranteed with the use of the simplex method.

Our experimental results indicate that the three-triangle metric when used alone results in poorly conditioned quadrilaterals. On the other hand, the quadrilaterals produced with the interior-triangle are generally well-conditioned; however, there is no theoretical guarantee that they will be valid as was illustrated by our test case with sharp corners. Therefore, we have introduced a hybrid method that uses the step produced by the interior-triangle method and then employs a line search based on backtracking, using the three-triangle metric as an acceptance criteria. This approach is guaranteed improve the mesh validity while at the same time improving the overall mesh quality. Experimental results for sets of test problems confirm the success of the hybrid approach used in conjunction with the three-triangle method, and, in particular, we recommend the use of the CA2 algorithm preceded by a few passes of Laplacian smoothing to untangle quadrilateral meshes.

For future work we have noted that this same approach can be extended to hexahedral meshes—we plan to pursue work in this direction. Similar work for hexahedral mesh untangling has been pursued by Knupp [17], and we will compare the results obtained by the generalization of the approach de-

scribed in this paper to those obtained by using his method. Finally, of central importance is the question as to whether the solution of these local, convex subproblems leads to a convergent global method with a unique solution. The difficulty with this problem is the observation that although the local (single vertex) optimization problems are linear and convex, the global optimization problem can easily seen to be nonlinear and nonconvex. It would be interesting to resolve the issue of the global convergence of this class of algorithms.

ACKNOWLEDGMENTS

We acknowledge Steve Vavasis of Cornell University for noting the equality of the triangle areas and the Jacobian determinant in the linear finite-element representation and Patrick Knupp of Sandia National Laboratories for his insights regarding the difficulty of untangling smoothly folded meshes.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] N. A. Calvo and S. R. Idelsohn. All-hexahedral mesh smoothing with a node-based measure of quality. *International Journal for Numerical Methods in Engineering*, 50(8):1957–1967, 2001.

[5] 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.

[6] J. Dompierre, P. Labbe, F. Guibault, and R. Camerero. Proposal of benchmarks for 3d unstructured tetrahedral mesh optimization. In *Proceedings of the 7th International Meshing RoundTable'98*, pages 459–478, 1998.

[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] T. D. Blacker et. al. CUBIT mesh generation environment. Technical Report SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, May 1994.

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

[10] 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.

[11] Lori Freitag and Patrick Knupp. Tetrahedral element shape optimization via the jacobian determinant and condition number. In *Proceedings of the Eighth International Meshing Roundtable*, pages 247–258. Sandia National Laboratories, 1999.

[12] Lori Freitag and Paul Plassmann. Local optimization-based simplicial mesh untangling and improvement. *International Journal of Numerical Methods in Engineering*, 49:109–125, 2000.

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

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

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

[16] Patrick Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part I - A framework for surface mesh optimization. *International Journal of Numerical Methods in Engineering*, 48(3):401–420, 2000.

[17] Patrick Knupp. Hexahedral mesh untangling and algebraic mesh quality metrics.

In *Proceedings of the Ninth International Meshing Roundtable*, pages 173–183, New Orleans, Louisiana, 2000.

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

[19] 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.

[20] 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.

[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.