Engineering with Computers (2001) 17: 269–286

Engineering with Computers

Guaranteed-Quality Simplical Mesh Generation with Cell Size and Grading Control

C. Ollivier-Gooch[1] and C. Boivin[2]

Advanced Numerical Simulation Laboratory, Department of Mechanical Engineering, University of British Columbia, Vancouver, BC, Canada

Abstract. *Unstructured mesh quality, as measured geometrically, has long been known to influence solution accuracy and efficiency for finite-element and finite-volume simulations. Recent guaranteed-quality unstructured meshing algorithms are therefore welcome tools. However, these algorithms allow no explicit control over mesh resolution or grading. We define a geometric length scale, similar in principle to the local feature size, which allows automatic global control of mesh resolution and grading. We describe how to compute this length scale efficiently and modify Ruppert's two-dimensional and Shewchuk's three-dimensional meshing algorithms to produce meshes matching our length scale. We provide proofs of mesh quality, good grading, and size optimality for both two- and three-dimensions, and present examples, including comparison with existing schemes known to generate good-quality meshes.*

Keywords. Mesh grading; Mesh quality; Mesh resolution; Quality guarantees; Unstructured mesh generation

1. Introduction

Unstructured meshes are widely used in scientific computing because automatic unstructured meshing codes can readily generate meshes for arbitrarily complex geometries. In contrast, structured meshing for complex geometries often requires significant user input to decompose the domain into blocks. In the rush to embrace unstructured mesh technology, however, one can lose sight of important issues regarding mesh quality. While we do not yet have a satisfactory theory relating geometric mesh quality metrics to convergence and accuracy properties of discrete solutions to physical problems, there are some things that we can say. We know that the accuracy [1] and solution efficiency [2,3] of finite-element discretizations are impaired by the presence of dihedral angles near 0° or 180°. Recent results indicate that this problem extends to finite volume methods as well [3].

In this context, we would like to use meshing algorithms that provide some guarantee on the geometric quality of the meshes they produce. During the past decade, several researchers have proven results guaranteeing high quality in some geometric measure for triangular and tetrahedral meshes. In two dimensions, Ruppert's [4] and Chew's [5] algorithms are similar enough that Shewchuk was able to analyze the two within a unified framework [6]. Shewchuk also presented a generalization of Ruppert's algorithm to three dimensions. In both two and three dimensions, the ratio of the circumradius of a cell to its shortest edge length can be bounded above. The two-dimensional bound for Shewchuk's improvement to Ruppert's scheme[3] is equivalent to guaranteeing a minimum angle of 30°.

Shewchuk's three-dimensional bound is somewhat less strong, in that small and large dihedral angles are allowed in the form of sliver tetrahedra. Slivers are tetrahedra whose points lie nearly on a single circle. Because the four points are nearly coplanar, the circumradius of the tetrahedron is often quite reasonable, sometimes even smaller than the shortest edge length. Shewchuk speculates about the possibility of removing these slivers from the mesh by

Correspondence and offprint requests to: Professor C. Ollivier-Gooch, Advanced Numerical Simulation Laboratory, Department of Mechanical Engineering, University of British Columbia, 2324 Main Mall, Vancouver, BC V6T 1Z4, Canada. E-mail: cfog@mech.ubc.ca

[1] Supported in part by NSERC operating grant OPG194467.
[2] Supported in part by an NSERC post-graduate scholarship.

[3] Although it is not strictly correct to do so, we will hereafter refer to Shewchuk's improved version of Ruppert's two-dimensional scheme as Ruppert's scheme to avoid confusion with Shewchuk's three-dimensional scheme.

point insertion and shows experimental results, but gives no proofs. Recently, several papers (for example, [7]) have been published with algorithms for sliver removal, including quality proofs. However, the provable bounds are too small to be of practical interest.

The common failing of both Ruppert's and Shewchuk's schemes is the length scale of the generated meshes. Each scheme produces meshes for which the edge length is provably no smaller than a constant factor times the *local feature size*, which is a measure of how close together nearby surfaces lie. This resolution is inadequate for many problems, even stipulating that adaptive refinement will be used. As an example, consider the case of stress calculation for a square plate with a square hole. The mesh generated by Ruppert's scheme, shown in Fig. 1(a) is clearly inadequate, and might not even capture the solution well enough to indicate that adaptive refinement is necessary; this problem is often even worse for fluid mechanics problems, where significant solution gradients can occur well away from the boundaries.

In this paper, we present extensions to Ruppert's and Shewchuk's algorithms to allow users better control over initial cell size and grading while retaining the other desirable properties of these schemes. Figures 1(b) and 1(c) show meshes with finer resolution that were generated by using our improved scheme.

To provide the proper context for discussion of our improvements, we will first review Ruppert's triangulation algorithm (as improved by Shewchuk) and Shewchuk's tetrahedralization algorithm briefly in Section 2. We give a detailed description in Section 3 of the modifications we have made to provide flexibility in specifying cell size and grading rate. We state quality results for both two- and three-dimensional versions of our scheme in Section 4; the proofs are presented as appendices. In Section 5, we present examples comparing mesh size, quality, and generation time for the original and improved Ruppert/Shewchuk schemes with our previous meshing code.

2. Ruppert's and Shewchuk's Guaranteed-Quality Meshing Schemes

Both Ruppert's and Shewchuk's schemes are *Delaunay insertion schemes*, which means that each algorithm presumes a constrained Delaunay mesh as a starting point and provides instructions on how to choose locations at which to insert new points into the mesh to achieve good mesh quality. In each case, there are restrictions on the initial mesh – not just any Delaunay mesh will do. We describe those restrictions first, then discuss the refinement algorithms.

2.1. The Beginning: Input Restrictions

For provable mesh quality and size bounds to apply, Ruppert's scheme requires input geometries to have no angles smaller than 60°. Shewchuk's three-dimensional scheme requires, roughly, that incident segments in the input be separated by an angle of 60°; that incident facets be separated by 60°; and that a segment incident on a facet not containing that segment be separated from the facet by arccos $\frac{1}{2\sqrt{2}} \approx 69.3°$. In practice, smaller angles are acceptable, although small angles (typically, those smaller than the angle bound that one seeks to enforce) require special treatment to prevent infinite refinement; both Ruppert [4] and Shewchuk [6] provide suggestions for how to do this.

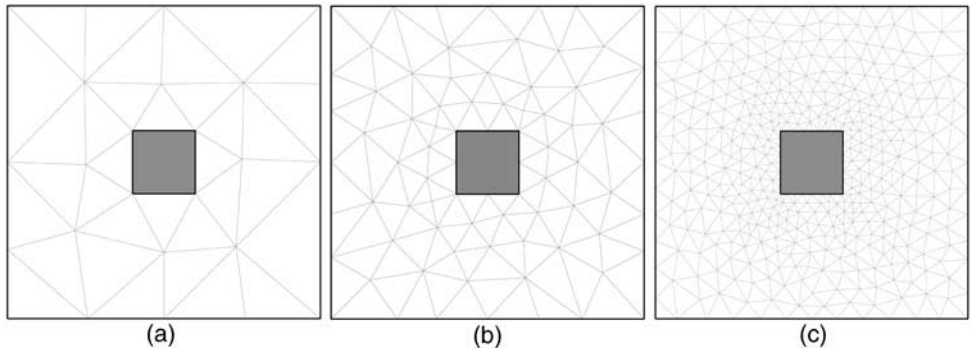


Fig. 1. Three meshes for a plate with a hole, using different mesh scaling parameters.

2.2. The Middle: Point Insertion Rules

Given an initial constrained Delaunay triangulation, both Ruppert's and Shewchuk's schemes proceed in the same general way. Each removes badly shaped cells by inserting points at their circumcenters. To avoid creating small features in the mesh, both schemes refuse to insert points that *encroach* into protective regions around boundary entities. Instead, points are inserted to split the encroached boundary entities themselves, with care taken to ensure that the triangulation remains constrained Delaunay. If the initial triangulation contains any encroached boundary entities, these are split before attempting to remove badly shaped cells.

2.2.1. Ruppert's Scheme

In Shewchuk's modification to Ruppert's scheme, a vertex is said to encroach on a boundary edge if it lies within the diametral lens shown in Fig. 2. This diametral lens consists of the region between two circular arcs passing through the endpoints of the edge, each with its center on the other arc.

Ruppert's two-dimensional scheme uses two simple insertion rules[4]:

1. If a triangle is badly shaped (i.e. it has an angle smaller than 30°), then a point is inserted at its circumcenter, *unless*
2. The new point encroaches on any visible boundary edges, in which case those edges are bisected instead. If this bisection point encroaches on any boundary edges, those edges must be split as well. When a boundary edge is split, all points in its diametral circle that are not boundary points are removed from the mesh.

Boundary edge splitting takes priority over bad cell removal.

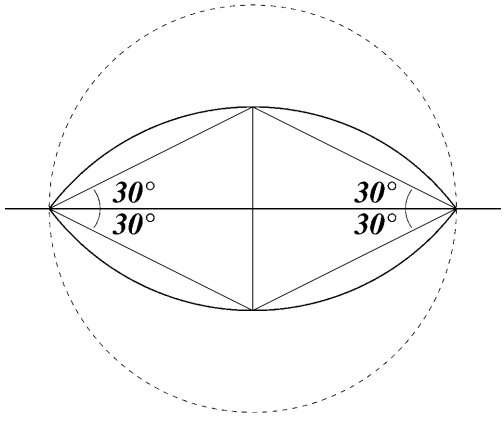


Fig. 2. A diametral lens around a boundary edge in two dimensions.

[4] We ignore here for clarity some technical details regarding handling of small input angles, etc, which are important for achieving the angle bound given in Section 2.3. Interested readers are encouraged to refer to Ruppert's and Shewchuk's work for more information.

2.2.2. Shewchuk's Scheme

Following Ruppert, Shewchuk defines a *boundary segment* as an edge present in the input geometry. If a boundary segment is divided into parts by subsequent point insertion, each part is called a *boundary subsegment*. Shewchuk also defines a *boundary facet* as a planar polygonal surface in the input, bounded by boundary segments. When a boundary facet is divided, either by triangulation or addition of points in its interior, the parts are referred to as *boundary subfacets*. A triangular boundary subfacet is encroached if a point lies within the *equatorial lens* of the triangle (shown in Fig. 3) formed by the intersection of two spheres passing through the vertices of the triangle, each with its center on the surface of the other sphere. A boundary subsegment is encroached if a point lies within the unique sphere that has the subsegment as its diameter.

We give below an outline of Shewchuk's scheme[5]:

1. If a tetrahedron is badly shaped – if its shortest edge is too small in comparison to its circumradius – then insert a point at its circumcenter UNLESS that point would encroach on any boundary subfacet or subsegment, in which case the cell circumcenter is not inserted. Instead,

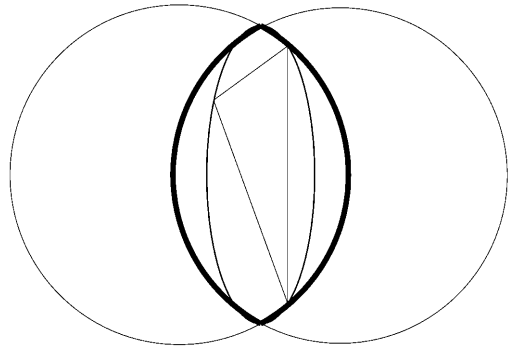


Fig. 3. Equatorial lens (bold arcs) formed by a triangular subfacet. Two-dimensional sections of the lens are shaped like the diametral lens around an edge.

[5] Again, we omit some details for clarity. Please see Shewchuk [6] for more information.

encroached boundary entities are split; if the original cell still exists after splitting boundary entities, then the cell is split.
2. If a vertex encroaches on a boundary subfacet *and* the normal projection of the vertex into the plane of the subfacet lies inside the subfacet, split the subfacet. Subfacets are split by inserting a point at their circumcenter UNLESS that point would encroach on any subsegment, in which case, the subfacet circumcenter is not inserted. Instead, all encroached subsegments are split. If the subfacet survives subsegment splitting, the subfacet is split afterwards. Before a subfacet is split, all points inside its equatorial sphere (not the equatorial lens, which is point-free) are deleted from the mesh.
3. If a boundary subsegment is encroached, it must be split. If the newly-inserted vertex encroaches on any other subsegments, those subsegments should be split as well. Care must be taken with handling of small input angles to prevent infinitely recursive insertion.

Subsegment splitting takes priority over subfacet splitting, which takes priority over bad cell removal.

2.3. The End: Mesh Quality Guarantees

Both Ruppert's and Shewchuk's schemes come with mesh quality guarantees. The guarantees for Shewchuk's modification to Ruppert's scheme are comfortably strong. The worst ratio of circumradius to shortest edge length can be limited to one, which implies a worst-case angle of 30°. If this bound is relaxed slightly to allow only a worst-case angle of $\arcsin(\sqrt{3}/4) \approx 25.7°$, the mesh is guaranteed to demonstrate good grading: the output mesh will not be uniformly fine. In practice, these bounds are not tight, and good grading can typically be obtained with smallest angles as large as 32°. Finally, Ruppert showed that the size of the output mesh will be within a constant factor of the size of the smallest possible mesh that satisfies the quality bound.

Shewchuk was able to prove a similar bound on geometric shape quality for tetrahedral meshes. For the version of the algorithm we describe, the best bound on the ratio of circumradius to shortest edge length for which Shewchuk was able to prove good grading is $\frac{2\sqrt{2}}{\sqrt{3}}$. Shewchuk also provides a counterexample showing that his scheme does not produce optimally-sized meshes compared with the smallest possible mesh that matches the input and satisfies the quality bound[6]. Although Shewchuk's scheme has an agreeably strong bound on mesh quality, it still allows some sliver tetrahedra, which are often problematic for solution of partial differential equations. Shewchuk demonstrates heuristically that Delaunay refinement can remove most of the worst tetrahedra from a mesh, although no proofs are available. He also speculates that mesh reconnection and smoothing are likely to be effective in removing such tetrahedra; our examples in Section 5 will demonstrate the effectiveness of these techniques.

3. Beyond Ruppert and Shewchuk: Controlled Cell Size and Grading

We seek better control over length scale without sacrificing guarantees of mesh quality, grading, and size optimality. We define a geometric length scale (in both two and three dimensions) based on the *local feature size lfs*(p), which is in turn defined as the radius of the smallest ball centered at that point that touches two disjoint parts of the domain boundary. We explicitly compute the local feature size for each input vertex, then define the geometric length scale $LS(p)$ in terms of this:

$$LS(p) = \frac{1}{R}\, lfs(p) \tag{1}$$

This gives control of resolution of input features, with approximately R length scales spanning any small feature. To achieve uniform resolution of geometric features over the entire domain, R is a constant; to allow user flexibility in determining appropriate resolution for a given problem, R is user-definable at run-time. Adaptive refinement will of course still be necessary for proper resolution of the physics of many problems.

In parallel with the properties of the local feature size, we require that the length scale grow no faster than linearly with distance. That is, for a new point p inserted into the mesh near a point q, then the length scale at p cannot exceed that at q by more than a fixed constant times the distance between p and q, regardless of the local feature size at p. For points inserted into the mesh, then, the length scale can be computed by using:

$$LS(p) = \min\left(\frac{lfs(p)}{R}, \min_{\text{neighbors}_{q_i}} LS(q_i) + \frac{1}{G}\left|\vec{q}_i - \vec{p}\right|\right) \tag{2}$$

[6] Shewchuk is silent on whether his algorithm produces meshes of optimal size if cell size comparable to the local feature size of the input is required.

Our length scale definition contains an α-Lipschitz function, with two parameters. One of these, R, controls the ratio of input boundary edge length to final mesh boundary edge length, with finer boundary discretization for larger values of R[7]. The other parameter, G, controls how rapidly cell size can change with distance. Larger values of G imply slower increase in mesh size leaving the vicinity of small geometric features in the mesh. Like R, G is a constant for the entire mesh. The local feature size can be recovered from our generalization by setting R and G both to unity.

We modify Ruppert's and Shewchuk's schemes to split cells that are too large in relationship to our length scale. The remainder of this section describes our approach in detail.

3.1. Setting an Initial Length Scale for Each Vertex

Ruppert's scheme does not actually use the local feature size directly, and so makes no provision to calculate it. We must calculate local feature size at the original vertices of the mesh to determine length scale there. To do this, we first construct a constrained triangulation of the input data. Then, for each vertex p, we list each of its vertex neighbors and all of the boundary entities (segments or facets) incident on one of p's neighbors, but not on p itself. Because these lists are constructed by using the mesh topology, segments invisible to vertex p are never entered in the list, which would be a problem with tree search approaches to finding nearby boundary entities. For each vertex and boundary entity, we find distance from p; the closest vertex or boundary entity determines the local feature size at p according to the definition.

3.2. Setting the Length Scale for New Vertices

For each newly-inserted vertex p, we compute the length scale $LS(p)$ by using the rigorous definition of Eq. (2). This requires that we know, for each vertex, the identity of its nearest and second-nearest boundary entities (those that determine its local feature size); accordingly, each vertex stores this information.

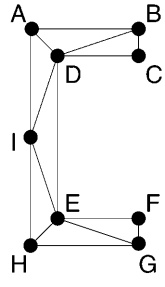
Initially, the mesh contains only boundary vertices, for which the length scale is known. A new vertex p is inserted. The immediate neighbors q_i of p are then checked. First, the maximum length scale allowed at p by grading is found (second half of Eq. (2)). Next, we construct a list of all boundary entities that determine the local feature size of any of the neighbors q_i.

If the new vertex is a boundary vertex, all boundary facets incident on cells incident on that vertex are also listed to prevent the problem illustrated in two dimensions in Fig. 4. In this instance, vertices A through H are input vertices, each with its length scale determined by the distance between itself and the nearest non-incident boundary entity. When vertex I is inserted, each neighbor is queried to find which entities determined its length scale. In this case, that list might include only A, AB, D, E, GH, and H – *not* including DE, which is clearly the entity required here. However, if all boundary facets and segments incident on cells incident on I (that is, cells AID, DIE, and EIH, which are incident on boundary segments DE and AH) are added to the list, then all is well.

Given all relevant boundary entities, the local feature size is easy to calculate from its definition. Then the definition of length (Eq. (2)) can be invoked. Once the correct length scale for a vertex is determined, this value is stored, along with the identities of the boundary entities defining the local feature size. This technique for computing the length scale for new vertices can be applied equally easily in two and three dimensions.

3.3. Deciding When to Divide a Cell

Now that the length scale is known at every vertex in the mesh, we can control the resolution of the mesh by changing the definition of a bad cell to take into account the size of the cell. We split any cell which is badly shaped, as before, but also split cells whose circumradius is too large compared with



Neighbor of I	Closest disjoint bdry entity
A	D (or CD)
D	AB (or AH)
E	GH (or AH)
H	E (or EF)

Fig. 4. Potential problem with length scale calculation.

[7] This implies that R should be one when meshing from well-resolved boundary discretizations.

the average of the length scale at their vertices. In two dimensions, we choose not to split right triangles with leg length equal to the length scale; this means that circumradii as large as $\frac{\sqrt{2}}{2}$ times the length scale are permissible. Likewise, in three dimensions we choose not to split right tetrahedra with all three edges meeting at the right corner having the same length as the length scale, implying a permissible circumradius of $\frac{\sqrt{3}}{2}$ times the length scale. This is the only change we make to the outline of Ruppert's and Shewchuk's algorithms.

3.4. Implementation of Encroachment Checking

A major implementation challenge with Shewchuk's scheme is determining whether a candidate vertex location encroaches on any nearby boundary entities. To address this, we choose to use Watson's scheme [8] for point insertion. We implement Watson's scheme in a two-step process. First, we gather information about which cells will be deleted from the mesh as a result of inserting a new vertex, as well as listing the existing faces of the mesh that must be connected to the new vertex. With this information in hand, we can identify encroached boundary faces, because any that exist are tagged for connection to the new vertex. Also, in three dimensions, we can identify any boundary subsegments on the hull and check them for encroachment, even though these subsegments may not have any incident boundary faces in the hull. With this approach, we can reject a new vertex that would encroach on the boundary before we modify mesh topology, avoiding the need for backtracking.

4. Quality Guarantees for the Modified Schemes

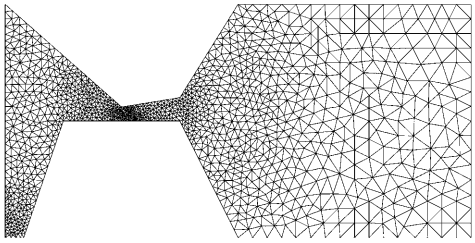
In both two-and three-dimensions, we have used an approach similar to Ruppert's [4] to show that our schemes produce good quality meshes of optimal size. In two dimensions, our results and the limitations placed on input geometry are the same as in Shewchuk's [6] extension of Ruppert's scheme. That is, we require that all input edges be separated by angles of at least 60°, and guarantee that the maximum angle in the output mesh will not exceed $\arcsin(\sqrt{3}/4) \approx 25.7°$, even with control over cell size and grading. We also obtain comparable guarantees on termination and mesh size optimality. The constants we obtain are more tightly bounded than Ruppert's for $G > 1$; this reflects fortuitously less slack in the analysis rather than any particular virtue of our scheme or of our analysis technique. The proof of these results may be found in Appendix A.

In three dimensions, our input restrictions are slightly different from Shewchuk's. In particular, our proofs require that incident facets be separated by dihedral angles of 60°; that a segment and facet sharing a vertex be separated by a dihedral angle of at least 60°; and that incident segments be separated by 90°. These conditions are sufficient to prove that the ratio of tetrahedron circumradius to shortest edge length may be bounded above by $\sqrt{2}$. At this limit, termination is guaranteed, but although the final mesh may be uniform if the ratio of circumradius to shortest edge length is precisely $\sqrt{2}$. This result is somewhat stronger than Shewchuk's result for comparable encroachment rules; the difference in provable bounds almost certainly is an artifact of input requirements or the method of proof rather than a substantive difference between the algorithms. In three dimensions as in two, we are able to show that our scheme produces meshes that are size optimal, in the sense that no other scheme can produce a mesh that has more than a constant factor fewer cells while still meeting the same shape quality bound *and* having edge lengths comparable to the length scale we use. The proof of these results may be found in the Appendices.

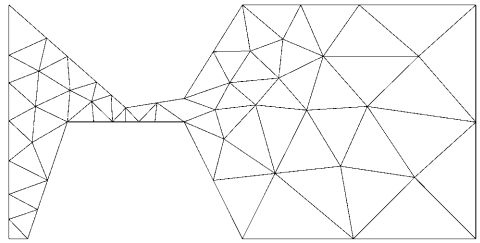
5. Examples

For both two and three dimensions, we present several example meshes. In each case, meshes generated using our new scheme are compared with meshes generated by the original Ruppert (2D) or Shewchuk (3D) scheme and with meshes generated by a reference mesher. Both reference codes have been described elsewhere [9], so we provide only a brief description.

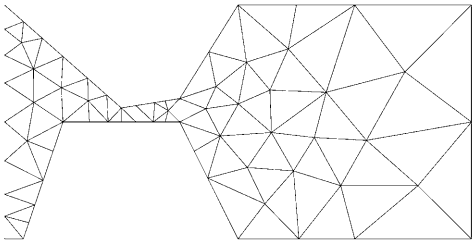
The two-dimensional reference code uses an algorithm similar to Bossen and Heckbert's pliant meshing algorithm [10]. This algorithm passes iteratively through the mesh, splitting edges that are too long, deleting vertices that are too close to others, and smoothing vertex locations. The scheme is guaranteed to produce meshes with a minimum angle of 30°, although for some cases the time required to reach this bound is prohibitive. The 2D



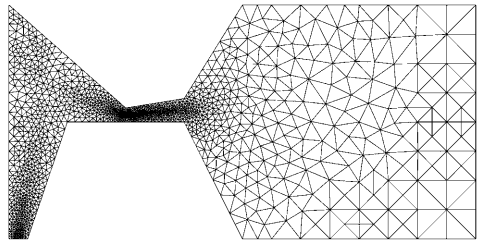
(a) Reference meshing code



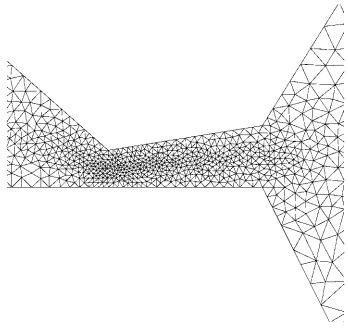
(b) Original Ruppert's scheme



(c) New scheme: R=G=1.



(d) New scheme: R=G=6.



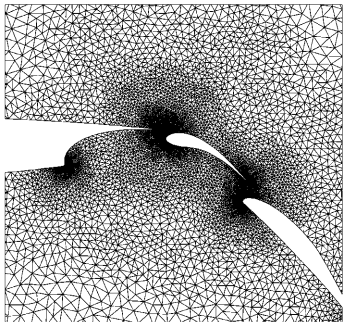
(e) Closeup of fine mesh (R=G=6)

Mesh	Cells	Rel. CPU per cell	Min ∠	Max ∠	Min AR	Relative edge length		
						Min	Mean	Max
Reference	2527	1.00	30.0°	109.2°	0.577	0.513	0.975	1.53
Ruppert	71	0.86	35.0°	98.7°	0.635	0.258	0.850	2.83
New, R=G=1	90	0.73	33.3°	105.5°	0.663	0.296	0.762	2.36
New, R=G=6	2876	0.47	34.0°	106.6°	0.645	0.399	0.931	1.67

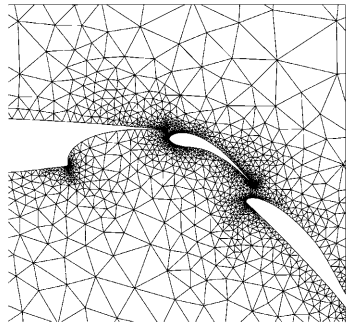
Fig. 5. Results for a two-dimensional test case.

reference mesher also provides reasonable control over mesh grading when a length scale function is provided. Because vertex smoothing is done on the fly, smoothing as a post-processing step does not improve meshes generated using this algorithm, and so no post-processing is performed.

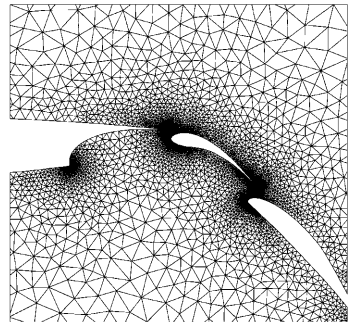
The three-dimensional reference code is a Delaunay refinement scheme that uses circumcenter insertion with simple heuristics to avoid inserting points too near boundaries. Again, control over mesh grading is possible by splitting cells that are too large. However, mesh quality is poor without extensive



(a) Reference mesher



(b) Ruppert's scheme



(c) Our scheme: R=1, G=6

Mesh	Cells	Rel. CPU per cell	Min ∠	Max ∠	Min AR	Relative edge length		
						Min	Mean	Max
Reference	41638	1.23	26.3°	113.9°	0.628	0.371	0.982	1.63
Ruppert	7548	0.77	30.2°	119.7°	0.621	0.122	0.637	26.8
New, R=1, G=6	34370	0.38	31.3°	117.3°	0.601	0.386	0.936	1.85

Fig. 6. Four-element airfoil test case.

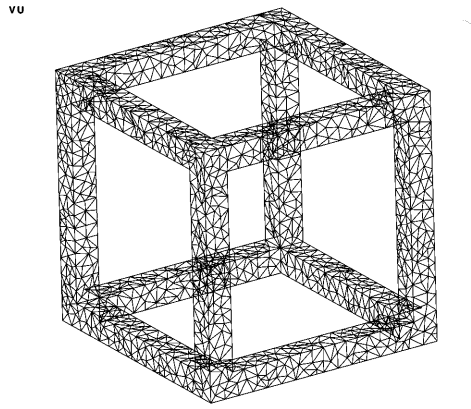
post-processing; we follow the recommendations in Freitag and Ollivier-Gooch [11]: first, we perform a pass of face- and edge-swapping to improve the minimum sine of dihedral angle, followed by two passes of optimization-based smoothing, then systematically work to remove remaining badly-shaped tetrahedra, and finish with two more passes of optimization-based smoothing. This procedure dramatically improves mesh quality for most cases, although at considerable expense in CPU time.

For Ruppert's and Shewchuk's schemes, and our improvements to those schemes, we post-process our meshes to improve the minimum sine of angles in the mesh (dihedral angles, in 3D). In the process, we must surrender (in 3D only) the strong bounds on circumradius to shortest edge length that Shewchuk's scheme guarantees. We consider this an excellent tradeoff, because numerical methods are sensitive to cells with poor dihedral angles, not those with small shortest edges. In three dimensions, we begin with local reconnection; in two dimensions, the Delaunay triangulation already maximizes the minimum sine of angles, so this step is unnecessary. In both two and three dimensions, we apply one pass of optimization-based smoothing, using the floating threshold approach described by Freitag and Ollivier-Gooch [11].

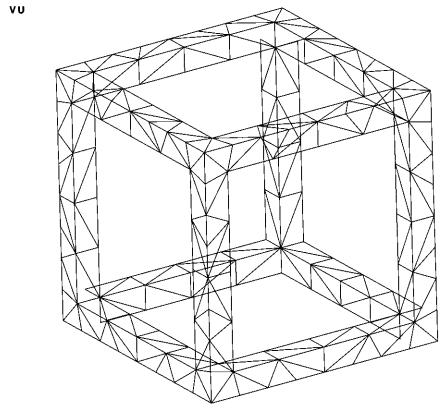
For these examples, we have used equatorial spheres to protect subfacets when using variants of Shewchuk's scheme in three dimensions, rather than the more complex choice of equatorial lenses. This difference affects the size of the generated mesh and the allowable angle between input facets, but not the provable bound of circumradius to shortest edge length.

5.1. Two-Dimensional Narrow-Gap Test

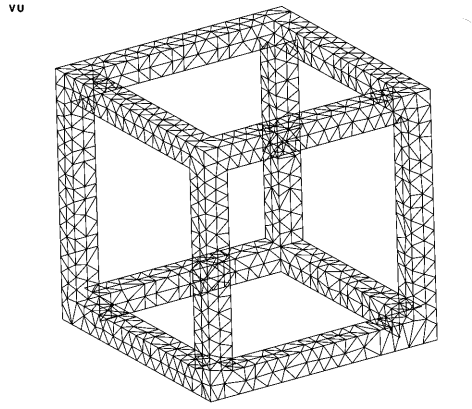
We begin with a geometry devised specifically to test length scale calculation, including handling of narrow gaps. Figure 5 shows meshes generated using the reference mesher, Shewchuk's modification to Ruppert's scheme, our new scheme with $R = G = 1$, and our new scheme with $R = G = 6$. Also included is a close-up of the last mesh in the 'neck' region. This close-up shows clearly that the length scale is smallest in the middle of the neck, where the local feature size is smallest. As the table indicates, each of the meshing schemes produces high quality meshes for this geometry, as measured either by angle or aspect ratio. In addition to angle quality, we compute relative edge length, as defined by edge length divided by length scale; this calculation is done twice for each edge, once for the length scale at each end point. A small relative edge length implies an edge that is much shorter than it should be. The schemes are similar in matching edge length to length scale. Note that Ruppert's scheme does well in this regard without explicit control of the length scale; our scheme does as well, or perhaps slightly better.



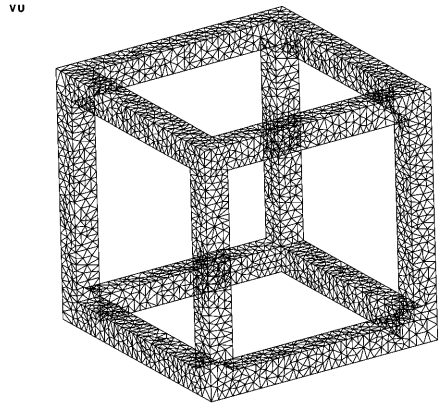
(a) Reference mesher



(b) Shewchuk's scheme



(c) New scheme, R=G=1.



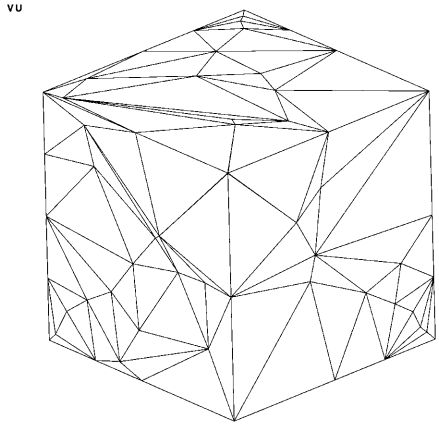
(d) New scheme, R=G=2.

Scheme	Cells	Rel. CPU per cell	Dihedral ∠		Relative edge length		
			Min	Max	Min	Mean	Max
Reference	13607	1.00	21.0°	159.0°	0.154	1.06	2.75
Shewchuk	384	1.07	26.6°	115.1°	0.884	1.59	2.45
New, $R = G = 1$	4189	0.64	26.2°	150.1°	0.221	1.01	2.56
New, $R = G = 2$	22945	0.48	25.3°	148.6°	0.333	1.10	2.86

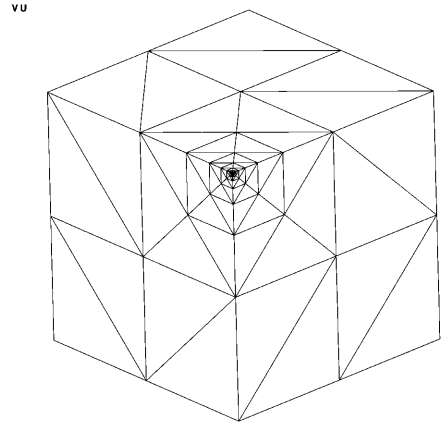
Fig. 7. Final meshes for cutout cube test case.

Our new scheme has an important advantage over both of the comparison schemes. Compared with the reference scheme, our scheme is about approximately twice as fast for meshes of non-trivial size[8], because our scheme (and Ruppert's) do not re-examine each vertex and face multiple times to determine whether the mesh is good enough locally. Even with careful marking of parts of the mesh as requiring no further improvement, the reference scheme is inefficient in this respect. Compared with Ruppert's scheme, our scheme is more flexible,

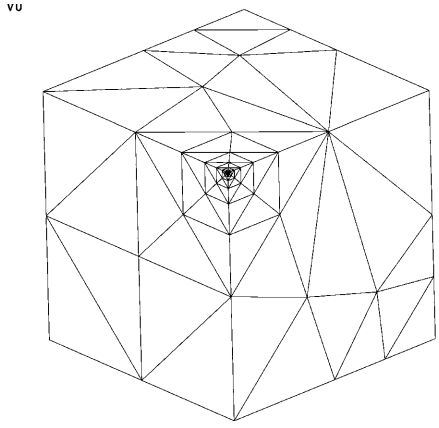
[8] The relative CPU time column in all tables in this paper shows relative time/cell generated, including post-processing and I/O.



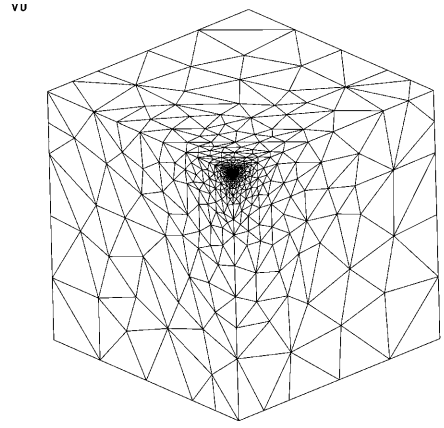
(a) Reference mesher



(b) Shewchuk's original scheme



(c) New scheme, with R=G=1



(d) New scheme, with R=1 and G=4

Scheme	Cells	Rel. CPU per cell	Dihedral ∠		Relative edge length		
			Min	Max	Min	Mean	Max
Reference	541	1.27	3.12°	168.9°	0.005	1.14	3.37
Shewchuk	134	0.68	26.5°	125.7°	0.44	1.03	3.19
New, G=1	216	0.57	29.0°	128.3°	0.25	0.89	1.80
New, G=4	3114	0.52	25.0°	145.2°	0.42	1.12	2.78

Fig. 8. Meshes generated inside a truncated cube.

allowing user control over cell size and grading without adversely affecting mesh quality.

5.2. Two-Dimensional Airfoil Test

As an example to demonstrate the impact of the grading parameter G, we present in Fig. 6 closeup views of meshes around a four-element airfoil. Once again, the meshes for the reference scheme and the variants of Ruppert's scheme give meshes of good and comparable geometric quality. With Ruppert's scheme, cell size varies quite rapidly away from the boundary. Whether this increase is *too* rapid depends upon the physical problem and dis-

cretization; we do not propose to dictate 'correct' grading rates based on any particular problem, but to allow flexibility to generate meshes appropriate for a range of problems. The mesh in Fig. 6(c) was generated using our scheme with $R = 1$ and $G = 6$ to approximately match the grading of the reference scheme; in this case, the speedup for the modified Ruppert scheme over the reference scheme is about a factor of 2.5 (as measured by CPU time/cell). This case demonstrates the same advantages of our new scheme as the previous case, while also highlighting the fact that Ruppert's scheme can occasionally allow edges that are significantly longer than desired, especially in parts of a mesh where the length scale increases rapidly.

5.3. Three-Dimensional Cutout Cube

In this example, we compare several meshes generated for a cube with much of its interior hollowed out, as shown in Fig. 7. These meshes were generated using our reference mesher, Shewchuk's original scheme, our new scheme with $R = G = 1$, and our new scheme with $R = G = 2$. This example shows the best-case performance for the reference scheme, producing a high-quality mesh for a case where the mesh is relatively fine and uniform. Mesh statistics for the final meshes are also given in Fig. 7. Both Shewchuk's scheme and our new scheme, in both variants, give somewhat better meshes than the reference scheme in terms of angle quality for this simple case while using the same or less CPU time per cell, including post-processing and I/O time. In all four cases, the relative edge lengths are pleasantly close to the ideal value of one. Our new scheme matches the required length scale well regardless of requested cell size and grading.

Table 1 shows the effect of the post-processing passes for each mesh. As Shewchuk speculates in his thesis, swapping and smoothing dramatically improve the quality of the meshes produced by his scheme; the same is true for our schemes, since they are based on Shewchuk's. The reference mesher sees significant benefit also, but only after special effort to remove badly-shaped tetrahedra during post-processing (see Freitag and Ollivier-Gooch [11] for details on this procedure). The reason that Shewchuk's scheme benefits so dramatically from post-processing is that the few badly-shaped cells it produces tend to be isolated, rather than falling in clusters. The second-to-last column in Table 1 shows the fraction of cells with dihedral angles less than 20° or larger that 150°, or solid angles smaller than 3°; Shewchuk's scheme and our new scheme produce fewer of these than the reference mesher. The last column in the table gives the fraction of faces of the badly-shaped cells that are not shared with a well-shaped tetrahedron. In this measure of isolation of bad cells, Shewchuk's scheme and ours are far superior to the reference scheme, which allows the mesh optimization routines more leeway to reconnect or relocate vertices advantageously.

5.4. Three-Dimensional Truncated Cube

As a final example, we present meshes generated in a cube that has had one corner cut off 0.1% of the

Table 1. Comparison of mesh improvement for cutout cube test case

Mesh	After . . .	Min ∠	Max ∠	Bad cells	Bad faces
Reference	insert	0.67°	178.4°	11.8%	28.1%
	swap	9.2°	164.6°	3.9%	37.9%
	tet fix	19.4°	160.6°	0.85%	26.9%
	smooth	20.0°	160.0°		
Shewchuk	insert	13.6°	160.5°	5.7%	(none)
	swap	26.6°	115.1°	(none)	(none)
	smooth	26.6°	115.1°		
New, $R = G = 1$	insert	0.92°	178.4°	3.9%	0.29%
	swap	15.2°	156.9°	0.05%	(none)
	smooth	26.2°	150.1°		
New, $R = G = 2$	insert	1.1°	178.0°	2.8%	3.7%
	swap	15.9°	155.7°	0.03%	(none)
	smooth	25.3°	148.6°		

way along each edge of the cube. This case is more challenging, because the geometric length scale of the boundary varies by three orders of magnitude. Here we compare the reference mesher, Shewchuk's scheme, our new scheme with $R = G = 1$, and our new scheme with $R = 1$ and $G = 4$, showing the effects of grading. The meshes are shown in Fig. 8, with the clipped corner pointing forwards; statistics are also given. The reference scheme, which refines only with the intent of matching a cell size distribution, produces quite a poor mesh for this case, both visually and statistically; from the picture, it is impossible even to tell that the clipped corner faces forward. In this case, the smallest dihedral angle was 1.7° after point insertion, but – with 27% of cells categorized as bad and fully 50% of the faces of bad cells shared with other bad cells – post-processing is ineffective. In contrast, Shewchuk's scheme and ours produce excellent meshes. The scheme with slower grading ($G = 4$) has many more cells, especially far from the clipped corner, as expected. In all four cases, the average relative edge length is again near one, as it should be, but the reference scheme has a very small minimum, indicating a small feature in the mesh. Finally, note that for this case, as for the cutout cube, our new scheme is about twice as fast per cell as the reference scheme for all but the smallest meshes. Most of this time difference is that our scheme requires less post-processing; some may also be due to differences in execution time between Watson insertion (in our new scheme) and Lawson insertion (insert-then-swap, in the reference scheme).

6. Conclusions

We have described improvements to Ruppert's two-dimensional scheme and Shewchuk's three-dimensional scheme that allow explicit control over mesh resolution and grading. Our new schemes combine flexibility in setting cell size and grading; generation of quality meshes; and efficiency.

We control cell size by defining a local length scale that has the properties that it is bounded above by the local feature size divided by a resolution parameter and that the length scale can not increase too fast, in the sense that the difference in length scale between two points cannot exceed the distance between them divided by a grading parameter. Appropriate values of the resolution and grading parameters will certainly vary from problem to problem, but we assume them to be constant within a given mesh. Because we split cells with circumradius larger than $\sqrt{D}/2$ times the length scale, our approach produces meshes with uniform resolution of geometric features.

In both two- and three-dimensions, we have proven quality guarantees for our new scheme. In two dimensions, we match the quality guarantees for Shewchuk's improvement to Ruppert's scheme: all angles in the final mesh are guaranteed to be larger than $\arcsin(\sqrt{3}/4)$; in practice, we can routinely achieve a minimum angle in excess of 30°. If the angle bound is not strict, then we can show that the final mesh is graded, i.e. that the mesh is not uniform. In three dimensions, we have shown that, for input segments and facets separated by 60°, we can achieve a bound of $\sqrt{2}$ on the ratio of circumradius to shortest edge length. If this ratio is strictly greater than $\sqrt{2}$, then we can also show that the resulting mesh is not uniform. Our bounds differ slightly from Shewchuk's, as do our requirements on input. These differences are likely related more to differences in proof technique that to the merits of the underlying schemes.

We have presented examples illustrating the behavior of our new schemes and comparing them with Ruppert's and Shewchuk's scheme, and with existing reference schemes. Compared to the reference schemes, our new schemes are significantly faster and produce better meshes. Compared to Ruppert's and Shewchuk's schemes, mesh quality and generation speed are comparable, but our scheme provides good, explicit control on cell size and grading.

Finally, we have demonstrated that the few poorly-shaped cells that Shewchuk's scheme and its derivatives produce can be effectively removed by local reconnection and smoothing. We also have analyzed the bad cells present in these meshes before and after post-processing, and conclude that Shewchuk's scheme benefits so dramatically from post-processing because the poor cells it does produce tend to be separated from each other by good-quality cells, allowing more freedom in reconnection and smoothing.

References

1. Babuska, I., Aziz, A. (1976) On the angle condition in the finite element method. SIAM J. Numer. Anal. 13, 214–226
2. Fried, I. (1972) Condition of finite element matrices generated from nonuniform meshes. AIAA J, 10, 219–221
3. Freitag, L. A., Ollivier-Gooch, C. F. (2000) A cost/benefit analysis of simplicial mesh improvement tech-

niques as measured by solution efficiency. Int. J. Computational Geometry, 10(4), 361–382
4. Ruppert, J. (1995) A Delaunay refinement algorithm for quality 2-dimensional mesh generation. J. Algorithms, 18(3), 548–585
5. Chew, L. P. (1993) Guaranteed-quality mesh generation for curved surfaces. Proceedings Ninth Annual Symposium on Computational Geometry, 274–280
6. Shewchuk, J. R. (1997) Delaunay Refinement Mesh Generation. PhD thesis, School of Computer Science, Carnegie Mellon University
7. Cheng, S.-W., Dey, T. K., Edelsbrunner, H., Facello, M. A., Teng, S.-H. (1999) Sliver exudation. Symposium on Computational Geometry, 1–13
8. Watson, D. F. (1981) Computing the *n*-dimensional Delaunay tessellation with application to Voronoi polytopes. Computer J. 24(2), 167–172
9. Ollivier-Gooch, C. F. (1998) An unstructured mesh improvement toolkit with application to mesh improvement, generation and (de-)refinement. AIAA 98–0218. AIAA 36th Aerospace Sciences Meeting and Exhibit, Reno, NV
10. Bossen, F. J., Heckbert, P. S. (1996). A pliant method for anisotropic mesh generation. Proceedings Fifth International Meshing Roundtable, 63–74
11. Freitag, L. A., Ollivier-Gooch, C. F. (1997) Tetrahedral mesh improvement using swapping and smoothing. Int. J. Numer. Meth. Eng. 40, 3979–4002

Appendix A: Proof of Mesh Quality in Two Dimensions

The proofs presented in this appendix follow closely the approach used by Ruppert [4]. We assume that boundary edges are protected by diametral lenses and that input angles are greater than 60°, to prevent adjacent edges from encroaching on each other. We will be able to show the same bounds that Shewchuk obtained for this encroachment definition. However, the constants in the proof are tighter for grading factor $G > 1$ (slower changes in cell size).

We now prove the following lemma, which, among other results, will establish an angle bound for finite cell size and hence for algorithm termination. This lemma is deliberately stated in language as similar as possible to Ruppert's Lemma 2, even to exact quotation of much of the phrasing, although details of the proof and the derived constants differ.

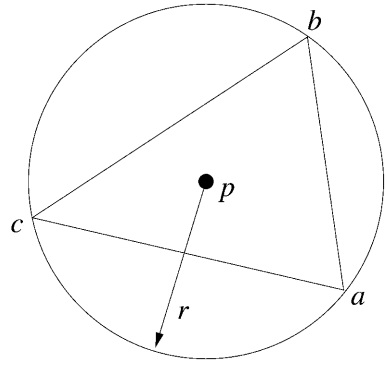


Fig. 9. Lemma 1, Statement 2: p added as circumcenter of large triangle T.

Lemma 1 (after Ruppert [4]). *For fixed constants C_L, C_T and C_S, determined below, the following statements hold*:

1. *At initialization, for each input vertex p, the distance to its nearest neighbor vertex is at least $lfs(p) \equiv R \cdot LS(p)$.*
2. *When a point p is chosen as the circumcenter of an overly-large triangle, the distance to the nearest vertex is at least $LS(p)/C_L$.* (p may be added to the triangulation, or may be rejected because it encroaches upon some segment.)
3. *When a point p is chosen as the circumcenter of a skinny triangle, the distance to the nearest vertex is at least $LS(p)/C_T$.* (Again, p may be added to the triangulation, or may be rejected because it encroaches upon some segment.)
4. *When a vertex p is added as the midpoint of a split segment, the distance to its nearest neighbor vertex is at least $LS(p)/C_s$.*

Proof. *Case 1.* Statement 1 of Lemma 1 is true by the definition of the length scale *LS* from the local feature size *lfs*, provided only that the constant R in Eq. (1) is ≥ 1.

Having established the truth of Lemma 1 for the initial mesh, we will proceed by induction to prove that it must be true for all meshes generated by the algorithm. As such, we will assume that Lemma 1 holds for all points in the mesh and determine the bounds on C_S, C_T, C_L, and G that are required for Lemma 1 to hold for newly inserted points.

Case 2. We consider first the case of insertion to split a large triangle, as shown in Fig. 9. By definition, the circumradius of $\triangle abc$ is larger than $\frac{\sqrt{2}}{2}$ times the average of the length scales at its vertices. The distance from p to the nearest point is the circumradius of $\triangle abc$, or

$$r \geq \frac{\sqrt{2}}{2} \frac{LS(a) + LS(b) + LS(c)}{3} \tag{3}$$

We can bound the length scale at p in terms of this same average using Eq. (2)

$$\begin{aligned} LS(p) &\leq LS(a) + \frac{r}{G} \\ LS(p) &\leq LS(b) + \frac{r}{G} \\ LS(p) &\leq LS(c) + \frac{r}{G} \\ LS(p) &\leq \frac{LS(a) + LS(b) + LS(c)}{3} + \frac{r}{G} \\ \frac{LS(a) + LS(b) + LS(c)}{3} &\geq LS(p) - \frac{r}{G} \end{aligned} \tag{4}$$

Combining inequalities (3) and (4) we get

$$r \geq \frac{\sqrt{2}LS(p)}{2} - \frac{r\sqrt{2}}{2G}$$

$$r \geq \frac{LS(p)}{\sqrt{2} + \frac{1}{G}} \tag{5}$$

This inequality places a lower bound on point spacing for points inserted to split large triangles, and confirms Statement 2 of Lemma 1, for any

$$C_L \geq \sqrt{2} + \frac{1}{G} \tag{6}$$

The lower bound on C_L becomes smaller for large values of G, corresponding to slow change in cell size.

Case 3. We consider next the case of insertion to split a badly shaped triangle, as illustrated in Fig. 10. Without loss of generality, we can label vertices so that a and b are connected by the shortest edge (of length $l_{\min}$), and a was inserted in the mesh after b (or both were input vertices). The radius of the vertex-free circle around a is r'. Four subcases for relating r' to $LS(p)$ arise, depending on why a was inserted in the mesh.

Subcase 3a: *a was an input vertex*. Then so was b, Statement 1 of Lemma 1 applies, and the distance $l_{\min} \geq R \cdot LS\ (a)$.
Subcase 3b: *a was inserted to split a large triangle*. The circumradius r' of that triangle is no larger than $l_{\min}$, because vertex b was not inside the circumcircle. Then Statement 2 of Lemma 1 applies, and $l_{\min} \geq r' \geq LS(a)/C_L$.
Subcase 3c: *a was inserted to split a badly shaped triangle*. By a similar argument and using Statement 3 of Lemma 1, and $l_{\min} \geq r' \geq LS(a)/C_T$.
Subcase 3d: *a was inserted to split an encroached boundary edge*. We know that b does not lie inside the diametral lens of the edge a split, because otherwise b would encroach on that edge. Statement 4 of Lemma 1 applies, and $l_{\min} \geq r' \geq LS(a)/C_S$.

If we satisfy $C_S \geq C_T = C_L \geq 1$ (which we will show is possible), then the inequality $l_{\min} \geq LS(a)/C_S$ (subcase 3d) causes the most difficulty in satisfying Statement 3 of Lemma 1 by making the length scale at p larger than that for any other subcase.

We can relate the radius r of the circumcircle of $\triangle abc$ to its smallest angle. The angle $apb = 2\theta$ by geometry, and trigonometry gives $l_{\min} = 2r \sin\theta$. The definition of length scale gives

$$LS(p) \leq LS(a) + \frac{r}{G} \leq C_S l_{\min} + \frac{r}{G}$$
$$= 2rC_S \sin\theta + \frac{r}{G}$$

This triangle is being split because θ is less than the required angle bound α. We can strengthen inequality (7) by replacing θ with α, and obtain

$$r \geq \frac{LS(p)}{\frac{1}{G} + 2C_S \sin\alpha}$$

Lemma 1 states that, for this case, $r \geq LS(p)/C_T$, so we require that

$$C_T \geq \frac{1}{G} + 2C_S \sin\alpha \tag{8}$$

Case 4. Now we turn our attention to the case in which vertex p is added to the mesh to split a segment s, because some vertex or triangle circumcenter lies inside the diametral lens of s. This case is illustrated in Fig. 11. There are three subcases:

Subcase 4a: *a lies on a segment t*, which can not share a vertex with s, because we have assumed that input edges are separated by 60°. Therefore, p and a lie on non-adjacent segments, and the length scale at p is $LS(p) \leq \frac{1}{R}|\vec{a} - \vec{p}|$. To satisfy Lemma 1 in this subcase, we therefore require that $C_S \geq \frac{1}{R}$. Because $R \geq 1$, this inequality is always satisfied for $C_S \geq 1$.
Subcase 4b: *a is a point at the circumcenter of a large triangle T*. a has of course been rejected for insertion, and all points inside the diametral circle of $\overline{bc}$ have been removed, so p has no vertex remaining in the mesh closer than a distance d. The circumradius r' of T is smaller than the shorter of $|\vec{a} - \vec{b}|$ and $|\vec{a} - \vec{c}|$, because T's circumcircle must be point-free. The worst case places a at the apex of the diametral lens, so $r' \leq \frac{2}{\sqrt{3}}d$. Also, we know from this lemma that $LS\ (a)/C_L \leq r'$. The definition

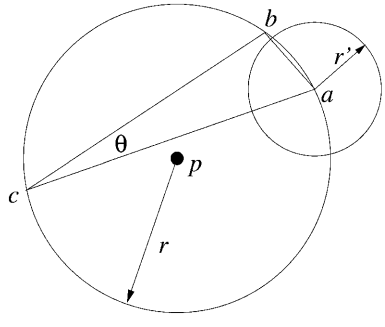


Fig. 10. Lemma 1, Statement 3: p added as circumcenter of badly shaped triangle T.

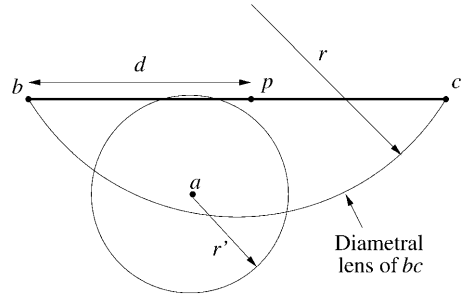


Fig. 11. Lemma 1, Statement 4: p added to split an encroached boundary edge. satisfied for $C_S \geq 1$.

of the length scale gives (for the worst case, where a approaches b or c)[9]

$$\begin{aligned} LS(p) &\le LS(a) + \frac{1}{G} d \\ &\le r' C_L + \frac{1}{G} d \\ &\le d\left(\frac{2}{\sqrt{3}} C_L + \frac{1}{G}\right) \end{aligned}$$

This inequality satisfies Lemma 1 provided that

$$C_S \ge \frac{1}{G} + \frac{2}{\sqrt{3}} C_L \tag{9}$$

Subcase 4c: *a is a point at the circumcenter of a skinny triangle*. The same reasoning can be applied as in the previous subcase, with the result that

$$C_S \ge \frac{1}{G} + \frac{2}{\sqrt{3}} C_T \tag{10}$$

To establish the truth of Lemma 1, we must find values of C_S, C_T, and C_L, that simultaneously satisfy Inequalities 6, 8, 9, and 10. We will establish tight bounds on each constant by requiring equality in each case. Using only 8, 9, and 10 we find that

$$C_S = \frac{1}{G} \frac{\sqrt{3} + 2}{\sqrt{3} - 4 \sin\alpha}$$

$$C_T = C_L = \frac{1}{G} \frac{\sqrt{3} + 2\sqrt{3} \sin\alpha}{\sqrt{3} - 4 \sin\alpha}$$

These values are bounded for any angle bound $\alpha \le \arcsin\left(\frac{\sqrt{3}}{4}\right)$, just as for Shewchuk's modification to Ruppert's scheme. The constants are nearly identical to Ruppert's, with the addition of a constant factor of $\frac{1}{G}$; this implies a tighter bound on shortest segment length in comparison to the length scale than Ruppert was able to show for the local feature size.

We can treat Equation 6 as establishing a lower bound on the grading rate $G \ge \frac{1}{C_L - \sqrt{2}}$. So long as G is finite, the mesh will be non-uniform. Relating G to the angle bound α,

$$G \ge \frac{1}{\sqrt{2}} \frac{\sin\alpha\ (2 + \sqrt{3})}{\sqrt{3} - 4 \sin\alpha}$$

The minimum theoretical grading rate G remains finite up to the previously established angle bound. If the lower bound for G is used, we have

$$C_S = \frac{\sqrt{2}}{\sin\alpha}$$

$$C_T = C_L = \frac{\sqrt{6}(1 + 2 \sin\alpha)}{\sin\alpha\ (2 + \sqrt{3})}$$

In summary, we have established that, for $R \ge 1$, we can generate meshes with the same angle bounds as Shewchuk's modification to Ruppert's scheme. In the process, we have placed bounds on the grading rate G and on the length of the shortest edge in the mesh relative to the local length scale (the constants C_S, C_T, and C_L give this information).

Appendix B: Proof of Mesh Quality in Three Dimensions

The proofs presented in this appendix follow closely the approach used by Ruppert [4], although more cases must be handled in three dimensions. We assume:

- Boundary subfacets are protected by equatorial lenses.
- When a boundary subfacet is split, any vertex on or inside its equatorial sphere is deleted from the mesh.
- Angles between facets are greater than 60° to prevent adjacent facets from encroaching on each other. This restriction only holds for pairs of facets that are visible to each other and for which one facet projects onto the other.[10]
- Any segment which projects onto a visible facet must be separated from it by a dihedral angle of at least 60°.
- Boundary segments are protected by diametral spheres, and must be separated from each other by 90°.

We now prove the following lemma, which among other results will establish a bound B on the ratio of tetrahedron circumradius to shortest edge length for finite cell size and hence for algorithm termination. This lemma is also deliberately stated in language as similar as possible to Ruppert's.

Lemma 2. *For fixed constants C_L, C_T, C_F and C_S, determined below, the following statements hold:*

1. *At initialization, for each input vertex p, the distance to its nearest neighbor vertex is at least lfs (p) = R · LS (p).*
2. *When a point p is chosen as the circumcenter of an overly-large tetrahedron, the distance to the nearest vertex is at least LS(p)/C_L. (p may be added to the tetrahedralization, or may be rejected because it encroaches upon some subfacet or subsegment.)*
3. *When a point p is chosen as the circumcenter of a skinny tetrahedron, the distance to the nearest vertex is at least LS(p)/C_T. (Again, p may be added to the tetrahedralization, or may be rejected because it encroaches upon some subfacet or subsegment.)*
4. *When a point p is chosen as the circumcenter of an encroached subfacet, the distance to the nearest vertex is at least LS(p)/C_F. (p may be added to the tetrahedralization, or may be rejected because it encroaches upon some subsegment.)*
5. *When a vertex p is added as the midpoint of a split segment, the distance to its nearest neighbor vertex is at least LS(p)/C_S.*

[9] These two worst cases can not, of course, apply simultaneously. With careful analysis, we could perhaps remove some slack from the constants we derive, but we see no hope for improving the angle bound for diametral lenses.

[10] Following Shewchuk, we say that facets are 'visible' to each other if there are no facets or segments between them in the domain to be meshed. A facet (or segment) 'projects' onto another facet if it intersects the right prism built on that facet.

Proof. *Case 1*. Statement 1 of Lemma 2 is true by the definition of the length scale *LS* from the local feature size *lfs*, provided only that the constant R in Eq. (1) is ≥ 1.

Having established the truth of Lemma 2 for the initial mesh, we will proceed by induction to prove that it must be true for all meshes generated by the algorithm. As such, we will assume that Lemma 2 holds for all points in the mesh and determine the bounds on C_S, C_F, C_T, C_L and G that are required for Lemma 2 to hold for newly inserted points.

Case 2. We consider first the case of insertion to split a large tetrahedron, as shown in Fig. 12. By definition, the circumradius of $\triangle abcd$ is greater than $\sqrt{3}/2$ times the average of the length scales at its vertices. The distance from p to the nearest point is the circumradius of T, or

$$r \geq \frac{\sqrt{3}}{2} \frac{LS(a) + LS(b) + LS(c) + LS(d)}{4} \tag{11}$$

We can bound the length scale at p in terms of this same average using Eq. (2)

$$LS(p) \leq LS(a) + \frac{r}{G}$$

$$LS(p) \leq LS(b) + \frac{r}{G}$$

$$LS(p) \leq LS(c) + \frac{r}{G}$$

$$LS(p) \leq LS(d) + \frac{r}{G}$$

$$LS(p) \leq \frac{LS(a) + LS(b) + LS(c) + LS(d)}{4} + \frac{r}{G} \tag{12}$$

Combining inequalities (11) and (12), we get

$$r \geq \frac{\sqrt{3} LS(p)}{2} - \frac{r\sqrt{3}}{2G}$$

$$r \geq \frac{LS(p)}{\frac{2}{\sqrt{3}} + \frac{1}{G}} \tag{13}$$

This inequality places a lower bound on point spacing for points inserted to split large tetrahedra, and confirms Statement 2 of Lemma 2 for any

$$C_L \geq \frac{2}{\sqrt{3}} + \frac{1}{G} \tag{14}$$

The lower bound on C_L becomes smaller for slow change in cell size (corresponding to large values of G).

Case 3. We consider next the case of insertion to split a badly shaped tetrahedron (one whose ratio of circumradius to shortest edge length, $r/l_{\min}$, exceeded a fixed bound B), as illustrated in Fig. 13. Without loss of generality, we can label vertices so that a and b are connected by the shortest edge (of length $l_{\min}$), and a was inserted in the mesh after b (or both were input vertices). The radius of the vertex-free circle around a is r'; five subcases for relating r' to $LS(a)$ arise, depending on why a was inserted:

Subcase 3a: *a was an input vertex*. Then so was b, and Statement 1 of Lemma 2 applies. The distance $l_{\min} \equiv |ab| \geq R \cdot LS(a)$.

Subcase 3b: *a was inserted to split a large tetrahedron*. The circumradius r' of that tetrahedron is no larger than $l_{\min}$, because vertex b was not inside the circumsphere of the tetrahedron a split. Then Statement 2 of Lemma 2 applies, and $l_{\min} \geq r' \geq LS(a)/C_L$.

Subcase 3c: a was inserted to split a badly shaped tetrahedron. By a similar argument as the previous subcase and using Statement 3 of Lemma 2, $l_{\min} \geq r' \geq LS\ (a)/C_T$.

Subcase 3d: a was inserted to split an encroached boundary subfacet. We know that b does not lie inside the equatorial sphere of the encroached subfacet (which is centered at a), because all points inside that sphere were deleted before a was inserted. Statement 4 of Lemma 2 applies, and $l_{\min} \geq r' \geq LS(a)/C_F$.

Subcase 3e: a was inserted to split an encroached boundary subsegment. We can use the same reasoning as the previous subcase, and apply Statement 5 of Lemma 2 to get $l_{\min} \geq r' \geq LS(a)/C_S$.

As we show, it is possible (in fact preferable) to choose C_S larger than all the other C's. Then the inequality $l_{\min} \geq LS(a)/C_S$ (subcase 3e) causes the most difficult in satisfying Statement 3 of Lemma 2 by making the length scale at p larger than that for any other subcase.

We know that the cell $\Delta abcd$ is being split because it is poorly shaped, implying that $r/l_{\min} > \mathrm{B}$. Combining this

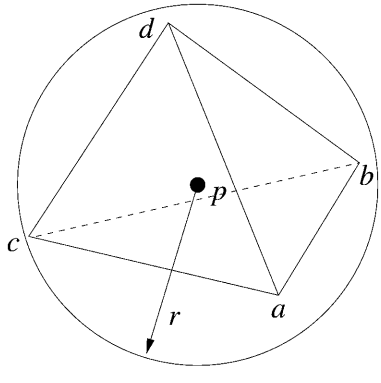


Fig. 12. Lemma 2, Statement 2: p added as circumcenter of large tetrahedron $\triangle abcd$.

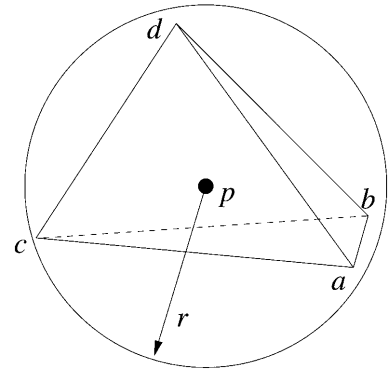


Fig. 13. Lemma 2, Statement 3: p added as circumcenter of badly shaped tetrahedron $\triangle abcd$.

result with that of subcase 3e and the relationship between length scale at p and a, we have[11]

$$LS(p) \le LS(a) + \frac{r}{G} \le C_S l_{\min} + \frac{r}{G} \le \frac{rC_S}{B} + \frac{r}{G} \quad (15)$$

Statement 3 of Lemma 2 says that for a point inserted to split a badly-shaped tetrahedron, $LS(p) \le rC_T$. For Lemma 2 to hold, this must hold whenever Inequality 15 does, so we require that

$$C_T \ge \frac{1}{G} + \frac{C_S}{B} \quad (16)$$

Case 4. Now we turn our attention to the case in which vertex p is added to the mesh to split a subfacet $f \equiv \triangle abc$, because some vertex or proposed tetrahedron circumcenter z lies inside the equatorial lens of f. This case is illustrated in Fig. 14. There are three subcases:

Subcase 4a: z lies on a segment or facet, which can not share a vertex with f, because we have assumed that facets are separated by 60° from segments and other facets that project on them, making encroachment impossible. Therefore, p and a lie on non-adjacent boundary entities, and the length scale at p is $LS(p) \le \frac{1}{R} |\vec{a} - \vec{p}|$. To satisfy Lemma 2 in this subcase, we therefore require that $C_F \ge \frac{1}{R}$. Because $R \ge 1$, this inequality is always satisfied for $C_F \ge 1$.

Subcase 4b: z is a point at the circumcenter of a large tetrahedron T. z has of course been rejected for insertion, and all points inside the equatorial sphere of subfacet $\triangle abc$ have been removed, so p has no vertex remaining in the mesh closer than a distance r. The circumradius r' of T is smaller than the shortest of $\overline{za}$, $\overline{zb}$, and $\overline{zc}$, because T's circumcircle must be point-free. The worst case places a at the apex of the equatorial lens of $\triangle abc$, so $r' \le \frac{2}{\sqrt{3}} r$. Also, we know from this lemma that $LS(Z)/C_L \le r'$. The definition of the length scale gives (for the worst case, where z approaches a, b or c)[12]

$$LS(p) \le LS(z) + \frac{1}{G} r \le r' C_L + \frac{1}{G} r \le r\left(\frac{2}{\sqrt{3}} C_L + \frac{1}{G}\right)$$

This inequality satisfies Lemma 2 provided that

$$C_F \ge \frac{1}{G} + \frac{2}{\sqrt{3}} C_L \quad (17)$$

Subcase 4c: z is a point at the circumcenter of a skinny tetrahedron. The same reasoning can be applied as in the previous subcase, with the result that

$$C_F \ge \frac{1}{G} + \frac{2}{\sqrt{3}} C_T \quad (18)$$

Case 5. Finally, we consider the case in which vertex p is added to the mesh to split a subsegment s, because some vertex or proposed tetrahedron circumcenter z lies inside the diametral sphere of s, as shown in Fig. 15. The same reasoning can be applied as in Case 4. However, because the subsegment is protected by a diametral sphere instead of an equatorial lens, the constant $2/\sqrt{3}$ must be replaced by $\sqrt{2}$. The constraints on the constants of Lemma 2 from this case are:

$$C_S \ge \frac{1}{G} + \sqrt{2} C_L \quad (19)$$

and

$$C_S \ge \frac{1}{G} + \sqrt{2} C_T \quad (20)$$

To establish the truth of Lemma 2, we must find values of C_S, C_F, C_T, and C_L that simultaneously satisfy inequalities (14), (16), (17), (18), (19), and (20) Provided we choose

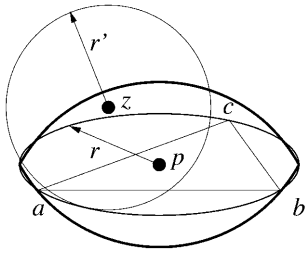


Fig. 14. Lemma 2, Statement 4: p added to split an encroached boundary facet.

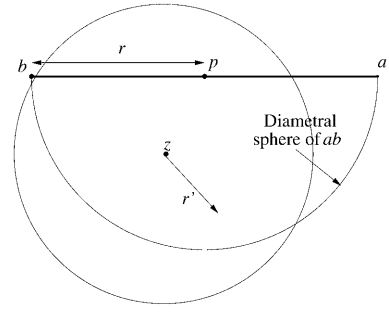


Fig. 15. Lemma 2, Statement 5: p added to split an encroached boundary edge.

[11] Note that each inequality in the above sequence is a possible source of slack in the constants we will later determine; in practice, the inevitable accumulation of slack here and elsewhere makes the algorithm perform better than we can prove.

[12] These two worst cases can not of course apply simultaneously. As in two dimensions, with careful analysis, we could perhaps remove some slack from the constants we derive, but we see no hope for improving the aspect ratio bound.

$$C_S \geq \frac{B}{G}\frac{1+\sqrt{2}}{B-\sqrt{2}} \tag{21}$$

$$C_T \geq \frac{1}{G}\frac{B+1}{B-\sqrt{2}} \tag{22}$$

$$C_T \geq C_L \geq \frac{2}{\sqrt{3}} + \frac{1}{G} \tag{23}$$

we will always satisfy inequalities (16), (19), and (20). The last of these implies that inequality (18) is more restrictive than inequality (17), so we can find a limit for C_F;

$$C_F \geq \frac{2}{\sqrt{3}} C_T + \frac{1}{G} \geq \frac{2}{\sqrt{3}}\frac{1}{G}\frac{B+1}{B-\sqrt{2}} + \frac{1}{G}$$

$$C_F \geq \frac{1}{G}\frac{B(1+\frac{2}{\sqrt{3}}) + (\frac{2}{\sqrt{3}} - \sqrt{2})}{B-\sqrt{2}} \tag{24}$$

It is easy to show that, for the lower bounds on each constant, C_S is in fact the largest, as required by Case 3.

All of the constants are bounded for $B > \sqrt{2}$. As $B \rightarrow \sqrt{2}$, the additional condition that $C_T \rightarrow C_L$ implies that

$$\frac{1}{G}\frac{1}{B-\sqrt{2}} = \frac{2}{\sqrt{3}\,(1+\sqrt{2})}$$

This condition is sufficient to keep all constants bounded except G, the grading parameter, which must go to infinity, implying that the length scale of the mesh will be uniform.

In summary, for any $R \geq 1$ and all bounds $B \geq \sqrt{2}$ on the ratio of circumradius to shortest edge length for the worst tetrahedron in the mesh, all statements of Lemma 2 hold, and the nearest neighbor to a vertex newly inserted into the mesh will be no closer than the length scale at that vertex divided by a constant factor. We have also established a limit on how quickly cells can change size (a lower bound on G); this bound implies that as $B \rightarrow \sqrt{2}$, the mesh will become uniform.

Appendix C: Termination and Size Optimality

In both two- and three-dimensions, we can use the quality lemmas to prove the following theorem about finite mesh size and mesh size optimality.

Theorem 1. *Given a vertex p in the output triangular or tetrahedral mesh, its nearest neighbor vertex q is at a distance at least* $LS(p)/(C_S + 1/G)$. *This implies mesh size optimality.*

Proof. Lemmas 1 (2D) and 2 (3D) handle the case where p is inserted after q. If q is inserted last, then we apply either lemma to q:

$$|\vec{q} - \vec{p}| \leq \frac{LS(q)}{C_S}$$

But $LS(p) \geq LS(q) + \frac{|\vec{q} - \vec{p}|}{G}$, so

$$|\vec{q} - \vec{p}| \geq LS(p) - \frac{|\vec{q} - \vec{p}|}{C_S}$$

and the theorem follows, with only minor algebra.

Because the shortest edge in the mesh must be longer than $\frac{LS_{\min}}{C_S + \frac{1}{G}}$, each cell has a finite size, and only a finite number of them will be required. Therefore, the algorithm will always terminate.

We can say more than that. Because the shortest possible edge is within a constant factor of the length scale locally, the smallest possible triangle (tetrahedron) is within the square (cube) of that same factor of the size of a triangle (tetrahedron) whose edges all match the length scale. This implies that the size of the mesh must be within a constant factor of the size of the smallest possible mesh whose cells meet the quality bound and whose edges have length within a constant factor of the length scale locally.