Revisiting Delaunay Refinement Triangular Mesh Generation on Curve-bounded Domains

Serge Gosselin and Carl Ollivier-Gooch

Department of Mechanical Engineering, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada

Email: *gosselin@mech.ubc.ca*

ABSTRACT

An extension of Shewchuk's Delaunay Refinement algorithm to planar domains bounded by curves is presented. A novel method is applied to construct constrained Delaunay triangulations from such geometries. The quality of these triangulations is then improved by inserting vertices at carefully chosen locations. Amendments to Shewchuk's insertion rules are necessary to handle cases resulting from the presence of curves. The algorithm accepts small input angles, although special provisions must be taken in their neighborhood. In practice, our algorithm generates graded triangular meshes where no angle is less than 30°, except near small input angles. Such meshes are suitable for use in numerical simulations.

1 INTRODUCTION

The mesh generation process, during which a geometry is tessellated into subdomains suitable for computation, is an integral part of CFD simulations. While unstructured mesh methods are known for their ability to automatically discretize complex geometries, satisfying a numerical solver's stringent need for quality can be difficult. Mesh quality is known to affect both the accuracy and efficiency of numerical simulations. The finite-element literature contains proof that solvers are impaired by element angles not bounded away from 0° and 180° [1]. Experiments also established that finite volume discretizations, when defined on poorly-shaped cells, suffer from similar problems [4].

In this context, being able to automatically generate high quality meshes can potentially reduce the time devoted to CFD pre-processing. The need for an algorithm exhibiting these qualities has lead to the appearance of the Delaunay refinement paradigm in mesh generation. By inserting vertices at the circumcenter of poorly-shaped triangles, Delaunay refinement algorithms eliminate small angles from a triangulation, thus improving its quality. The first practically relevant algorithms are due to Ruppert [9] and Chew [3]. These not only produce meshes of provably good quality, but also provide theoretical guarantees regarding mesh sizing and, under certain input conditions, termination. Ruppert's and Chew's schemes are sufficiently similar for Shewchuk to analyze them in a common framework [10, 13]. His findings lead to an hybrid algorithm improving quality guarantees and relaxing input restrictions. Shewchuk's algorithm is presented in section 3.

The aforementioned algorithms have the common shortcoming of only accepting input domains whose boundaries are piecewise linear. This restriction can be impractical in applications where geometric information comes, for instance, in the form of parametric curves. By defining a condition on curve sampling, Boivin and Ollivier-Gooch extended Delaunay refinement's application to geometries bounded by curves [2]. The condition allows Shewchuk's algorithm to correctly handle insertion near boundaries. However, in many cases, the sampling is not dense enough for the mesh to conform to the domain's boundary. They propose a series of heuristics to overcome this difficulty. We argue that these heuristics are tedious to implement correctly and that they can fail in some cases.

In this paper, we study the boundary conformity problem in the context of Delaunay refinement mesh generation on curve-bounded domains. We introduce a pre-refinement technique which allows us to construct a Delaunay triangulation constrained to the domain's boundary. Shewchuk's algorithm can then be applied to this triangulation to improve its quality. To prevent topology-infringing insertions, modifications to Shewchuk's insertion rules are necessary. Section 4 covers these topics.

Finally, we note that special care must be taken when geometric curves sharing a common endpoint are separated by an angle of less than 60°. In such situations, the argument used to prove termination no longer holds [11]. Pav [8] thoroughly describes the failure mechanisms triggered by small angles. By modifying the algorithm's behavior in the neighborhood of these small angles, Miller et al. [5] offer a simple and elegant solution to the termination problem. We follow the same idea and adapt their method to curves. This is presented in section 5.

In summary, we propose a Delaunay refinement method that produces triangular meshes where no angle is less than 30°, except perhaps near small angles. The presence of these small angles does not however affect termination. The implementation is included within the GRUMMP mesh generation libraries [7] (sources available through the project's website). The interface to define, build and query the geometry is derived from the Common Geometry Module [14]. Meshes produced by our algorithm are presented in section 6.

2 Problem definition

The mesh generation problem can be defined in terms of the input to the meshing algorithm, and of its desired output.

The input to the meshing algorithm is a Piecewise Smooth Curve Complex (PSCC) consisting of a set of points, $\mathcal{P} \subseteq \mathbb{R}^2$ and a set of curves $\mathcal{C}$ such that

1. curves in $\mathcal{C}$ are either bounded by two endpoints in $\mathcal{P}$, or are closed and bounded by a single endpoint in $\mathcal{P}$,
2. two curves in $\mathcal{C}$ can only intersect each other at their endpoints and
3. points in $\mathcal{P}$ intersect $\mathcal{C}$ only at an endpoint.

Note that if all curves in $\mathcal{C}$ are straight segments, the previous definition is equivalent to that of a Planar Straight Line Graph (PSLG) — the input typically accepted by most Delaunay refinement algorithms.

Upon termination, the algorithm outputs a simplicial complex $\mathcal{S}$ formed by the union of a set of vertices $\mathcal{V}$, a set of edges $\mathcal{E}$ and a set of triangles $\mathcal{T}$, satisfying these properties:

1. **Constrained Delaunay**: Every simplex in $\mathcal{S}$ satisfies the constrained Delaunay property with respect to $\mathcal{V}$.

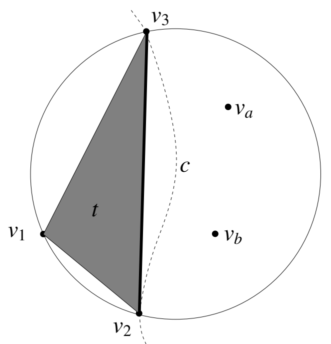

Figure 1: A constrained Delaunay triangle t (shaded). Vertices v_a and v_b are located inside the circumcircle of t but cannot be seen from its interior. Visibility is blocked by $\overline{v_2 v_3}$, a boundary edge discretizing curve c.

2. **Quality**: Most triangles in $\mathcal{T}$ are of good quality.
3. **Vertex count**: $\left|\mathcal{V} \setminus \mathcal{P}\right|$ is small.

The first condition stipulates that the output of the algorithm is a constrained Delaunay triangulation (CDT). A simplex is said to be constrained Delaunay with respect to $\mathcal{V}$ if [12]

- it conforms to the domain's boundary
- there exist a circle circumscribing the simplex, such that the circle does not contain any vertex visible [1] from the simplex relative interior.

In a PSCC, a vertex $v \in \mathcal{V}$ satisfies conformity if $\mathcal{P} \subseteq \mathcal{V}$ while an edge $e \in \mathcal{E}$ satisfies conformity if its interior does not intersect any curve in $\mathcal{C}$. If all simplices in $\mathcal{V} \cup \mathcal{E}$ conform to the boundary, then the triangulation conforms to the boundary. If all entries in $\mathcal{S}$ respect both these conditions, then their union forms a CDT. A constrained Delaunay triangle is depicted in figure 1.

The quality of a triangle $t \in \mathcal{T}$ is measured by its circumcenter-to-shortest edge ratio r/l. This ratio is directly related to $\theta_{\min}$, the smallest angle of t, by the formula $r/l = (1/2\sin\theta_{\min})$. Experiments demonstrate that our algorithm terminates and that upon termination, the output CDT has $r/l \leq 1$ for all $t \in \mathcal{T}$, except possibly near small input angles. Poor quality triangles located "across" small angles in the input PSCC will survive Delaunay refinement.

[1] A vertex is visible from a simplex if there exists a line going from the vertex to any point inside the simplex that does not cross any input curves

3 Delaunay Refinement

Delaunay refinement algorithms are incremental vertex insertion schemes. As a starting point, they require a Delaunay (or possibly a constrained Delaunay) triangulation of the input domain, which they then proceed to improve by inserting new vertices at well-chosen locations. In this paper, we adapt an algorithm introduced by Shewchuk [10] so it can accept curve-bounded domains as input.

Given a valid initial *constrained* Delaunay triangulation, Shewchuk's algorithm eliminates bad quality triangles by inserting a new vertex at their circumcenter. To avoid adding a vertex outside the domain or too close to a boundary (thus creating an unnecessarily small feature), inserting a vertex encroaching on the boundary is forbidden. A boundary edge is said to be encroached upon if a *visible* vertex is located inside its diametral lens (see figure 2). Hence the following two insertion rules:

- If a triangle is badly shaped, i.e. its circumradius-to-shortest edge ratio is above a certain threshold, a vertex is inserted at its circumcenter, UNLESS
- the proposed insertion location encroaches on one or more boundary edges. In this case, the vertex is not inserted; instead, the encroached edges are bisected. If a bisection point encroaches on other boundary edges, those must also be bisected. Before bisecting a boundary edge, all visible *interior*[2] vertices located inside its diametral circle (ref. figure 2) must be deleted from the mesh.

Note that boundary edge splitting always has priority over bad triangle insertion.

Shewchuk's algorithm terminates and outputs a size-optimal triangulation T where all triangles have $r/l \leq \frac{2\sqrt{3}}{3}$ i.e. $\theta_{\min} \geq 25.7°$. Termination is proved using a compaction argument. For this argument to hold, input segments sharing a common endpoint must not subtend angles of less than 60°. Since T is size-optimal [13], the number of vertices inserted ($\left|\mathcal{V} \setminus \mathcal{P}\right|$) is typically small. It has also been shown that no edges in T have lengths smaller than a constant factor times some local feature size [9]. T is therefore a graded triangulation i.e. triangle sizes can vary over different parts of the domain.

It is important to note that, in practice, the algorithm is known to terminate with the minimum angle bound

[2]A vertex is said to be "interior" if it is not located on the domain's boundary.

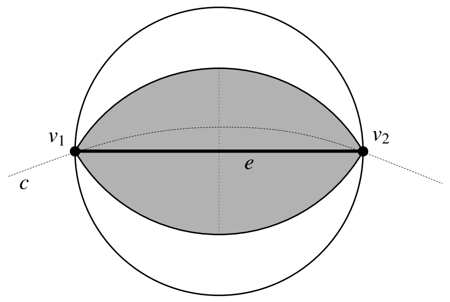

Figure 2: The diametral lens (shaded) and diametral circle of boundary edge e. The lens is the interior of two circular arcs intersecting at v_1 and v_2. Each arc has its center on the other arc, along the bisector of e.

set to 30°. Further increase leads to a degradation of the size-optimality guarantee; in the limit, T must be of infinite size to satisfy the angle bound.

4 Application to curves

This section presents a solution to the meshing problem defined in section 2. Triangulations are built by following three successive operations. Firstly, the curves in $\mathcal{C}$ are discretized. An initial set of vertices $\mathcal{V}_0$ and an initial set of edges $\mathcal{E}_0$ are obtained. These sets are then augmented to form $\tilde{\mathcal{V}}_0$ and $\tilde{\mathcal{E}}_0$, two sets in which all items are constrained Delaunay. A constrained Delaunay triangulation is built and given as input to a slightly altered version of Shewchuk's algorithm. New vertices are inserted in the mesh until quality and size criteria are met.

4.1 Curve discretization

As defined in section 2, the mesh generation algorithm is given a PSCC as input. To triangulate the PSCC, the curves in $\mathcal{C}$ must first be discretized. The discretization operation places vertices along a curve $c \in \mathcal{C}$ and then connects two successive vertices with a boundary edge.

We observe that maintaining quality properties is more easily achieved when refining a mesh than when coarsening it. Thus, choosing a proper discretization measure is important. On the one hand, boundaries must be kept as coarse as possible — it is always possible to refine them later — but on the other hand, the refinement algorithm must not fail because of a discretization that is too coarse to properly approximate the curves.

Exploiting the fact that Shewchuk's algorithm refuses to insert a vertex that will encroach on the boundary, Boivin and Ollivier-Gooch [2] propose a curve sampling condition where the union of all diametral lenses ultimately encloses the curves in $\mathcal{C}$. A protection zone is thus created in the neighborhood of the boundary, preventing vertices from being inserted outside the domain. To achieve this, they limit the total variation of the tangent angle $TV(\theta) = \int_c |d\theta|$ — where θ is the angle subtended by the tangent to curve c — to $\pi/6$ between two consecutive vertices along c. We use the same sampling condition, noting that a denser sampling is also acceptable.

From this initial vertex set, Boivin and Ollivier-Gooch directly build a Delaunay triangulation. They observe that in many cases, satisfying boundary conformity is made impossible by the coarseness of the discretization. Forcing boundary edges into the triangulation would lead to crossovers, and thus to an invalid triangulation. They acknowledge the situation and propose heuristics to remedy the problem. The next section presents a more straightforward approach to satisfy conformity while preserving the constrained Delaunay nature of the discretization.

4.2 Boundary recovery and initial triangulation

Remember from section 2, that all simplices in a constrained Delaunay triangulation must themselves be constrained Delaunay. Instead of constructing a triangulation from $\mathcal{V}_0$ and $\mathcal{E}_0$, we propose to enrich the sampling beforehand to create an augmented set of boundary vertices $\tilde{\mathcal{V}}_0 \supseteq \mathcal{V}_0$. The objective is to obtain a set of boundary edges $\tilde{\mathcal{E}}_0$ which all satisfy the constrained Delaunay property. This refinement operation must efficiently detect if any vertices are located inside a boundary edge's diametral circle *and* if so, determine if these vertices are visible from any location on the boundary edge.

To determine if the boundary necessitates further refinement, we rely on two spatial indexing trees, τ_1 and τ_2. τ_1, contains the coordinates of the vertices of $\tilde{\mathcal{V}}_0$ and τ_2, the axis-aligned, minimal bounding rectangles of the edges in $\tilde{\mathcal{E}}_0$. Both are implemented in the form of a R-Tree.

Boundary refinement proceeds as follows: For an edge $e \in \tilde{\mathcal{E}}_0$ bounded by vertices v_1 and v_2, τ_1 is queried to find if any vertices are contained in its diametral circle. If such a vertex exists, it is then tested for visibility. If a vertex is found to lie inside the diametral circle of e *and* is visible from the interior of e, then a new vertex, v, is inserted on the curve, thereby splitting e into two new edges, $e_1 = \overline{v_1 v}$ and $e_2 = \overline{v v_2}$. The insertion location is chosen such that the two newly created edges have equal $TV(\theta)$ between their endpoints. Finally, v is added to $\tilde{\mathcal{V}}_0$, e is removed from $\tilde{\mathcal{E}}_0$, and e_1 and e_2 are added to $\tilde{\mathcal{E}}_0$; the trees τ_1 and τ_2 are updated accordingly. The process is repeated until no edge in $\tilde{\mathcal{E}}_0$ is found to contain a visible vertex in its diametral circle. At this point, all edges are constrained Delaunay.

Visibility of vertex v can be blocked from boundary edge e in two ways:

1. An unmeshed region — a hole in the geometric domain — lies between v and e.
2. One or many edges in $\tilde{\mathcal{E}}_0$ are located between v and e, such that it is impossible to define a line from v to any point on e without intersecting a member of $\tilde{\mathcal{E}}_0$.

Let's assume v to be inside the diametral circle of e. We must verify that v is visible from e against the two situations above. The following assumptions about the geometry interface implementation are necessary:

- e knows its *parent* curve i.e. the curve it discretizes,
- a curve can determine whether a vertex lies on its right side or on its left side and
- a curve knows whether it bounds a meshed region onits right side and/or onits left side.

Visibility testing proceeds as follows: e first identifies its parent curve c_e. The geometry interface is then queried to determine whether v lies on the left or right side of e *and* of c_e. If the queries return different sides, e crosses another edge and as such, e is split regardless of visibility. Otherwise, if v lies on the side of an unmeshed region bounded by c_e then v is *not* visible from e and no split is needed. If however v lies on the side of a meshed region, then a second test is necessary to establish if the visibility is occluded by other nearby boundary edges.

As illustrated in figure 3, for v to be occluded from e, edges discretizing boundary curves, or part thereof, must intersect triangle t. τ_2 is queried for intersections with the bounding box of t. A subset $\bar{\mathcal{E}} \subseteq \tilde{\mathcal{E}}_0$ containing edges potentially making v invisible from e is obtained. We argue that it is sufficient to verify that segments $\overline{vv_1}$ and $\overline{vv_2}$ intersect a member of $\bar{\mathcal{E}}$ to declare v to be invisible from e. If, in actuality, there is a gap through which v can be seen from e (as is the

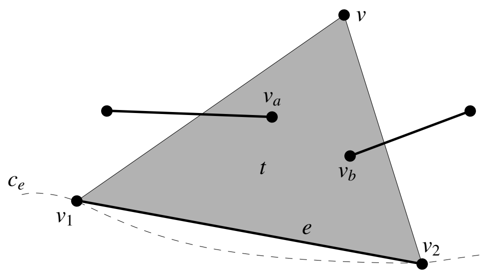

Figure 3: Visibility of v from e. It is sufficient to locate intersections along $\overline{vv_1}$ and $\overline{vv_2}$ to declare v invisible from e. v_a or v_b will eventually force a split of e.

case in figure 3), then there must be other vertices in the diametral circle of e. One of these vertices will eventually be declared visible and e will be split.

Once all boundary edges have been made constrained Delaunay, a triangulation is constructed from $\tilde{\mathcal{V}}_0$. The Delaunay triangulation of a large square bounding box enclosing all members of $\tilde{\mathcal{V}}_0$ is first created. Vertices are then incrementally added using the Bowyer-Watson method [13] thereby preserving the triangulation's Delaunay property. At this point, most boundary edges do not have a vertex inside *or* on their diametral circle. As such, they will be part of the Delaunay triangulation. The remaining edges can always be recovered through a series of edge flips. Because only a few edge flips are typically required, boundary recovery is very fast. As a final step, triangles located outside the computational domain, including those attached to the corners of the bounding box, are deleted. A constrained Delaunay triangulation of the input PSCC is thus obtained.

4.3 Triangulation Refinement

The constrained Delaunay triangulation constructed in the previous section is not readily suitable for CFD. The triangles might be too big to properly capture solution features, or they might be of such poor quality that solution convergence would be impossible. As described in section 3, new vertices can be added to the triangulation to improve its quality. Furthermore, since Delaunay refinement offers a guarantee on mesh grading, a Lipschitz sizing field can be defined over the computational domain to control mesh density. This sizing field can be obtained, for instance, by estimating the error in a numerical solution, thereby producing a solution-adapted mesh. Note that the sizing field is not necessary if a quality mesh is needed, regardless of triangle sizing. Our current implementation defines a sizing field in a manner similar to that described in [6], where a geometric length scale is defined as a function of the domain's geometry and two user-defined parameters. A triangle that is too large with respect to the sizing field or that is of poor quality is qualified as *needing refinement*.

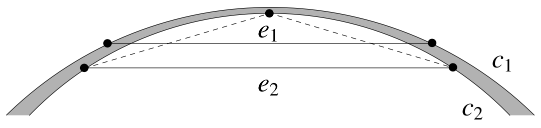

Figure 4: Curves c_1 and c_2 bound an unmeshed region (shaded) and are arbitrarily discretized by edges e_1 and e_2. A split of e_2 will result in a boundary crossover.

To correctly handle vertex insertion in the presence of curved geometries, two modifications to the insertion procedure presented in section 3 are necessary.

Firstly, instead of bisecting an encroached edge by inserting at its midpoint, the new vertex is placed directly *on* the edge's parent curve. Encroachment splits then contribute to reducing the Hausdorff distance between the curve and its discretization. The split location can be chosen as a point that is equidistant to the edge's endpoints or, as was done in [2], by selecting a point such that the two newly created edges have equal $TV(\theta)$ over their span. Splitting encroached edges in this manner can however result in boundary crossovers. As illustrated in figure 4, a coarse boundary discretizations may cause a vertex to be inserted "behind" existing edges. For this reason a third insertion rule must be appended to those of section 3.

The Delaunay refinement algorithm accepting PSCC as input proceeds as follows:

- If a triangle needs refinement, a vertex is inserted at its circumcenter, UNLESS

- the proposed insertion location encroaches on one or many boundary edges. In this case, the vertex is not inserted; instead, the encroached edges are split by inserting on their parent curves, UNLESS,

- at least one proposed split location causes boundary crossovers. In this case, the split is not performed. Instead, the edges getting crossed are split by inserting on their parent curves. If a split location encroaches on other boundary edges, those must also be split. Before splitting a boundary edge, all visible interior vertices inside its diametral circle must be deleted from the mesh.

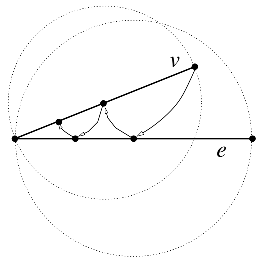

Figure 5: Runaway encroachment. Vertex v encroaches on boundary edge e. An attempt to fix encroachment will cause infinite insertions.

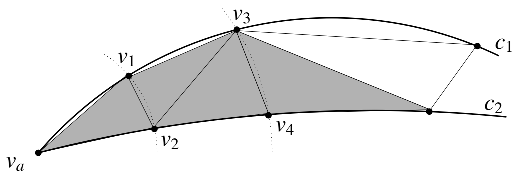

Figure 6: Small angle protection. Curves c_1 and c_2 are separated by a small angle. Vertices v_1 through v_4 are on concentric shells centered at v_a. An insertion to improve the shaded triangles' quality will never be attempted.

Crossover preventing splits always have priority over encroachment splits, which have priority over circumcenter insertions.

To efficiently detect if a boundary insertion will cause edges to intersect, tree τ_2, defined in section 4.2, is used once again. If the split were to happen, two new edges would be created. Potential intersections are identified by querying τ_2 with the bounding boxes of these edges. τ_2 must therefore be updated during the refinement process so that it always contains the current boundary discretization.

Insertion ordering is controlled by a priority queue. A priority is computed based on a triangle's size and quality. A large triangle will be given higher insertion priority than one having bad quality. Because of the algorithm's size-optimality guarantee, only a few insertions are typically necessary to satisfy the quality criterion once the sizing field is respected.

5 SMALL INPUT ANGLES

Shewchuk recognizes that his algorithm can fail to terminate when the input geometry harbors small angles. Figure 5 illustrates one of the failure mechanisms where midpoint-midpoint interactions lead to runaway encroachment splits . In such a situation, the compaction argument used to prove termination no longer holds. A cascade of smaller and smaller features will be created in an attempt to fix encroachment.

To "block" runaway encroachment splits, Ruppert originally suggested modifying the edge bisection rule around small angles [9]. Instead of inserting at the edge's midpoint, new vertices are placed at the intersection between the edge and concentric shells centered at the apex of small angles. Miller et al. [5] improved on the concentric shell idea. They were able to prove termination for piecewise linear geometries containing arbitrarily small angles. Their algorithm never attempts to remove a bad quality triangle if the edge opposed its smallest angle is bounded by two vertices on a shell. Figure 6 illustrates how this method is adapted to curves.

Let's assume e_1 and e_2 to be two boundary edges connected at apex vertex v_a and having c_{e1} and c_{e2} as parent curves. v_a is at the base of a small angle if the tangents to c_{e1} and c_{e2}, evaluated at v_a, form an angle of less than 60°. We define a *small angle complex* as the set of boundary edges sharing a common apex vertex and whose parent curves form contiguous small angles (a single apex vertex can be at the base of many small angle complexes). Finally, we note $\|e_{\min}\|$, the length of the shortest edge in a given small angle complex.

The method assumes that small angles are protected by a shell. New vertices must therefore be added to construct these shells. This operation is performed right after initial curve discretization (ref. section 4.1). Vertices are added as follows: The boundary edges present in two small angle complexes — bounded by two apex vertices — are initially split in the usual way. Then, given a small angle complex Σ, a shell of radius $r_S = \frac{1}{2}\|e_{\min}\|$ is obtained. Every boundary edge of Σ is split by inserting a new vertex on its parent curve, at a distance r_S from the apex vertex. The process is repeated for all small angle complexes.

Triangles whose shortest edge spans a small angle are tagged so that no attempt will be made to improve their quality (see figure 6). If, at any time, a boundary edge in some small angle complex becomes encroached and needs to be split, a new shell of radius $r'_S = \frac{1}{2}r_S$ is computed and the edges in the small angle complex are all split in the manner just described. The list of "untouchable" triangles is updated accordingly.

We do not provide proof that this adaptation of Miller et al. offers a termination guarantee. In practice, the performance of this method is far superior to Ruppert's method originally implemented by Boivin and Ollivier-Gooch [2]. In our experiments, mesh refinement never failed in the presence of small input angles.

6 Results

We present triangular meshes produced by our algorithm. Each mesh is accompanied by an histrogram describing the angle distribution within the triangulation. Figures 7 shows the close-up of the coarsest possible quality mesh around a 4-element airfoil. The mesh in figure 8 uses the same input geometry, but the mesh satisfies a sizing field. Finally, figure 9 presents the mesh of a generic curve-bounded geometries containing a few small angles. All angles in the output meshes are above $30°$ except close to small angle inputs.

7 Conclusion

A mesh generation algorithm producing quality triangulation from curve-bounded domains was presented. Working from a previously known curve sampling condition, we developed a novel method formalizing the construction of a constrained Delaunay triangulation from a PSCC. This triangulation satisfies the input requirements of Shewchuk's Delaunay refinement algorithm. A slightly modified version of Shewchuk's algorithm is used to improve the triangulation's quality through vertex insertion. Termination in the presence of small angles is also addressed.

Work to extend Delaunay refinement to volumes bounded by piecewise smooth surfaces is already under way. Discretization techniques similar to those presented in this paper are utilized. In the long run, we would like to automatically generate provably good tetrahedral meshes directly from clean CAD models.

8 Acknowledgments

This work has been funded by the Canadian Natural Sciences and Engineering Research Council under Special Research Opportunities grant SRO-299160.

References

[1] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976.

[2] C. Boivin and C. F. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185–1203, 2002.

[3] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 274–280. ACM, 1993.

[4] L. A. Freitag and C. Ollivier-Gooch. A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency. *International Journal of Computational Geometry and Applications*, 10(4):361–382, 2000.

[5] G. L. Miller, S. E. Pav, and N. J. Walkington. When and why Ruppert's algorithm works. In *Proceedings of the Twelfth International Meshing Roundtable*, pages 91–102. Sandia National Laboratories, 2003.

[6] C. Olliver-Gooch and C. Boivin. Guaranteed-quality simplicial mesh generation with cell size and grading control. *Engineering with Computers*, 17(3):269–286, 2001.

[7] C. F. Ollivier-Gooch. GRUMMP — Generation and Refinement of Unstructured, Mixed-element Meshes in Parallel. http://tetra.mech.ubc.ca/GRUMMP, 1998–2005.

[8] S. E. Pav. *Delaunay Refinement Algorithms*. PhD thesis, Department of Mathematical Sciences, Carnegie Mellon University, 2003.

[9] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92, 1993.

[10] J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1997.

[11] J. R. Shewchuk. Mesh generation for domains with small angles. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pages 1–10. ACM, 1998.

[12] J. R. Shewchuk. Constrained Delaunay tetrahedralizations and provably good boundary recovery. In *Proceedings of the Eleventh Meshing Roundtable*, pages 193–204. Sandia National Laboratories, September 2002.

[13] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):27–74, 2002.

[14] T. J. Tautges. The common geometry module (CGM). Technical Report SAND2004-6252, Sandia National Laboratories, December 2004.

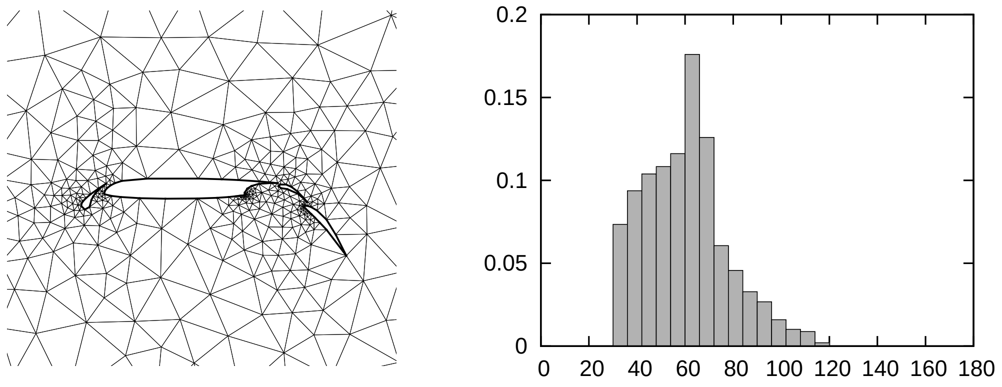

Figure 7: Close-up of the coarsest possible quality mesh around a 4-element airfoil. The mesh has 985 triangles.

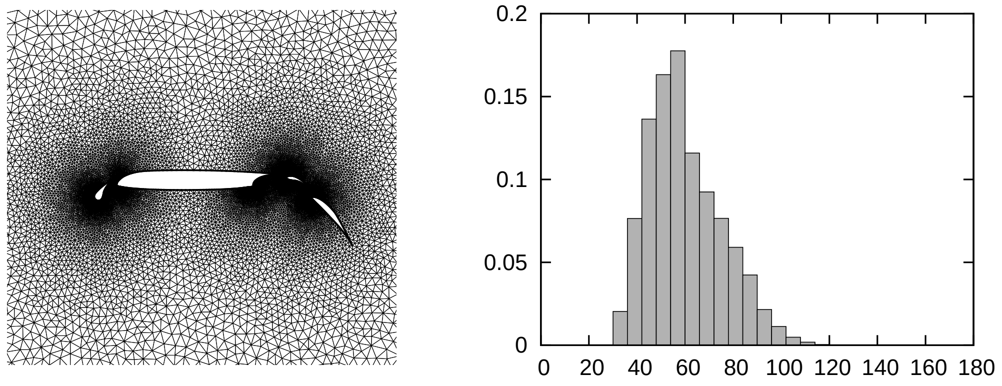

Figure 8: Close-up of a mesh around a 4-element airfoil with sizing control. The mesh has 45350 triangles.

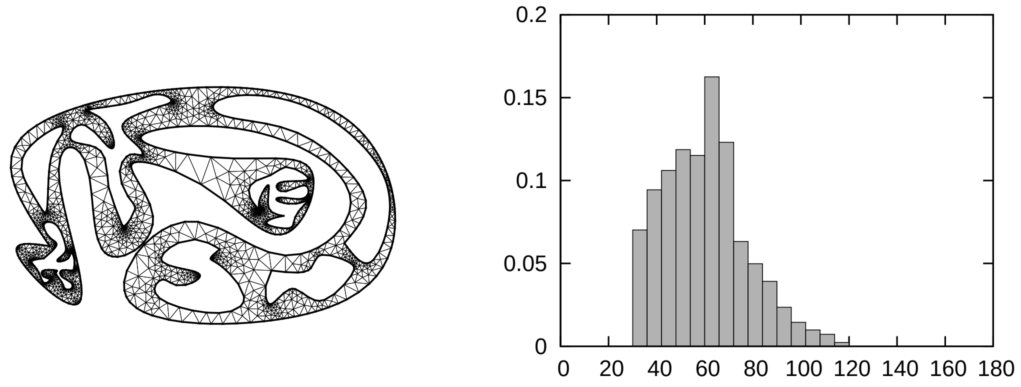

Figure 9: A generic curve-bounded geometry containing a few small angles. The mesh has 2295 triangles.