Delaunay Refinement Algorithms for Numerical Methods

Alexander Rand

April 30, 2009

Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Noel Walkington, Chair
Shlomo Ta'asan
Giovanni Leoni
Gary Miller

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: mesh generation, Delaunay refinement, finite element interpolation

Abstract

Ruppert's algorithm is an elegant solution to the mesh generation problem for non-acute domains in two dimensions. This thesis develops a three-dimensional Delaunay refinement algorithm which produces a conforming Delaunay tetrahedralization, ensures a bound on the radius-edge ratio of nearby all tetrahedra, generates tetrahedra of a size related to the local feature size and the size of nearby small input angles, and is simple enough to admit an implementation. To do this, Delaunay refinement algorithms for estimating local feature size are constructed. These estimates are then used to determine an appropriately sized protection region around acutely adjacent features of the input. Finally, a simple variant of Ruppert's algorithm can be applied to produce a quality mesh. Additionally, some finite element interpolation results pertaining to Delaunay refinement algorithms in two dimensions are considered.

Acknowledgments

My work has been supported by many people of whom a number will surely be omitted. First and foremost, I acknowledge the years of support from my family: my parents, my grandparents, and my sister. They have truly shaped who I am today.

Many of my undergraduate professors encouraged my decision to pursue further studies in mathematics. The most influential was Brian Borchers who surely knew I would seek a Ph.D. in mathematics long before I did. I was also inspired by Ivan Avramidi, Rakhim Aitbayev, W. D. Stone, Bo Deng and Glenn Ledder.

My advisor Noel Walkington cannot be thanked enough. He has always pushed me to overcome challenges which I would not complete on my own and provided critical insights exactly the right moments.

I am grateful for the assistance from all of the mathematics department faculty and staff. I have learned a great deal about research in mathematics from Shlomo Ta'asan. Many other professors in the department have given me invaluable instruction and advice including Giovanni Leoni, Bill Hrusa, Dimitry Kramkov, Jack Schaeffer, Ago Pisztora, William Williams, and Dejan Slepcev.

I have also gained from many discussions with computer scientists at CMU. I understand many things about the field of mesh generation thanks to Gary Miller and his students Benoit Hudson, Todd Phillips, and Don Sheehy.

My research is so heavily influenced by Steven Pav that it is hard to believe that we have never spoken.

My time in graduate school has been much more enjoyable than necessary thanks to my many office mates in the Physical Plant Building including Pall Melsted, David Offner, Pietro Siorpaes, Anne Yust, Prasad Chebolu, Jeff Eisenbeis, Stephen Pankavich, and Nakul Shankar. Wean Hall was also filled with mathematics students with whom I regularly interacted including Brian Seguin, Robert Aguirre, Maxim Bichuch, Peter Lumsdaine, Oleksii Mostovyi, Daniel Specter, and Christina Capena.

This section would not be complete without mentioning Maria Emelianenko, Jason Howell, Carsten Steinebach, and Joseph Young.

Partial funding for this work was provided by the National Science Foundation Grant DMS-0811029.

Most of all, I appreciate the years of support from Barbara Anthony, who is many things including my best friend.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Overview

Consider computing a numerical approximation to the solution of a partial differential equation in a two dimensional polygonal domain via the finite element method. The classical convergence theory involves a conforming triangulation of the domain which satisfies a lower bound over all the angles in the triangulation and gives an error estimate depending on the length of the longest side of each triangle. In the simplest case, one requires a uniform bound on the longest edge of each triangle, but in other cases a graded/adaptive triangulation is desirable. A mesh generator should be able to produce a triangulation based on the sizing restriction given by the user.

Ruppert proposed a simple mesh generation algorithm which elegantly solves this problem [39]. The algorithm is the prototypical Delaunay refinement algorithm: these algorithms are characterized by the use of a Delaunay triangulation as the "mesh" and the incremental insertion of circumcenters of unacceptable simplices to improve this mesh. (Delaunay refinement was first introduced by Chew [13] and the term is often closely associated with the idea of circumcenter insertion for mesh refinement. We will use the term to encompass a broader class of algorithms which allow the insertion Steiner vertices at points other than circumcenters.) For non-acute input domains, Ruppert's algorithm produces a mesh which is (up to a constant factor) as coarse as any conforming triangulation which satisfies the desired minimum angle condition: in other words, the size of the resulting mesh is necessary due to the input geometry. Moreover, it is simple to further refine the triangulation according to a user-given sizing function by continuing to insert circumcenters where necessary. One key to Ruppert's analysis is definition of the local feature size (informally, the distance from a point to the second nearest feature in the input) which is shown to be the correct size for the mesh produced. The primary limitation of Ruppert's algorithm is the restrictive condition that all features of the input (i.e. segments in the "boundary" of the domain) must meet at non-acute angles.

As Ruppert's algorithm is extended to mesh more general domains, it is desirable to preserve several properties of the mesh: triangles satisfy a minimum angle condition and are as large as the input complex allows. For acute input, this is not always possible: a small angle in the input *must* means a triangle with a small angle must be present in any conforming triangulation. However, a compromise can be reached: a triangulation satisfying a similar relationship between the

size of the triangles and the local feature size with respect to the input can be produced which contains no large angles and thus admits good interpolants for the finite element method (by interpolation results in [2, 22]). This estimate on the size of the resulting triangulation deteriorates as the smallest angle of the input approaches zero. Despite this, algorithms of this form are very effective in practice and are widely used.

The analysis of Delauany refinement algorithms for general three dimensional domains follows the same guiding principle as those in two dimensions: relax requirements on the resulting mesh when necessary to ensure termination of the algorithm. Compared to the elegant theory of Ruppert's algorithm for 2D non-acute domains, these algorithms suffer from two major drawbacks. The first is the issue of case explosion. Ruppert's original proof involves five cases (a few of which are trivial) and is simple to verify. As the algorithm is extended, the number of cases can often double with each modification of the algorithm. To reduce (yet not completely eliminate) this issue, we will focus on isolating the key properties of Ruppert's algorithm and proving analogous results for the simplest possible algorithms. This will be seen to be an effective method for developing a sound Delaunay refinement algorithm without a single monolithic analysis. The result will be to form a sequence of Delaunay refinement algorithms which gradually build the desired output properties and isolate different technical details.

The second uniquely three dimensional issue pertaining to Delaunay refinement is the sliver tetrahedra. A sliver tetrahedra has a poor aspect ratio but a good circumradius-shortest edge ratio: thus it appears acceptable to Delaunay refinement algorithms but is poor for the finite element method. To alleviate the problems associated with slivers, three lines of research have been followed: methods for removing slivers from bounded radius-edge meshes have been shown to produce sliver free meshes based on some theoretical constants [8, 26**?**], experimental approaches have been developed that often remove all slivers but lack rigorous guarantees [18, 24], and the convergence of a simple numerical method in the presence of slivers has been shown [32]. The algorithms we will develop are compatible with each of these programs but do not further these lines of research.

The goal of this thesis is to develop a 3D Delaunay refinement algorithm which

- produces a conforming Delaunay tetrahedralization,
- ensures a bound on the radius-edge ratio of nearby all tetrahedra,
- generates tetrahedra of size related to the local feature size and size of nearby small input angles, and
- is simple enough to admit an implementation.

In the process, we develop an alternative 2D Delaunay refinement algorithm for which our algorithm (and other previous algorithms) is an extension and discuss where the difficulty arises when attempting to extend the best/most used 2D algorithms. This isolates the key obstacles (in addition to the issue of slivers) that tetrahedral mesh generation faces which are not present in triangular mesh generation.

While possibly the most studied feature of Delauany refinement algorithms is the property that the simplices in the meshes generated are not too small, of nearly equal importance is the fact that these simplices are small enough. Ruppert's analysis uses the requirement of mesh quality to ensure that the mesh generated (and any quality triangulation of the input complex) is sufficiently small. In Chapters 3 and 4, we show that the the fact that the resulting mesh is small enough near important parts of the input is a result of the definition of encroachment in the algorithm and does not rely on the quality criteria. This leads to simple Delaunay refinement algorithms which yield practical estimates on the local feature size over the set of $(d-2)$-dimensional features of the input.

In Chapters 5 and 6, Delauany refinement algorithms for generating quality conforming Delaunay triangulations of arbitrary input complexes are given. The estimates in the previous chapters are exactly what is necessary to determine an acceptable size for protected regions near acute input angles which are needed to ensure termination of the algorithm. In 2D, it is possible to avoid explicitly computing a protection size around acute input angles (see [30]), but it is not known how to generalize this approach to 3D. We carefully describe and analyze the simplest Delauanay refinement algorithms in 2D and then extend these approaches to 3D. The result is a practical algorithm which resembles the previously unimplemented algorithms of Cheng and Poon [11] and Pav and Walkington [36]. In the process, we will be required to mesh certain smooth complexes (not just straight segments and planar faces) and this involves a stronger version of a result on curved meshing in 2D.

Finally in Chapter 7, we will consider the mesh generation problem in two dimensions by reconsidering the requirements of the finite element method. In the 1970s, it was shown that the requirement that all simplices have bounded aspect ratio can be relaxed: the minimum angle condition can be replaced by a maximum angle condition (which has also inspired a mesh generation algorithm [31]). We reinterpret standard interpolation estimates to see the relationship between interpolation error (the important quantity in the analysis of finite element methods) and the circumradius of a triangle (the important quantity in the analysis of Delaunay refinement methods). We describe a simple mesh generation algorithm in the spirit of Chew's original Delaunay refinement method [13] which utilizes this relationship.

This simple 2D mesh generation algorithm really highlights the difference between Delaunay refinement in two and three dimensions. In 2D, ensuring the resulting Delaunay triangulation conforms to the input is as simple as initially splitting adjacent input segments at equal length and then splitting segments with non empty diametral balls until termination. Then, interpolation error estimates can be determined in terms of only the lengths of edges in the triangulation, even in the presence of bad angles! In 3D, producing a conforming mesh is quite involved, a bounded radius-edge mesh does not imply the desired interpolation estimates and open questions persist in the characterization of interpolation error over general tetrahedra. The purpose of this thesis is to understand which properties of 2D Delaunay refinement algorithms can be extended to 3D and to investigate the simplest such algorithms which can actually be implemented.

Chapter 2

Delaunay Refinement Preliminaries

2.1 Basic Definitions

Two important types of sets are simplices (triangles in general dimension) and balls (disks in general dimension). Simplices always refer to closed sets while balls are always open sets. The ball of radius r centered at x will be denoted $B(x, r)$. Without ambiguity, given a simplex s, the circumball of s, denoted $B(s)$, is the smallest ball containing all of the vertices of s on its boundary. Similarly, for a curve c with endpoints p_1 and p_2, then $B(c) := B(\overline{p_1 p_2})$.

We will focus on algorithms for generating simplicial meshes which are Delaunay triangulations of point sets in $\mathbb{R}^d$ where d is either 2 or 3. For completeness, the definition of Delaunay triangulation is now given.

Definition 2.1.1. Let $\mathcal{P}$ be a finite subset of $\mathbb{R}^d$.

- The **Delaunay triangulation** of $\mathcal{P}$, denoted $\mathrm{DT}(\mathcal{P})$ refers to the set of simplices with vertices in $\mathcal{P}$ and with circumballs which are disjoint from $\mathcal{P}$.
- Two vertices $p, q \in \mathcal{P}$ are called **Delaunay neighbors** if there is some simplex $t \in \mathrm{DT}(\mathcal{P})$ with vertices p and q.

The term Delaunay tetrahedralization will be used when working specifically in 3D, but Delaunay triangulation will be used when discussing the general case (as well as in 2D situations). From the definition of Delaunay triangulation, the existence of Delaunay neighbors can be characterized by the proposition below.

Proposition 2.1. *[Delaunay Property] Let $\mathcal{P}$ be a finite subset of $\mathbb{R}^d$. Let B be a ball with point $q \in \mathcal{P}$ on the boundary of B. If there is a point of $\mathcal{P}$ inside B, then q has a Delaunay neighbor that is inside B.*

There are many ways to assess the quality of a triangulation. For example, [25] identifies five qualitative properties of simplices (or other polyhedra) and suggests an approach for forming metrics which effectively differentiate simplices based on any combination of these properties. However, we will focus on the metrics described below.

Definition 2.1.2. Let t be a simplex.

- The **aspect ratio** of t, denoted $AR(t)$, is the ratio of the smallest sphere containing t to the largest sphere contained in t.
- The **circumradius-shortest edge ratio**, or **radius-edge ratio**, of t, denoted $\mathrm{RE}(t)$, is the ratio of the circumradius of t to the length of the shortest edge of t.

The standard analysis of the finite element method relies on a uniform bound on the aspect ratio over the triangulation. Analysis of Delaunay refinement algorithms naturally involves the radius-edge ratio. In 2D, aspect ratio and radius-edge ratio are equivalent, while in 3D sliver tetrahedra have good radius-edge ratios but arbitrarily poor aspect ratios.

The mesh generation algorithms discussed will be given some description of the area/volume to be meshed and produce a conforming Delaunay triangulation. We will assume that the input is described as a piecewise linear complex, defined below. Separate definitions are given in two and three dimensions. This makes the simplicity in the 2D case clear and avoids excessive notation which is necessary for a definition in general dimension.

Definition 2.1.3. In two dimensions:

- A **2D piecewise linear complex** (PLC), $\mathcal{C} = (\mathcal{P}, \mathcal{S})$, is a pair of sets of input vertices $\mathcal{P}$ and input segments $\mathcal{S}$, such that the endpoints of each segment of $\mathcal{S}$ are contained in $\mathcal{P}$ and the intersection of any two segments of $\mathcal{S}$ is also contained in $\mathcal{P}$.
- A PLC $\mathcal{C}' = (\mathcal{P}', \mathcal{S}')$ is a **refinement** of the PLC $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ if $\mathcal{P} \subset \mathcal{P}'$ and each segment in $\mathcal{S}$ is the union of segments in $\mathcal{S}'$.
- A PLC $\mathcal{C}^* = (\mathcal{P}^*, \mathcal{S}^*)$ is a **subcomplex** of the PLC $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ if either $\mathcal{C}^* = \mathcal{C}$ or there is a feature $t \in \mathcal{P} \cup \mathcal{S}$ such that
$$\mathcal{P}^* = \{p \in \mathcal{P} \,|\, p \subset t\} \text{ and}$$
$$\mathcal{S}^* = \{s \in \mathcal{S} \,|\, s \subset t\}.$$

Definition 2.1.4. In three dimensions:

- A **3D piecewise linear complex** (PLC), $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$, is a triple of sets of input vertices $\mathcal{P}$, input segments $\mathcal{S}$, and polygonal input faces $\mathcal{F}$ such that the boundary of any feature or the intersection of any two features is the union of other lower-dimensional features in the complex.
- A PLC $\mathcal{C}' = (\mathcal{P}', \mathcal{S}', \mathcal{F}')$ is a **refinement** of the PLC $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ if $\mathcal{P} \subset \mathcal{P}'$ and each segment in $\mathcal{S}$ is the union of segments in$\mathcal{S}'$ and every face in $\mathcal{F}$ is the union of faces in $\mathcal{F}'$.
- A PLC $\mathcal{C}^* = (\mathcal{P}^*, \mathcal{S}^*, \mathcal{F}^*)$ is a **subcomplex** of the PLC $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ if either $\mathcal{C}^* = \mathcal{C}$ or there is a feature $t \in \mathcal{P} \cup \mathcal{S} \cup \mathcal{F}$ such that
$$\mathcal{P}^* = \{p \in \mathcal{P} \,|\, p \subset t\},$$
$$\mathcal{S}^* = \{s \in \mathcal{S} \,|\, s \subset t\}, \text{ and}$$
$$\mathcal{F}^* = \{f \in \mathcal{F} \,|\, f \subset t\}.$$

When refining a PLC, certain simplices near the boundaries of input features have special importance in the analysis. These are defined below.

Definition 2.1.5. Consider a refinement $(\mathcal{P}', \mathcal{S}', \mathcal{F}')$ [or $(\mathcal{P}', \mathcal{S}')$] of an input PLC $(\mathcal{P}, \mathcal{S}, \mathcal{F})$ [or $(\mathcal{P}, \mathcal{S})$].

- An **end segment** is a segment in $\mathcal{S}'$ for which at least one endpoint is an input vertex in $\mathcal{P}$.
- An **end triangle** is a triangle in $\mathcal{F}'$ for which at least one vertex lies on an input segment in $\mathcal{S}$.
- The **spindle** of a segment s in $\mathcal{S}'$, denoted $\text{Spind}(s)$, is the set containing
  - s if s is not an end segment, or
  - s and all end segments adjacent to s if s is an end segment.

For a simplex s, R_s denotes its circumradius. For any point q inserted into the mesh by our refinement algorithms, r_q denotes the insertion radius of point q, i.e. the distance from q to its nearest neighbor in $\mathcal{P}'$ when it is inserted into the Delaunay triangulation.

An appropriate notion of feature size is essential in the analysis of Delaunay refinement algorithms. The standard definition of local feature size is given below as well as another useful sizing function (called mesh feature size).

Definition 2.1.6. Let PLC $\mathcal{C}'$ be a refinement of PLC $\mathcal{C}$.

- The i-**local feature size** at point x with respect to $\mathcal{C}$, $\text{lfs}_i(x, \mathcal{C})$ is the radius of the smallest closed ball centered at x which intersects two *disjoint* features of $\mathcal{C}$ of dimension no greater than i.
- The i-**mesh feature size** at point x with respect to $\mathcal{C}$, $\text{mfs}_i(x, \mathcal{C})$ is the radius of the smallest closed ball centered at x which intersects two features of $\mathcal{C}$ of dimension no greater than i.
- The **nearest neighbor function**, $N(x, \mathcal{P}') := \text{lfs}_0(x, \mathcal{C}')$, returns the distance from x to its second nearest neighbor in $\mathcal{P}'$.

The above definitions do not require any distinction between the input PLC and its refinement. However, we state the definitions in this way as local feature size functions will usually be evaluated with respect to some initial PLC while the nearest neighbor function will be analyzed on the intermediate or resulting triangulations. Figure 2.1 depicts the feature size at the vertex of a mesh during a possible refinement. Note that the feature size is defined at all points in $\mathbb{R}^d$, not just vertices of the mesh. Each of these functions is Lipschitz (with constant 1). For a fixed PLC, local feature size is strictly positive while mesh feature size can equal zero.

If the argument supplied to any of the above feature size functions is a set of points, rather than a point, then the result is defined to the be infimum of the function over the set,

$$\text{lfs}_i(s, \mathcal{C}) := \inf_{x \in s} \text{lfs}_i(x, \mathcal{C}).$$

To simplify the notation in the most common cases, a few conventions will be followed.

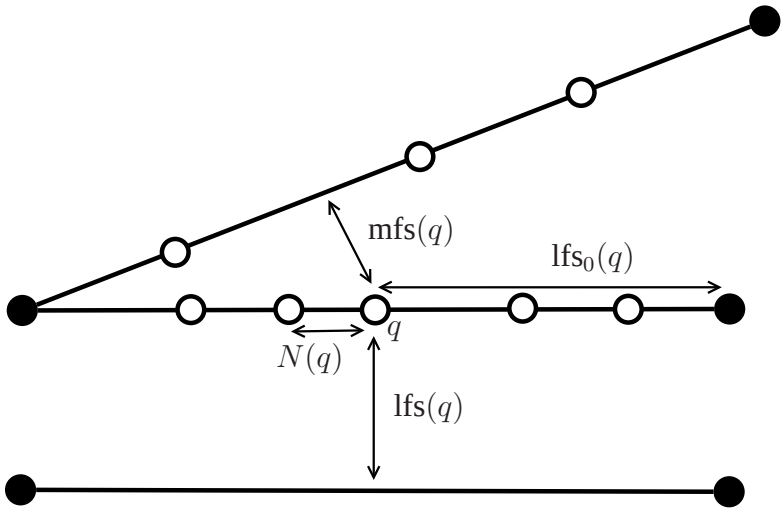


Figure 2.1: Example of sizing functions in Definition 2.1.6 for a 2D PLC. The black points represent input points while the white points represent vertices inserted during the refinement.

Conventions

- If the PLC argument is omitted in the mesh or local feature size function, it is assumed to be the input complex, e.g. $\text{lfs}_i(x) := \text{lfs}_i(x, \mathcal{C})$.
- If the vertex set argument is omitted in the nearest neighbor function, it is assumed to be the vertex set of the current refined complex, e.g. $N(x) := N(x, \mathcal{P}')$.
- If the dimension argument is omitted in the mesh or local feature size function, it is assumed to be $(d-1)$, e.g. $\text{lfs}(x, \mathcal{C}) := \text{lfs}_{d-1}(x, \mathcal{C})$.

Often it will be important to show identical estimates on the local feature size of end segments and the mesh feature size of non-end segments. It is useful to refer to these two cases with the same notation.

Definition 2.1.7. The i-**feature size** of segment s is defined as follows.

$$\text{fs}_i(s) = \begin{cases} \text{lfs}_i(s) & \text{if } s \text{ is an end segment,} \\ \text{mfs}_i(s) & \text{if } s \text{ is a non-end segment.} \end{cases}$$

Given simplex s in $\mathcal{C}'$, point x is called an i-**feature size witness** for s if x is contained in a feature of $\mathcal{C}$ of dimension at most i which is disjoint from s. Given simplex s in $\mathcal{C}'$, point x is called a **local feature size witness** for s if x is contained in a feature of $\mathcal{C}$ which is disjoint from another feature of $\mathcal{C}$ containing s. Simplex s' is called a i-feature size witness for simplex s if every point of s' is an i-feature size witness for s.

The definition of feature size is closely related to the definition of local gap size used by Cheng and Poon [11]. The notion of i-feature size witness is used by recognizing that if x is an

i-feature size witness of segment s, then

$$\text{fs}_i(s) \leq \text{ dist}(x, s).$$

In Chapters 5 and 6, we will consider meshing particular inputs which include curved, rather than straight, features. These complexes belong to the class of piecewise smooth complexes which are defined below.

Definition 2.1.8. A **2D piecewise smooth complex** (PSC), $\mathcal{C} = (\mathcal{P}, \mathcal{S})$, is a pair of sets of input vertices $\mathcal{P}$ and non-self-intersecting smooth input curves $\mathcal{S}$, such that the boundary points of each curve of $\mathcal{S}$ are contained in $\mathcal{P}$ and the intersection of any two curves of $\mathcal{S}$ is also contained in $\mathcal{P}$.

Definition 2.1.9. A **3D piecewise smooth complex** (PSC), $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$, is a triple of sets of input vertices $\mathcal{P}$, non-self-intersecting smooth input curves $\mathcal{S}$, and non-self-intersecting smooth input faces $\mathcal{F}$ such that the boundary of any feature or the intersection of any two features is the union of other lower-dimensional features in the complex.

The definitions of a refinement of a PSC, a subcomplex of a PSC and local/mesh feature size are identical to those given for PLCs. This definition of local feature size does not include any dependence on the curvature or distance to the medial axis of a curve or surface (as is done in [20]). While in general local feature size should include this information, the simpler definition will be sufficient for the simple PSC that will be considered.

The termination and quality guarantees of Delaunay refinement algorithms involve a number of parameter relating to the input PLC and the triangulation produced. The notation for these parameters is given in Table 2.1.

Table 2.1: Important parameters for Delaunay refinement algorithms.

α	The smallest angle between any two adjacent features of the input complex.
α_1	The smallest angle between an input segment and any other adjacent input feature of the input complex.
α_2	(3D only) The smallest angle between any two adjacent input faces.
τ	The radius-edge threshold for the Delaunay refinement algorithm.
κ	(2D only) The minimum angle threshold for the Delaunay refinement algorithm.

2.2 Ruppert's Algorithm

Ruppert's algorithm [39] serves as a prototype for the algorithms we will consider. In the most basic form, the algorithm accepts a non-acute 2D PLC as input and produces a quality, conforming Delaunay triangulation. Quality is achieved in the sense that all triangles satisfy a uniform minimum angle condition via a bound on the radius-edge ratio.

Ruppert's algorithm is given in Algorithm 2.1. In this description, a segment s is called "encroached" if there is any vertex in its diametral ball. A triangle is poor quality if has a radius-edge ratio larger than τ.

Algorithm 2.1 Ruppert's Algorithm

```
Compute the Delaunay triangulation of the input points.
Queue all encroached segments and poor quality Delaunay triangles.
while the queue of simplices is nonempty do
  Pop the front simplex s from the queue.
  if s is a segment then
    Insert the midpoint of s.
  end if
  if s is a triangle then
    Compute the circumcenter c of s.
    if c encroaches upon a segment s' then
      Queue s'.
    else
      Insert c into the Delaunay triangulation.
    end if
  end if
  Remove any queued triangles which no longer exist in the triangulation.
  Queue any newly encroached simplices and poor quality triangles.
end while
```

An important advance in Ruppert's work is the observation that the insertion radius of every vertex added to the mesh is bounded below by the local feature size (which is a strictly positive function). This fact is the key in proving that the size of all resulting triangles in the mesh are proportional to the local feature size of the input.

Theorem 2.2. *If $\alpha \geq \frac{\pi}{2}$ and $\tau > \sqrt{2}$ [or $\sin\kappa < \frac{1}{2\sqrt{2}}$], Ruppert's algorithm terminates. The resulting triangulation conforms to the input, is graded to the local feature size and only contains triangles with radius-edge ratio less than τ [or all angles of resulting triangles are at least κ].*

Remark. This theorem can be extended to allow $\alpha \geq \frac{\pi}{3}$ [42]. The largest known input angle which causes the algorithm to fail is $\alpha = \frac{2\pi}{7}$ [35].

Proof. Termination and grading are implied by the following inequality which is shown inductively at all vertices which are proposed for insertion into the mesh.

$$\text{lfs}(q) \leq \begin{cases} r_q & \text{if } q \text{ is an input point,} \\ C_1 r_q & \text{if } q \text{ is a segment midpoint,} \\ C_2 r_q & \text{if } q \text{ is a circumcenter.} \end{cases}$$

For each of the input points, the inequality above is immediate as

$$\text{lfs}(q) \leq \text{lfs}_0(q) \leq r_q.$$

This provides the base case for the remaining inductive proof.

We will find appropriate constants $C_1 \geq C_2$ such that this inequality holds. Further restrictions on C_1 and C_2 will be addressed throughout the argument. Consider the following cases corresponding to different types of vertices inserted by the algorithm.

Case 1. Let q be a midpoint of some subsegment to be inserted and further that the point encroaching the subsegment is also on a segment. By the requirement that the input be non-acute, the encroaching point is on a disjoint segment. Thus, $\text{lfs}(q) \leq r_q$, so $C_1 \geq 1$ must be required.

Case 2. Let q be the circumcenter of poor quality triangle t which is proposed for insertion. Let p be the newer vertex on the shortest edge of t. Then r_p is at most the length of this short edge which is less than $\frac{|p-q|}{\tau}$, since t is a poor quality triangle. Now, the Lipschitz property of the local feature size can be applied.

$$\begin{aligned}
\text{lfs}(q) &\leq \text{lfs}(p) + |p-q| \\
&\leq C_1 r_q + |p-q| \\
&\leq \left(\frac{C_1}{\tau} + 1\right) |p-q| \\
&\leq \left(\frac{C_1}{\tau} + 1\right) r_q.
\end{aligned}$$

This gives the requirement

$$C_2 \geq \frac{C_1}{\tau} + 1. \tag{2.1}$$

Case 3. Let q be a midpoint of a subsegment which is inserted due to an encroaching (but rejected) circumcenter, c. Local feature size at q is estimated through this point c.

$$\begin{aligned}
\text{lfs}(q) &\leq \text{lfs}(c) + |c-q| \\
&\leq C_2 r_c + |c-q| \\
&\leq \left(C_2\sqrt{2} + 1\right) r_q.
\end{aligned}$$

This gives the requirement

$$C_1 \geq C_2\sqrt{2} + 1. \tag{2.2}$$

Combining the restrictions in the last two cases, it is possible to choose any C_2 large enough to satisfy the below inequality and then choose C_1 as follows.

$$\begin{aligned}
C_2 &\geq \frac{1+\tau}{\tau - \sqrt{2}}, \\
C_1 &:= \sqrt{2}C_2 + 1.
\end{aligned}$$

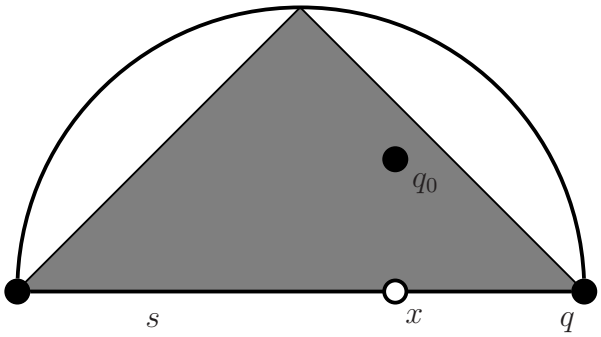


Figure 2.2: Figure for the proof of Theorem 2.3. The shaded region must contain q_0 and is a subset of the diametral disk of s.

These values are only valid when $\tau > \sqrt{2}$. This restriction cannot be lifted without modifying the algorithm and a much more thorough analysis [30].

The algorithm only terminates if no triangles are skinny and all subsegments are not encroached, so we concluded that the resulting mesh conforms to the input and contains no skinny triangles. □

The previous theorem shows that the size of the mesh produced by Ruppert's algorithm is no smaller than (a constant times) the local feature size of the input. It can also be shown that the mesh produced is no larger than (a different constant) times the local feature size of the mesh. The simplest example of a theorem of this type is given below. While this is not the strongest statement which can be shown in this case, it is similar to upcoming results in Chapters 3 and 4.

Theorem 2.3. *If $\alpha \geq \frac{\pi}{2}$ and $\tau > \sqrt{2}$ (or $\sin\kappa < \frac{1}{\sqrt{2}}$), following the termination of Ruppert's algorithm, the following inequality holds for any input vertex q_0 of the mesh:*

$$N(q_0, \mathcal{P}') \leq \sqrt{2}\,\mathrm{lfs}(q_0).$$

Proof. Let q_0 be an input vertex such that $N(q_0, \mathcal{P}') > \sqrt{2}\,\mathrm{lfs}(q_0)$. Since

$$N(q_0, \mathcal{P}') = \mathrm{lfs}_0(q_0, \mathcal{C}') \leq \mathrm{lfs}_0(q_0),$$

the local feature size at q_0 must be realized by an input segment disjoint from q_0. Let $x \in s \in \mathcal{S}'$ be a point on a segment disjoint from q_0 such that $\mathrm{lfs}(q_0) = \mathrm{dist}(q_0, x)$. Further, let $q \in \mathcal{P}'$ be the endpoint of s nearest to q_0 as depicted in Figure 2.2.

$$\begin{aligned} 2\,\mathrm{lfs}(q_0)^2 &\leq N(q_0, \mathcal{P}')^2 \\ &\leq |q_0 - q|^2 \\ &= \mathrm{lfs}(q_0)^2 + |x - q|^2 \end{aligned}$$

Thus $|x - q_0| = \mathrm{lfs}(q_0) \leq |x - q|$ which implies that q lies in the diametral disk of s. Since no segments are encroached when Ruppert's algorithm terminates, conclude that the desired bound holds at termination. □

2.3 The Generic Delaunay Refinement Algorithm

There are several ways to modify and generalize Algorithm 2.1. This is essential to weaken the restriction in Theorem 2.2 to allow acute input. All of the meshing algorithms which we will consider (and effectively all algorithms which can be labeled "Delaunay refinement") match the form of Algorithm 2.2.

Algorithm 2.2 Delaunay Refinement

```
Create an initial Delaunay triangulation.
Queue all unacceptable simplices.
while the queue of simplices is nonempty do
  if it is safe to split the front simplex then
    Take an action based on the front simplex.
    Queue additional unacceptable simplices.
  end if
  Remove the front simplex from the queue.
  Dequeue any queued simplices which no longer exist.
end while
```

To specify an algorithm from Algorithm 2.2, it necessary to describe the following statements.

Action	Where should a vertex (a Steiner point) be inserted to "split" a simplex? Should other (usually lower dimensional) features be queued for splitting?
Priority	In what order should be queue be processed?
Unacceptability	Which simplices are unacceptable?
Safety	Which simplices are safe to split?

Additionally, we require that each of these operations involve only local computations in the Delaunay triangulation of the current point set. In our view, any algorithm which matches the form of Algorithm 2.2 and can be updated based on the local Delaunay triangulation is a Delaunay refinement algorithm and any algorithm that does not fit these to requirements is not.

First, we consider how Ruppert's algorithm (Algorithm 2.1) specializes Algorithm 2.2. The key four descriptions are given in Algorithm 2.3.

Some of these specifications for Ruppert's algorithm are so simple that they can be easily overlooked. However, it is important to generalize Delaunay refinement algorithms in each of the four ways considered above for different purposes. Here is a brief description of how this has been done in the literature. This generic algorithm covers virtually all algorithms which are considered Delaunay refinement.

Algorithm 2.3 Ruppert's Algorithm

Action	If triangle's circumcenter encroaches upon a segment, the encroached upon segment is added to the queue. Otherwise, when a simplex is processed, its circumcenter is inserted.
Priority	Segments are processed before triangles.
Unacceptability	A segment is unacceptable if it has a nonempty diametral disk. A triangle is unacceptable if it has a radius-edge ratio greater than τ.
Safety	It is safe to split any simplex.

2.3.1 Action

There are many different actions that the mesh can take to remove bad simplices. Chew's first Delaunay refinement algorithm [13] used the insertion of circumcenters to remove poor quality triangles from the mesh. Inserting the circumcenter is a natural choice for Delaunay triangulations since this gives the furthest guaranteed distance between the point and any others in the mesh based only on the simplex which is being split. Ruppert's algorithm added the idea of yielding to lower dimensional features. Off-center vertices and general selection regions have also been studied [12, 23, 45] using the same yielding procedure as Ruppert's algorithm. An example of a very different action taken by the algorithm can be seen with Chew's second Delaunay refinement algorithm [14]. This method maintains a constrained Delaunay triangulation, involves a different yielding procedure and occasionally removes vertices from the mesh following certain midpoint insertions. The algorithm of Miller, Hudson, and Phillips includes a yielding procedure in which circumcenters yield to input vertices which have not been inserted into the mesh [21].

2.3.2 Priority

The priority queue for most Delaunay refinement algorithms involves prioritizing lower dimensional simplices before higher dimensional ones [28, 39]. For time efficient algorithms, this priority queue must be modified [1, 21, 27], typically requiring simplices queued for quality to be processed before those queued based on encroachment. Prioritizing queued simplices of equal dimension (often by circumradius) has also been used in some algorithms [27, 38].

2.3.3 Unacceptability

There are typically two types of unacceptability criteria: encroachment criteria which ensure that the required input features exist in the refined mesh, and quality requirements which are desirable of the output mesh. For the encroachment criteria, the most common approach involves asking if a simplex has a nonempty circumball. This is useful since any simplex with a empty circumball must appear in the Delaunay triangulation. Methods which utilize constrained Delaunay triangu-

lations often relax this requirement and consider protecting a smaller lens around each segment or ignore an explicit encroachment criteria all together [4, 14].

The second type of unacceptability criteria, the quality criteria, is usually based on the radius-edge ratio (or the closely related Voronoi quality) of the mesh. Quality also may be specified via a user defined sizing parameter.

2.3.4 Safety

Meshing non-acute domains does not typically require any check that a simplex is safe to split. When handling domains with small angles, typical approaches involve not splitting triangles based on quality if they are near a skinny input angle in some sense [30, 41]. In 3D, the Tetgen code [43, 44] relies on a similar principle for determining when to stop refining near small input angles.

Chapter 3

Estimating Feature Size in 2D

We develop an algorithm for estimating the local feature size of a mesh at input points of a 2D PLC. This estimate is important in order to ensure the termination of quality Delaunay refinement algorithms in the presence of acute angles between adjacent input segments. While there are a number of effective algorithms for quality mesh generation in 2D [30, 41], this algorithm is developed as a natural predecessor to the 3D version given in Chapter 4.

Local feature size is estimated at each input vertex in terms of the distance to its nearest Delaunay neighbor in the resulting triangulation. This is a local quantity in the maintained Delaunay triangulation. The algorithm is very similar to Ruppert's algorithm with two key differences: triangles are not split based on radius-edge quality and certain segments are not split to prevent infinite encroachment sequences near acute angles.

3.1 2D Algorithm

The algorithm for estimating feature size is divided into two steps. These steps are labeled according to the highest dimensional features in the input complex which are considered in the step. Thus Step 0 below only depends upon the input vertices while Step 1a involves input vertices and segments. This convention will be followed with all of the algorithms in the next few chapters.

Algorithm 3.1 Estimate Feature Size 2D

(Step 0) Compute the Delaunay triangulation of the set of input vertices.

(Step 1a) Estimate lfs at all input points via Delaunay refinement.

3.1.1 Step 0

Compute the Delaunay triangulation of the set of input vertices.

Step 0 involves computing the Delaunay triangulation of the set of input points. Following this computation, there is a simple estimate on lfs_0 at each of the input points.

Proposition 3.1. *Following Step 0, for each vertex q_0 in the input PLC, $N(q_0) = \text{lfs}_0(q_0)$.*

3.1.2 Step 1a

Estimate lfs at all input points via Delaunay refinement.

Step 1a of Algorithm 3.1 is a Delaunay refinement algorithm specified by the four rules given in Algorithm 3.2. It is important to recognize that adjacent segments have *not* been split to equal lengths as a preprocess to this algorithm. To ensure termination, a segment s is not allowed to split if the encroaching vertex is on a segment adjacent to s and the resulting segments are shorter than the shortest segment in $\text{Spind}(s)$. This criteria is reflected in the unacceptability rule.

Algorithm 3.2 Estimate Feature Size 2D - Step 1a

Action	Insert the circumcenter of a segment.
Priority	Simplices (only segments in this case) may be processed in any order.
Unacceptability	A segment s is unacceptable if it has an endpoint q with a Delaunay neighbor p inside the diametral disk of s and either p is a 1-feature size witness for s or s is more than twice the length of the shortest segment in $\text{Spind}(s)$.
Safety	It is safe to split any simplex.

First, it is shown that the algorithm terminates and that the distance to the nearest neighbor provides an appropriate upper bound on local feature size in the resulting mesh. This estimate is similar to those shown in Ruppert's analysis.

Theorem 3.2. *Algorithm 3.1 terminates. For any input vertex q_0,*

$$\frac{1}{2}\,\text{lfs}(q_0) \leq N(q_0, \mathcal{P}')$$

holds throughout the algorithm.

Proof. Let $q_0 \in \mathcal{P}$ be any input vertex. Initially, $N(q_0) = \text{lfs}_0(q_0) \geq \text{lfs}(q_0)$ so the base case holds. Suppose a vertex q is inserted in the mesh as the midpoint of segment s and q is the closest neighbor to an input vertex q_0. If s is disjoint from q_0, then $\text{lfs}(q_0) \leq |q_0 - q|$. If s is incident to q_0, then (by the unacceptability rule) the vertex encroaching s, denoted q', must be on a segment which is disjoint from q_0. Thus, $\text{lfs}(q_0) \leq |q_0 - q'| \leq 2|q_0 - q|$. This bound ensures the termination of the algorithm. □

Next, it can be shown that the distance from an input point to its nearest neighbor in the resulting mesh also provides a lower bound on the local feature size.

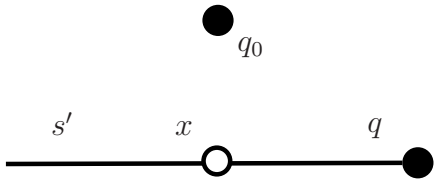


Figure 3.1: Diagram for proof of Theorem 3.3

Theorem 3.3. *Upon the termination of Algorithm 3.1,*

$$N(q_0, \mathcal{P}') \leq \sqrt{2}\,\text{lfs}(q_0)$$

for any input vertex $q_0 \in \mathcal{P}$.

Proof. If $N(q_0, \mathcal{P}) = \text{lfs}(q_0)$ (i.e. the local feature size at q_0 is realized by an input point), then the statement follows by

$$N(q_0, \mathcal{P}') \leq N(q_0, \mathcal{P}) = \text{lfs}(q_0).$$

Otherwise, $\text{lfs}(q_0) = \text{dist}(q_0, s)$ for some segment $s \in \mathcal{S}$ disjoint from q_0 (i.e. the local feature size of q_0 is realized by a segment s). Let x be the nearest point on segment s to q_0. Let $s' \in \mathcal{S}'$ be a subsegment of s containing x and let q be the nearest endpoint of s' to q_0. This situation is depicted in Figure 3.1.

Now, suppose that $N(q_0, \mathcal{P}') > \sqrt{2}\,\text{lfs}(q_0)$. Then the following inequalities hold.

$$\begin{aligned} 2\ \text{lfs}(q_0)^2 &< N(q_0, \mathcal{P}')^2 \\ &< |q_0 - q|^2 \\ &= \text{lfs}(q_0)^2 + |x - q|^2. \end{aligned}$$

Conclude that $|x - q_0| = \text{lfs}(q_0) \leq |x - q|$. This inequality implies that q_0 lies in the diametral disk of s. So either s is unacceptable or q is an input vertex and there is a vertex $p \in \mathcal{P}'$ in $B(\overline{q_0 q}) \setminus B(q_0, |x - q|)$ which lies on a segment adjacent to s since q_0 and q cannot be Delaunay neighbors. The ball $B(q_0, |x - q|)$ must be empty by the assumption on the local feature size of q_0. Thus vertex $p \in B(\overline{q_0 q}) \setminus B(q_0, |x - q|)$ and it follows that $|p - q| \leq |x - q| \leq \frac{|s|}{2}$ as seen in Figure 3.2. Thus $|s|$ is at least double the length of the segment between p and q which is in the spindle of s. So s is unacceptable.

Since upon termination there are no unacceptable segments, the desired bound must hold. □

The constants in Theorem 3.2 and Theorem 3.3 are both sharp and independent of the smallest input angle in the mesh. Note that the inequality in Theorem 3.3 is identical to that in Theorem 2.3 for Ruppert's algorithm in the non-acute case: acute input angle slightly complicate the algorithm but do not weaken the result.

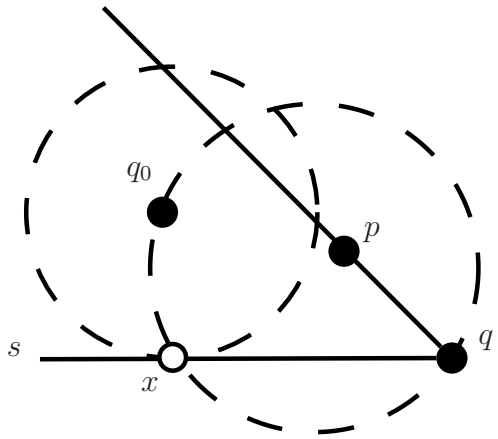


Figure 3.2: If s is an end segment, then $B(q, |x-q|) \bigcap (B(\overline{q_0 q}) \setminus B(q_0, |x-q_0|)) = \emptyset$.

Chapter 4

Estimating Feature Size in 3D

The idea of the previous chapter can be extended to 3D Delaunay refinement. In this case, it becomes important to estimate local feature size and 1-feature size on all segments (the $(d-2)$-dimensional features) of the input complex. While the distance from an input vertex to its nearest neighbor was used to estimate feature size in 2D, the 3D analogy uses the length of segments in the resulting mesh. Following the algorithm, the length of a segment provides a suitable bound of the feature size at any point on the the segment.

4.1 3D Algorithm

Algorithm 4.1 will yield the desired feature size estimates in terms of segment lengths. Step 1b and Step 2b are specific Delaunay refinement algorithms which will be described later. Each of the other steps is a simple procedure which occurs in a single pass over the Delaunay triangulation.

Algorithm 4.1 Estimate Feature Size 3D

(Step 0) Compute the Delaunay tetrahedralization of the set of input vertices.
(Step 1a) Split adjacent segments at equal lengths based on 0-local feature size.
(Step 1b) Estimate fs_1 on all segments via Delaunay refinement.
(Step 2a) Split segments to improve the 1-feature size estimate.
(Step 2b) Estimate lfs on all segments via Delaunay refinement.

The following two theorems demonstrate that in the resulting PLC the length of each segment is a good estimate for the local feature size or 1-feature size of that segment. The first is a lower bound on segment lengths. This theorem will be shown by techniques used in the standard analysis of Ruppert's and other Delaunay refinement algorithms. The second theorem is an upper bound on the lengths of the segments. Theorems of this type have generally not been shown in previous analysis.

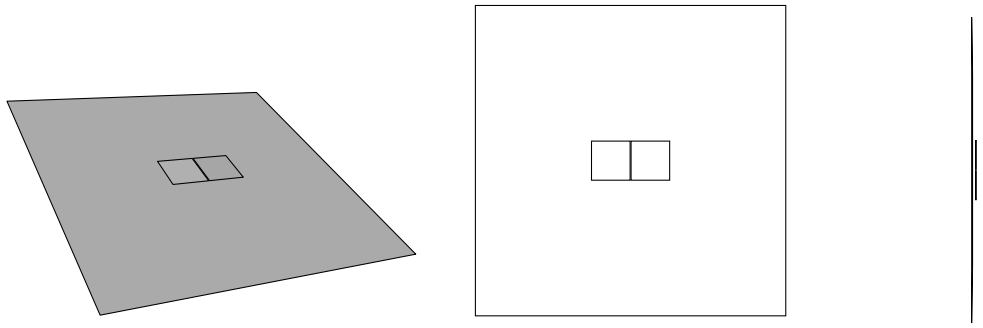


Figure 4.1: This illustrative example consists of 3 faces: one large square which is slightly below two smaller squares which are side by side.

Theorem 4.1. *At any point during Algorithm 4.1, all segments $s \in \mathcal{S}'$ satisfy*

$$\min\left(\frac{1}{16}\,\mathrm{fs}_1(s), \frac{1}{4}\,\mathrm{lfs}(s)\right) \leq |s|.$$

Theorem 4.2. *Following the termination of Algorithm 4.1, all segments $s \in \mathcal{S}'$ satisfy*

$$|s| \leq \min\left(\sqrt{2}\,\mathrm{fs}_1(s), \frac{5}{3}\,\mathrm{lfs}(s)\right).$$

In order to show these two theorems, output conditions on the PLC are determined following each step of the algorithm. Step 2b will yield a mesh satisfying precisely these conditions in the theorems.

We illustrate this algorithm by considering the results of each step on a simple PLC. The example consists of three squares (contained inside a sufficiently large bounding box): one large square which is slightly below two coplanar, disjoint squares as seen in Figure 4.1. Observe that the small feature size between the sides of the two small squares will be realized in Step 1b, while the feature size between the small planes and the large plane will be realized in Step 2b.

4.1.1 Step 0

Compute the Delaunay tetrahedralization of the set of input vertices.

Computing the Delaunay tetrahedralization of the input points is a natural first step in many Delaunay refinement algorithms. Typically, this is done with the Bowyer-Watson algorithm [5, 46], since this uses the routines for incremental insertion of vertices which are necessary throughout the later steps of the algorithm. In our running example, this simply leads to the Delaunay triangulation of each of the squares as seen in Figure 4.2.

Proposition 4.3. *Following Step 0, the following inequalities hold at all input points of the mesh:*

$$\mathrm{lfs}_2(q_0) \leq \mathrm{lfs}_1(q_0) \leq \mathrm{lfs}_0(q_0) = N(q_0, \mathcal{P}').$$

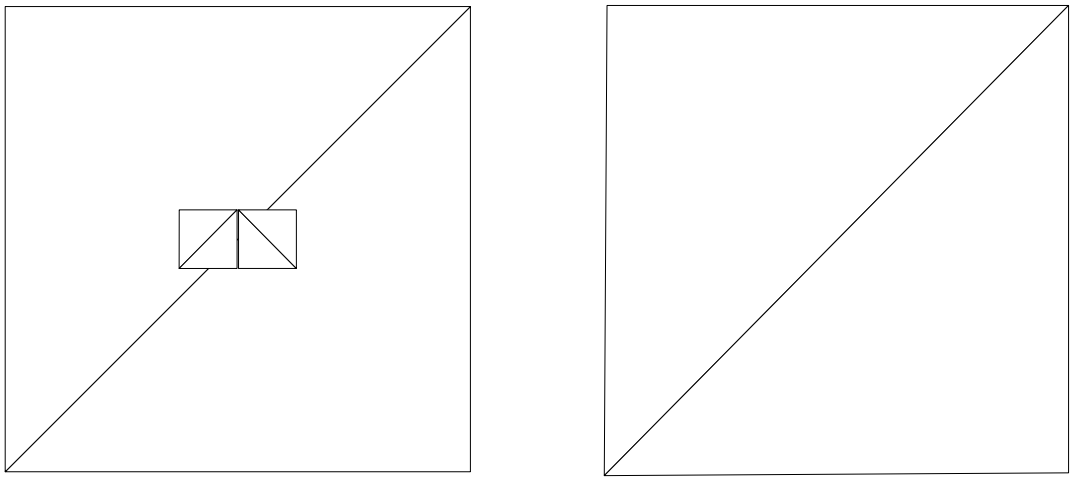


Figure 4.2: (Left) Example mesh following Step 0. (Right) Enlarged mesh of one of the smaller squares.

While this gives a poor run-time in the worst cases, it is common to many Delaunay refinement algorithms, especially those which handle small angles in the input in both two and three dimensions [9, 28, 42, 43, 44].

4.1.2 Step 1a

Split adjacent segments at equal lengths based on 0-local feature size.

This step consists of a single pass of each of the input points. For each input point q_0, all segments containing this point are split at a distance of $\frac{N(q_0,\mathcal{P}')}{3}$ away from q_0. The result of this step on the running example can be seen in Figure 4.3. Notice that small faces are split at a small distance on one side due to the close proximity of their corners to each other.

After completing Step 1a, a number of properties hold which are summarized in the next proposition.

Proposition 4.4. *Following Step 1a, the following hold.*

(I) $N(q_0, \mathcal{P}') = \frac{1}{3}\,\mathrm{lfs}_0(q_0)$ *holds for all input points* $q_0 \in \mathcal{P}$.

(II) Adjacent segments do not encroach each other.

(III) If s_n *is a non-end segment and* s_e *is an adjacent end segment,* $|s_n| \geq |s_e|$.

(IV) If s_e *and* s'_e *are end segments and* $s'_e \notin \mathrm{Spind}(s_e)$*, then* $\mathrm{dist}(s_e, s'_e) \geq \max(|s_e|, |s'_e|)$.

Property IV is particularly important. As segments in the mesh are refined further, this property continues to hold and thus will hold throughout the remaining steps of the algorithm. This ensures that spindles of end segments corresponding to different input point are sufficiently far apart.

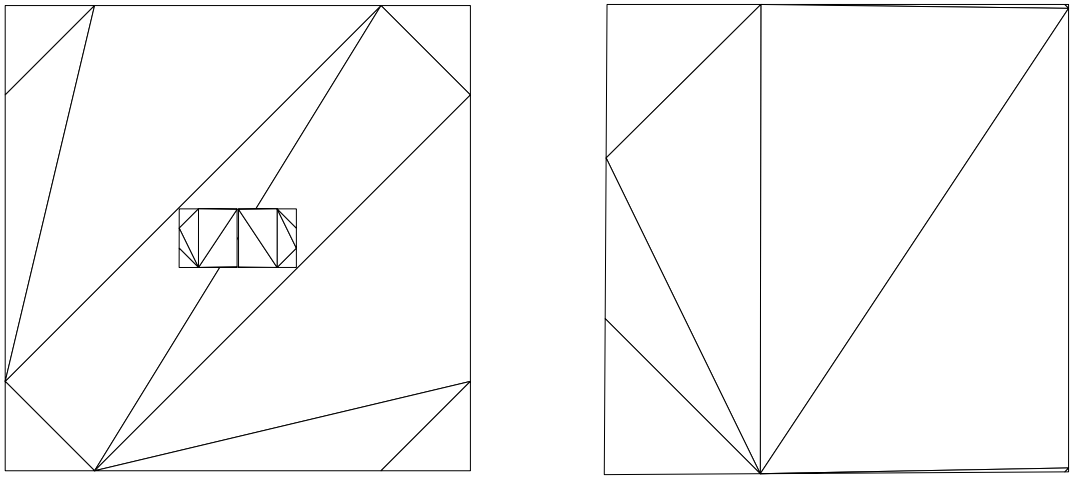


Figure 4.3: (Left) Example mesh following Step 1a. (Right) Enlarged mesh of one of the smaller squares.

4.1.3 Step 1b

Estimate fs_1 on all segments via Delaunay refinement.

The goal of this stage is to bound the length of each segment in the mesh by the 1-feature size of that segment. This occurs via a Delaunay refinement algorithm specified in Algorithm 4.2.

Algorithm 4.2 Estimate Feature Size 3D - Step 1b

Action	Insert the midpoint of a segment.
Priority	Longer segments are processed first.
Unacceptability	Segment s is unacceptable if there is an endpoint q of a segment in $\text{Spind}(s)$ with Delaunay neighbor p such that p is a 1-feature size witness for q and $\lvert q - p \rvert < \lvert s \rvert$.
Safety	It is safe to split any segment.

By the specification given, checking if a simplex is unacceptable requires that only Delaunay neighbors of the endpoints of the segment in question need to be queried. This is an important property of Ruppert's algorithm that we carefully maintain.

Consider the result of applying this step to the earlier example as seen in Figure 4.4. Notice that the main effect is that the nearby edges of the two small squares refine to realize the feature size. The other segments are only split a few times.

First, we show that the algorithm described terminates and that the length of each segment is bounded below by its feature size. This argument uses the same arguments as the "usual" proofs of termination and grading of typical Delauany refinement algorithms.

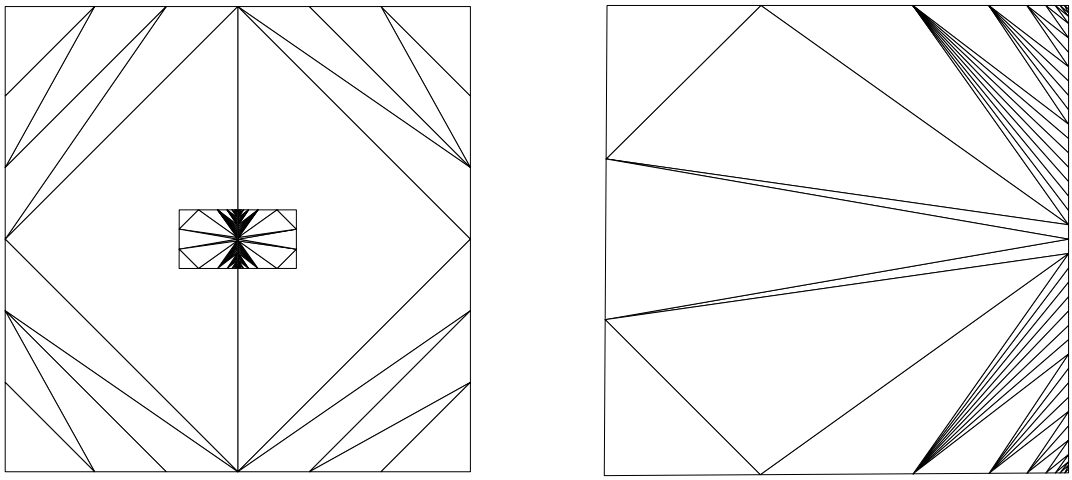


Figure 4.4: (Left) Example mesh following Step 1b. (Right) Enlarged mesh of one of the smaller squares.

Lemma 4.5. *Throughout Step 1b, any segment s in the refinement satisfies*

$$\frac{1}{4}\,\text{fs}_1(s) \leq |s|.$$

Proof. Inductively, we show that the lower bound holds at all segments throughout this step.

Base Case. Following Step 1a, any end segment, s_e, containing input point q_0 has length

$$|s_e| = \frac{1}{3}\,\text{lfs}_0(q_0).$$

The definition of the 1-feature size implies that

$$\text{lfs}_0(q_0) \geq \text{ lfs}_1(q_0) \geq \text{ lfs}_1(s_e) = \text{ fs}_1(s_e).$$

Thus $|s_e| \geq \frac{1}{3}\,\text{fs}_1(s_e)$.

For any initial non-end segment, s_n, there is an adjacent end segment s_e such that $|s_n| \geq |s_e|$. Since s_e contains an input point (which is a 1-feature size witness for s_n), it follows that

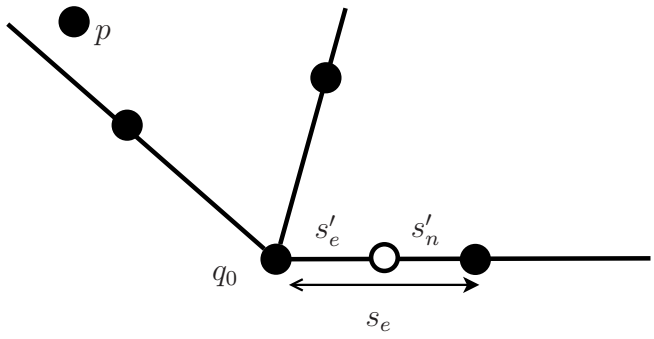
$$|s_n| \geq |s_e| \geq \text{ fs}_1(s_n).$$

Thus, the lower bound on segment lengths holds initially.

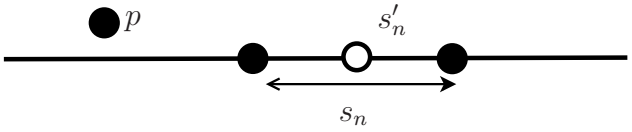
The inductive step is shown in two cases corresponding to the insertion of end segment midpoints and the insertion of non-end segment midpoints. These cases are depicted in Figure 4.5.

Case 1. Consider an end segment s_e from q_0 to q' which is split at midpoint q, forming an new end segment s'_e and a non-end segment s'_n. This means that there is a point p on a disjoint feature to s_e which is of distance at most $|s_e|$ from some adjacent end segment to s_e.

$$\text{fs}_1(s'_e) \leq \text{ dist}(s'_e, p) \leq \text{ dist}(q_0, p) \leq |s_e| + |s_e| = 4|s'_e|.$$



(a) Case 1



(b) Case 2

Figure 4.5: Two Cases in Lemma 4.5.

For the non-end segment s'_n, q_0 is a 1-feature size witness. Observe that

$$\text{fs}_1(s'_n) \leq \text{ dist}(s'_n, q_0) = |s'_n|,$$

so the desired inequality holds.

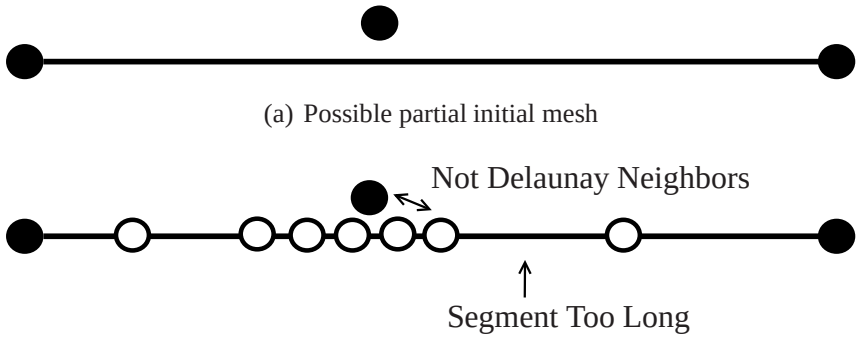
Case 2. Consider non-end segment s_n which is split. This means that there is a feature size witness p such that $\text{dist}(s_n, p) \leq |s_n|$. Then for either of the new end segments created, denoted s'_n,

$$\text{fs}_1(s'_n) \leq \text{ dist}(s'_n, p) \leq |s'_n| + |s_n| = 3|s'_n|.$$

We conclude that $\frac{1}{4}\,\text{fs}_1(s) \leq |s|$ for all segments in the mesh created during Step 1b. This lower bound on feature size of all segments ensures termination of the algorithm. □

Next we seek to bound the length of each segment from *above* in terms of the feature size. In the previous lemma, the ordering of the queue of segments is not necessary. In order to get the upper bound, an arbitrary order does not work. To see this, consider a mesh including a portion similar to Figure 4.6(a). If segments to the left are refined first, a situation similar to Figure 4.6(b) could arise. Then, there is a segment on the right side which is longer than its distance to the input point which is not on the segment. This segment may not see this nearby point on its Delaunay cavity. Note: this requires another point to block the long segment from seeing the nearby disjoint point, but this point could be far away and thus not causing the long segment to split.

By prioritizing the queue by segment length, this situation cannot arise, and it is possible to bound the resulting segments lengths by 1-feature size. In order to prove this, a number of geometric facts are necessary.



(a) Possible partial initial mesh

(b) Possible refinement

Figure 4.6: Lemma 4.9 does not hold without specifying a refinement order.

Proposition 4.6. *Let s and $\bar{s}$ be segments with $|s| \geq |\bar{s}|$. If* $\text{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$*, then there are endpoints p on s and $\bar{p}$ on $\bar{s}$ such that $|p - \bar{p}| < |s|$.*

Proof. Suppose that all pairs of endpoints are such that $|p - \bar{p}| \geq |s|$. The Pythagorean theorem and the fact that $|s| \geq |\bar{s}|$ imply that

$$\text{dist}(p, \bar{s}) \geq \frac{\sqrt{3}}{2}|s|$$

for either endpoint p of s. Again applying the Pythagorean theorem yields that

$$\text{dist}(s, \bar{s}) \geq \frac{|s|}{\sqrt{2}}$$

which completes the proof. □

The constant above is sharp. Consider two skew segments, the first with endpoints $(-1, 0, 0)$ and $(1, 0, 0)$ and the second between $(0, -1, \sqrt{2})$ and $(0, 1, \sqrt{2})$. Then the distance between the two segments of $\sqrt{2}$ and the distance between any pair of endpoints is 2.

The next proposition characterizes a special property which holds when the spindle of an end segment contains segments of equal length.

Proposition 4.7. *Let s_e be an end segment such that*

$$|s_e| = \min_{s' \in \text{Spind}(s_e)} |s'|$$

Let s_n be a non-end segment on an input segment which is adjacent to s_e. If $|s_e| > |s_n|$ and $\text{dist}(s_n, s_e) \leq \frac{|s_n|}{\sqrt{2}}$*, then there are endpoints of s_n and s_e, given by q_n and q_e, respectively, such that $|q_n - q_e| \leq |s_n|$.*

Proof. Let q_0 be the input vertex contained in s_e. Pick $x_e \in s_e$ and $x_n \in s_n$ such that

$$|x_e - x_n| = \text{dist}(s_n, s_e).$$

Since s_n and s_e are coplanar, at least one of the points (either x_n or x_e) is an endpoint of its segment. Since s_n and s_e are not parallel, the choice of x_n and x_e is unique. First, we argue that x_n is an endpoint of s_n. This follows because if not, then the nearest point on s_n is the nearest point on the line containing s_n to x_e. For either endpoint of s_e, the nearest point on this line is at most a distance $|s_e|$ away from q_0. Since s_e is the shortest segment in its spindle, this point cannot be contained in s_n.

So, x_n must be an endpoint of s_n. Since x_n is a vertex, it will be denoted q_n. If x_e is an endpoint of s_e, the desired bound holds since by letting q_e be this endpoint, we observe,

$$|q_e - q_n| = \text{dist}(s_n, s_e) \leq \frac{|s_n|}{\sqrt{2}} < |s_n|.$$

Otherwise, x_e lies in the interior of s_e. Let a,b,c and d denote the distances shown in Figure 4.7(a). Note that $c = \text{dist}(s_n, s_e)$ and $b + d = |s_e|$. Also, $a \geq |s_e|$ and thus

$$\angle q_0 q_e q_n \geq \angle q_0 q_n q_e. \tag{4.1}$$

Moreover, the following inequalities hold.

$$\begin{aligned} |s_e|^2 &\leq a^2 \\ &= b^2 + c^2 \\ &\leq b^2 + \frac{|s_n|^2}{2} \\ &\leq b^2 + \frac{|s_e|^2}{2} \end{aligned}$$

Thus $b^2 \geq \frac{|s_e|^2}{2} \geq c^2$. This means that $\angle q_n q_0 q_e \geq \frac{\pi}{4}$. Combining this with (4.1) implies that

$$\angle q_0 q_e q_n \geq \frac{3\pi}{8} > \frac{\pi}{4}.$$

This means that $c > d$ and thus

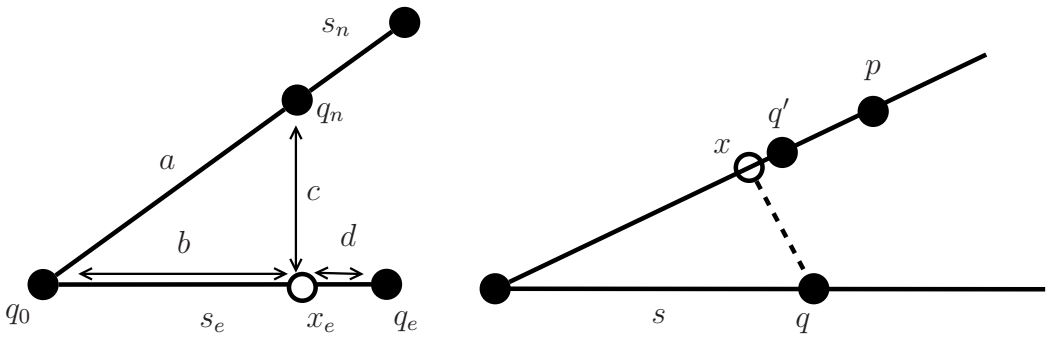
$$|q_e - q_n|^2 = c^2 + d^2 \leq 2c^2 \leq |s_n|^2$$

which completes the proof. □

Proposition 4.8. *Let s be a segment with endpoint q such that*

$$|s| = \min_{s' \in \text{Spind}(s)} |s'|.$$

Let p be a Delaunay neighbor of q such that p is not a feature size witness for s, p is not an endpoint of any segment in the spindle of s, and $|q - p| < |s|$. Then p belongs to a segment s_p such that $|s_p| \leq |q - p|$.



(a) Proposition 4.7 (b) Proposition 4.8

Figure 4.7: Diagrams for the proofs of two propositions.

Proof. If p lies on the same input segment as q, then there clearly exists s_p between p and q such that $|s_p| \leq |p - q|$.

Otherwise, s is an end segment and p lies on an adjacent input segment, denoted s_0. Let x be the nearest point on this adjacent input segment to q. The result holds due to the following sequence of inequalities:

$$|q - p| > |x - p| > |q' - p| \geq |s_p|.$$

See Figure 4.7(b). □

With these facts, it is possible to show the desired bound on segments following Step 1b.

Lemma 4.9. *At the end of Step 1b, all segments satisfy*

$$|s| \leq \sqrt{2}\,\mathrm{fs}_1(s).$$

Proof. The following statement is shown inductively.

Inductive Hypothesis: If segment s is not queued and $|s| > \sqrt{2}\,\mathrm{fs}_1(s)$, then the following two statements hold.

1. If q_0 is an input point, $q_0 \notin s$ and q is an endpoint of s, then $|q - q_0| \geq |s|$.
2. If $\bar{s}$ is a 1-feature size witness for s such that $\mathrm{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$, q is an endpoint of s, and $\bar{q}$ is an endpoint of $\bar{s}$, then $|q - \bar{q}| \geq |s|$.

The split size property below follows from the inductive hypothesis. From this property, it will be clear that the inductive hypothesis is sufficient to imply the inequality in the lemma. Also, when proving the inductive hypothesis, it will be useful to apply the split size property at earlier steps in the algorithm, rather than use the inductive hypothesis directly.

Split Size Property. If the inductive hypothesis holds, any longest segment s such that $|s| > \sqrt{2}\,\mathrm{fs}_1(s)$ is on the queue.

Assuming the inductive hypothesis, we show that the split size property holds. Let s be a segment such that $|s| > \sqrt{2}\,\text{fs}_1(s)$ and s is not on the queue. Then by the first property of the inductive hypothesis, the feature size of s is not realized by an input point. Thus, there exists a segment $\bar{s}$ which is a 1-feature size witness for s and $\text{dist}(s, \bar{s}) = \text{fs}_1(s)$. By Proposition 4.6, $\bar{s}$ is longer than s (otherwise the segments would have endpoints that are nearby, which violates the inductive hypothesis). The inductive hypothesis and Proposition 4.7 imply that s must be a 1-feature size witness for $\bar{s}$ (since otherwise $\bar{s}$ must be an end segment adjacent to the input feature containing non-end segment s). Combining these facts yields

$$|\bar{s}| > |s| > \sqrt{2}\,\text{fs}_1(s) \geq \sqrt{2}\,\text{fs}_1(\bar{s}).$$

We conclude that s is not the longest segment failing the feature size bound.

From the split size property, conclude that if the inductive hypothesis holds, then the upper bound on feature size of segments holds when the algorithm terminates. Thus the above inductive hypothesis is sufficient to imply the lemma. Next, we show that the inductive hypothesis holds in the base case.

Base Case. First consider initial end segments. Let q_0 be an input vertex contained in end segment s_e. Proposition 4.4 ensures that for all other vertices $\bar{q}$ at the end of Step 1a which are not on an end segment adjacent to s_e, $|q_0 - \bar{q}| \geq 2|s_e|$. Thus, the inductive hypothesis holds for all end segments. Next, consider non-end segments. Let s_n be a non-end segment between end segments s_e and s'_e. Any vertex in the mesh which is not an endpoint of s_n is a 1-feature size witness for s_n. If there is another vertex in the mesh which is of distance less than $|s_n|$ to an endpoint of s_n, then s_n must be queued by some 1-feature size witness which is a Delaunay neighbor to an endpoint of s_n.

Next, we proceed to the inductive step. The inductive hypothesis must be checked on all segments. There are two types of segments for which this must be verified: segments that existed before the most recent vertex insertion and segments that where formed by this insertion.

Case 1. Consider any *newly* formed segment s and suppose s violates the inductive hypothesis. If the first criterion of the inductive hypothesis fails, let $\bar{q}$ denote the input point such that $|\bar{q} - q| < |s|$ for some endpoint q of s. Otherwise, the second criterion fails meaning that $|s| > \sqrt{2}\,\text{fs}_1(s)$, s is not on the queue, and (by Proposition 4.6) there is a point $\bar{q}$ and endpoint q of s such that $|q - \bar{q}| < |s|$ and $\bar{q}$ is a feature size witness for s. In either case, $\bar{q}$ is a feature size witness for s and the distance between q and $\bar{q}$ is less than $|s|$. Since s is not queued, this means that q and $\bar{q}$ cannot be Delaunay neighbors.

Now by the Delaunay property there is some point p in $B(\overline{q\bar{q}})$ which is a Delaunay neighbor of q. Again, p cannot be a 1-feature size witness for s, as this would cause s to be queued.

If s is an end segment, notice that no such p can exist. Every non-end segment adjacent to a segment in the spindle of s has length of $|s|$ or $2|s|$, and thus there is no point p on one of these segments at a distance of less than $|s|$ which is not an endpoint of some segment in $\text{Spind}(s)$.

If s is a non-end segment, consider the segment $\hat{s}$ which was split forming s. If $\bar{q}$ is a 1-feature size witness for $\hat{s}$, then $\hat{s}$ fails the desired feature size bound (see Figure 4.8). However vertex p, which was inserted before q, was inserted as the midpoint of a segment of length $2|p-q| < 2|s|$. This violates the split size property (and thus the inductive hypothesis). Otherwise $\hat{s}$ is an end segment and $\bar{q}$ lies on an adjacent input segment (see Figure 4.9). Let q_0 be the input vertex which is an endpoint of $\hat{s}$. In this case, there is a vertex, denoted $\bar{p}$, on the input segment containing $\bar{q}$ such that

$$|q - q_0| = |\bar{p} - q_0|.$$

The diametral ball between q and $\bar{p}$ only intersects line containing segment s in the interior of s. So q has a Delaunay neighbor in this ball and this Delaunay neighbor must be a 1-feature size for s. Thus s is unacceptable.

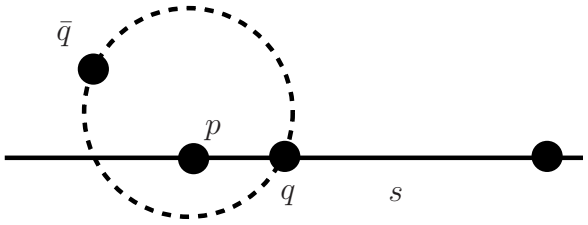


Figure 4.8: Diagram for Case 1 in which s is a non-end segment and p prevents q and $\bar{q}$ from being Delaunay neighbors.

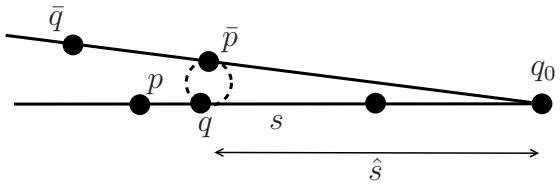


Figure 4.9: Diagram for Case 1 in which s is a non-end segment formed as the result of the split of an end segment.

Case 2. Consider any segment s which is not newly formed. Again we assume this segment s fails the inductive hypothesis and seek a contradiction. The first criteria of the inductive hypothesis cannot fail as the input vertices and endpoints of s did not change in the most recent insertion to the mesh (and thus this statement holds by the inductive hypothesis). So, to fail the inductive hypothesis, s cannot be queued, $|s| > \sqrt{2}\,\mathrm{fs}_1(s)$, and there is a segment $\bar{s}$ which is a 1-feature size witness for s, $\mathrm{dist}(s, \bar{s}) \leq \frac{|s|}{\sqrt{2}}$, and $\bar{s}$ has an endpoint $\bar{q}$ is such that $|\bar{q} - q| < |s|$ for some endpoint q of s. This point $\bar{q}$ must be the most recent point added to the mesh since the inductive hypothesis held at the previous step and thus applies to s.

Let $\hat{s}$ denote the super-segment of $\bar{s}$ which was split by the insertion of $\bar{q}$ and let q' denote the unlabeled endpoint of s. Next, we possibly relabel $\bar{s}$ and q if there is a better selection for our purposes.

(Relabel 1) By possibly relabeling, $\bar{s}$ can be selected to be the subsegment of $\hat{s}$ which is closer to s. This swap can be made because if the original selection of $\bar{s}$ was incorrect, then the closer subsegment also satisfies the same set of necessary properties. Then the nearest point on $\hat{s}$ to s must be in the interior of $\hat{s}$ since $\text{dist}(s, \hat{s}) \leq \frac{|s|}{2}$ while $\text{dist}(q, \hat{s}) > \frac{|s|}{\sqrt{2}}$ and $\text{dist}(q', \hat{s}) > \frac{|s|}{\sqrt{2}}$ (by the inductive hypothesis).

(Relabel 2) Suppose q' is the nearest point on s to $\bar{s}$ and $|q' - \bar{q}| \leq |s|$. Then replace q by q'.

Since s is not on the queue, then q and $\bar{q}$ cannot be Delaunay neighbors. As in Case 1, q must have a Delaunay neighbor in $B(\overline{q\bar{q}})$, denoted p, which cannot be a 1-feature size witness for q. If p is the endpoint of some segment on the spindle of s, replace q with p and s with the segment in $\text{Spind}(s)$ which contains p. This new s must have the same length as the original s as otherwise s would be queued. The Delaunay property can be applied again since the new q and $\bar{q}$ cannot be neighbors. This can be repeated until a point p is found which is not the endpoint of a segment in $\text{Spind}(s)$, lies in $B(\overline{q\bar{q}})$ and is not a 1-feature size witness for s. This configuration is depicted in Figure 4.10. As p cannot be a 1-feature size witness for s, p cannot be an input point and thus p belongs to some segment s_p.

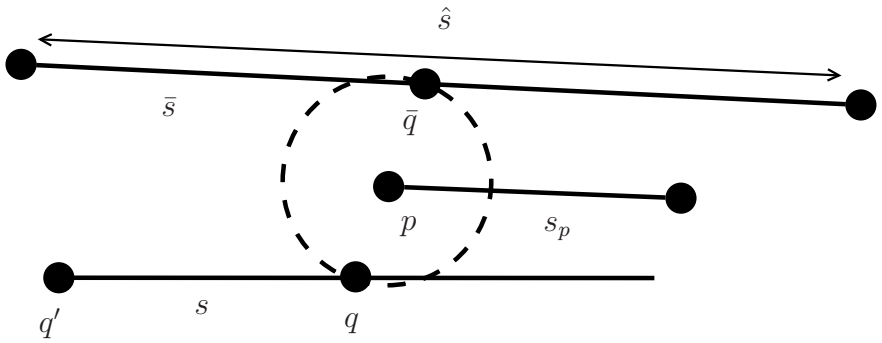


Figure 4.10: Segment s fails the inductive hypothesis, $\bar{q}$ is a nearby feature size witness to s and p is not a 1-feature size witness for s.

Next, we show that s is a 1-feature size witness for $\bar{s}$. If not, s must be a non-end segment on an input feature which is adjacent to $\bar{s}$ (since $\bar{s}$ is a 1-feature size witness for s). In this situation, p cannot exist since it would lie in the end segment adjacent to q (which is in $\text{Spind}(\bar{s})$). See Figure 4.11. Thus s is a 1-feature size witness for $\bar{s}$.

Now, we will utilize the Delaunay property, the split size property and a couple geometric facts to assert the following inequalities:

$$|s| > |q - \bar{q}| > |q - p| > |s_p| \geq |\bar{s}| \geq \frac{|s|}{2}.$$

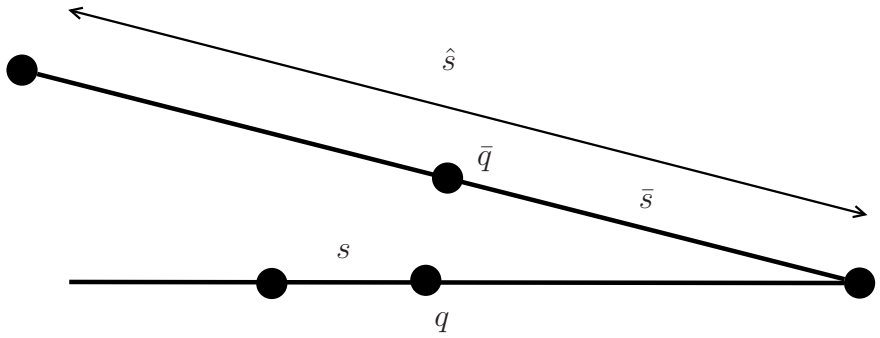


Figure 4.11: If s is a non-end segment and $\bar{s}$ is an end segment, p cannot exists as it must lie in the end segment adjacent to q.

Each of these inequalities is now justified.

(i) $|s| > |q - \bar{q}|$ follows from the assumption that s fails the inductive hypothesis.
(ii) $|q - \bar{q}| > |q - p|$ follows from the construction of p and the Delaunay property.
(iii) $|q - p| > |s_p|$ is a result of Proposition 4.8.
(iv) $|s_p| \geq |\bar{s}|$ follows from the split size property at the time when p was inserted in the mesh.
(v) $|\bar{s}| \geq \frac{|s|}{2}$ is a result of the split size property before $\bar{q}$ is inserted in the mesh.

Finally, a contradiction will be achieved by showing $|\bar{s}| \geq |q - \bar{q}|$ in three different subcases.

Subcase A. Suppose that q is the nearest point on s to $\bar{s}$. Letting $\bar{x}$ be the nearest point on $\bar{s}$ to s as in Figure 4.12(a), observe that

$$\frac{|s|^2}{2} + |\bar{x} - \hat{q}|^2 > |q - \bar{x}|^2 + |\bar{x} - \hat{q}|^2 = |q - \hat{q}|^2 \geq |s|^2.$$

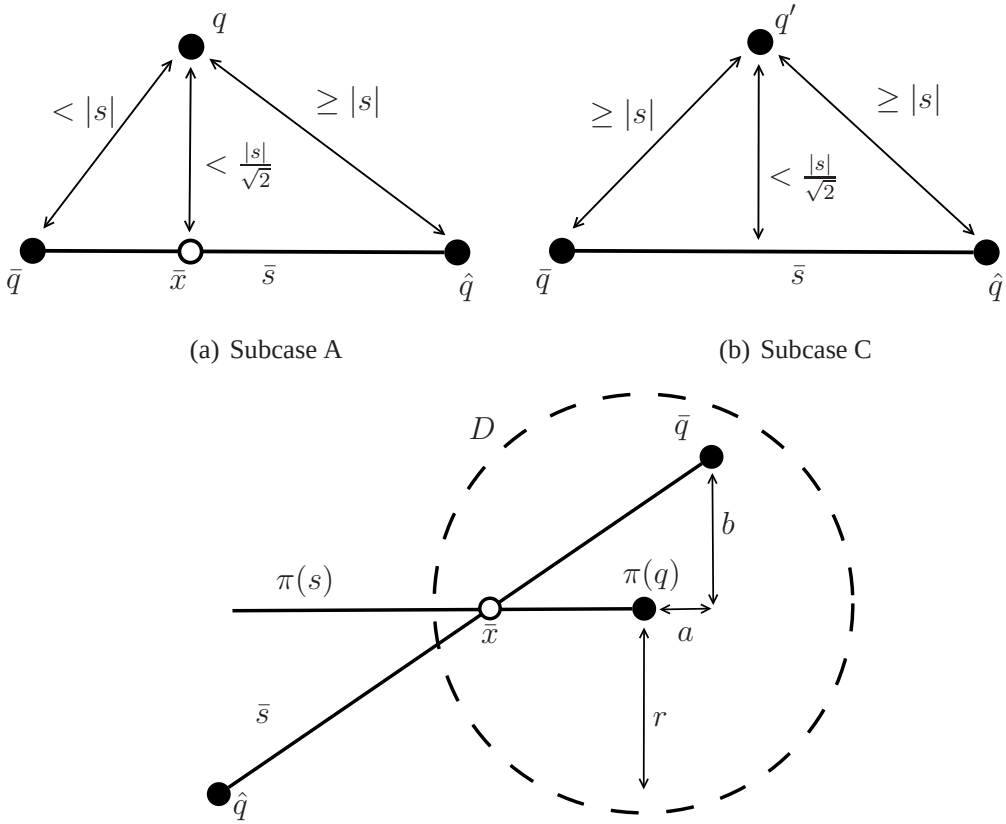
Thus, $|\bar{x} - \hat{q}| > \frac{s}{\sqrt{2}} > |q - \bar{x}|$. See Figure 4.12(a). Then $|\bar{s}|$ can be estimated using the Pythagorean theorem:

$$\begin{aligned} |\bar{s}|^2 &\geq |\bar{q} - \bar{x}|^2 + |\bar{x} - \hat{q}|^2 \\ &> |\bar{q} - \bar{x}|^2 + |q - \bar{x}|^2 \\ &= |q - \bar{q}|^2. \end{aligned}$$

Thus $|\bar{s}| \geq |q - \bar{q}|$.

Subcase B. Let the nearest point on s to $\bar{s}$, denoted x, be in the relative interior of s. In this case, note that the nearest points between the lines containing s and $\bar{s}$ occur in the segments s and $\bar{s}$. This means that $|x - \bar{x}|$ is orthogonal to s and $\bar{s}$.

Let P be the plane containing $\bar{s}$ which is orthogonal to $|x - \bar{x}|$ and let π denote the projection of points into plane P, depicted in Figure 4.12(c). Let r be the radius of the disk $D = P \cap B(q, |s|)$ so $r^2 + |x - \bar{x}|^2 = |s|^2$. Then $\bar{q} \in D$ (by the failure of the inductive hypothesis) and $\hat{q} \notin D$



(a) Subcase A

(b) Subcase C

(c) Subcase B

Figure 4.12: Diagram for Case 2 of Lemma 4.9

(since $|\bar{s}| \geq |s|$). For an appropriate point p to exist, $(x-q)\cdot(\bar{q}-\pi(q)) < 0$. If s is a non-end segment, this is a result of the fact that p cannot lie in the interior of s and this does not lie in $\pi(s)$. In the case of an end segment, q' is an input point and the distance from q' to p must be at least $\frac{3}{2}|s|$, since $|s_p| \geq \frac{|s|}{2}$. The law of cosines gives that $\cos(\angle p\pi(q)\pi(q')) \leq -\frac{1}{4}$ and thus $(x-q)\cdot(\bar{q}-\pi(q)) < 0$ holds.

Let a be the length of the component of $\bar{q}-q$ in the direction of s and let b be the length of the component of $q-\bar{q}$ which is orthogonal to both s and $x-\bar{x}$ as seen in Figure 4.12(c). This gives the following sequence of inequalities:

$$\begin{aligned} |\bar{s}|^2 = |\bar{q}-\hat{q}|^2 &\geq (r+a)^2 + b^2 \\ &> r^2 + a^2 + b^2 \\ &\geq \frac{|s|^2}{2} + a^2 + b^2 \\ &\geq |x-\bar{x}|^2 + a^2 + b^2 \\ &= |q-\bar{q}|^2. \end{aligned}$$

Thus we have achieved the desired inequality, $|\bar{s}| > |q-\bar{q}|$.

Subcase C. Suppose that q' is the nearest point on s to $\bar{s}$ as depicted in Figure 4.12(b). By (Relabel 2), we can conclude that the distance from each of the endpoints of $\bar{s}$ to q' is at least $|s|$. The minimum distance from q' to $\bar{s}$ is less than $\frac{|s|}{\sqrt{2}}$, and so $|\bar{s}| > |s|$. Combining with inequality (i), this implies that $|\bar{s}| > |q-\bar{q}|$.

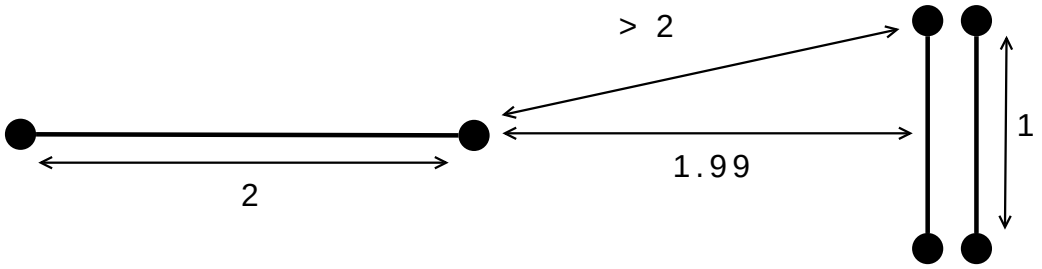
The inequality $|\bar{s}| > |q-\bar{q}|$ holds in each of the three cases, and thus a contradiction has been reached in each case. Conclude that the inductive hypothesis does hold and the lemma follows. □

The estimates in the Lemma 4.17 will prove essential in the later steps. The current refinement ensures that the length of each segment is a good estimate (up to a factor of $4\sqrt{2}$) of the 1-feature size on the segment.

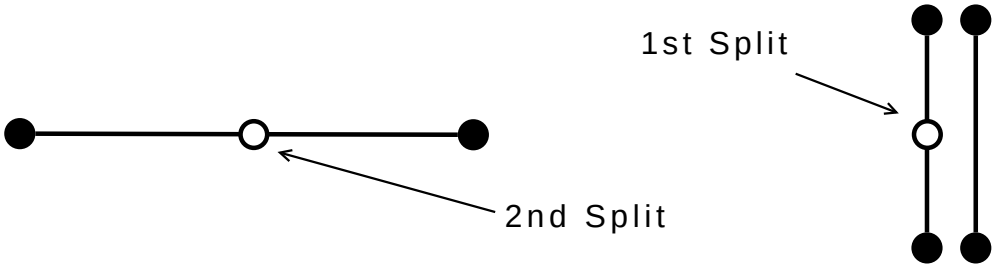
The proof of the theorem in this step relies on an inductive hypothesis which implies that the longest segment s such that $|s| > \sqrt{2}\,\mathrm{fs}_1(s)$ is always on the queue. So if s is such that $|s| > \sqrt{2}\,\mathrm{fs}_1(s)$ and s is not on the queue, then at all previous steps, any segment split had length of at least $|s|$.

Naturally, the proofs would be much simpler if it could be shown that the length of segments being split formed a nonincreasing sequence. Unfortunately, this is not the case. Consider a mesh as outlined in Figure 4.13(a). Notice that the length two segment is not queued initially, as all points are sufficiently far from its endpoints. When the leftmost of the length one segments is split, this midpoint causes the longer length two segment to be queued. See Figure 4.13(b).

While the length of segments which are split increases, this example does not break the inductive hypothesis! It is important to notice in this case that the initial long segment (of length two which will be denoted by s) has a feature size of 1.99 meaning that initially, $|s| < \sqrt{2}\,\mathrm{fs}_1(s)$.



(a) Initial Mesh



(b) Refined Mesh

Figure 4.13: Sequence of split segment lengths is not monotone.

4.1.4 Step 2a

Split segments to improve the 1-feature size estimate.

This step is a simple operation: split all segments into fourths. This is needed as the feature size bound on the segment lengths found in the previous section is not quite strong enough for the algorithm in the next step. Figure 4.14 shows the result of this step on our initial example.

This additional refinement strengthens to the bound determined in Step 1b which will be needed in the analysis of Step 2b. The stronger estimate that will be used is given in the following lemma.

Lemma 4.10. *Following Step 2a, for any segment s in the mesh satisfies*

$$|s| \leq \frac{1}{2\sqrt{2}} \operatorname{lfs}_1(s).$$

Proof. Following Step 1b, $|\hat{s}| \leq \sqrt{2} \operatorname{fs}_1(\hat{s}) \leq \sqrt{2} \operatorname{lfs}_1(\hat{s})$, for all segments $\hat{s}$. If s is a subsegment of $\hat{s}$, then $\operatorname{lfs}_i(\hat{s}) \leq \operatorname{lfs}_i(s)$. Now let s be one of the four segments created during this step from segment $\hat{s}$. Then,

$$\begin{aligned} |s| &= \frac{1}{4}|\hat{s}| \\ &\leq \frac{\sqrt{2}}{4} \operatorname{lfs}_1(\hat{s}) \\ &\leq \frac{1}{2\sqrt{2}} \operatorname{lfs}_1(s). \end{aligned}$$

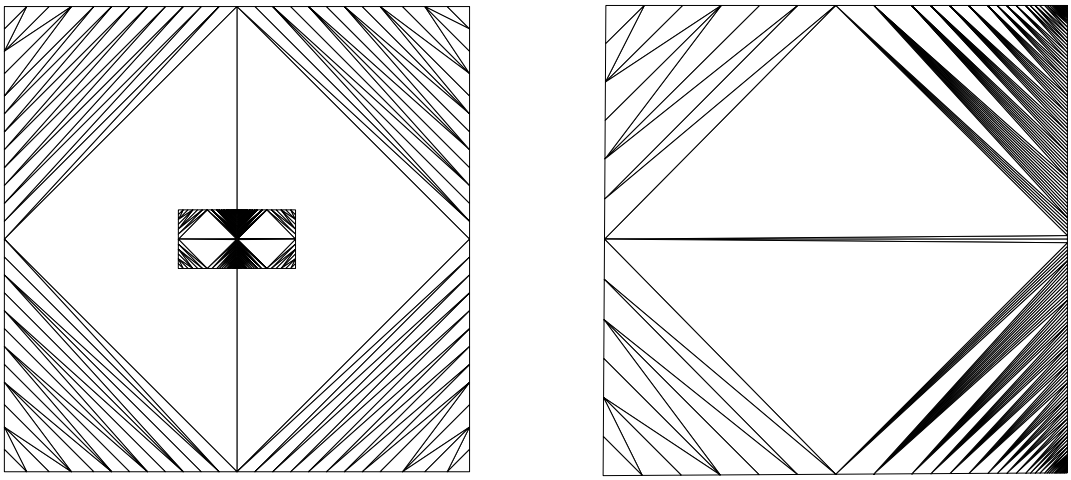


Figure 4.14: Example mesh following Step 2a.

□

Note that the estimate $|s| \leq \frac{1}{2\sqrt{2}} \operatorname{fs}_1(s)$ may *not* hold for some segments in the mesh produced during Step 2a. This is due to the fact that when end segments are split, newly formed non-end segments may have 1-feature size which is much smaller than the 1-feature size of the original end segment .

The next lemma is the natural successor to Lemma 4.5.

Lemma 4.11. *Following Step 2a, for any segment s in the mesh,*

$$\frac{1}{16} \operatorname{fs}_1(s) \leq |s|.$$

Proof. Let s be a subsegment of some segment $\hat{s}$ which existed at the end of Step 1b. It follows that

$$\operatorname{fs}_1(s) \leq \operatorname{fs}_1(\hat{s}) \leq 4|\hat{s}| = 16|s|$$

and the lemma holds. □

4.1.5 Step 2b

Estimate lfs on all segments via Delaunay refinement.

In this step, segments and triangles (in the current Delaunay triangulation of the faces) are split to estimate the local feature size on the segments. This is performed via a Delaunay refinement algorithm which is given in Algorithm 4.3.

The priority rule in this algorithm is backwards from the standard approach in Delaunay refinement: higher dimensional simplices are processed first. This fact will be used in the proof

Algorithm 4.3 Estimate Feature Size 2D - Step 2b

Action	Insert the circumcenter of a proposed segment or triangle.
Priority	Triangles are given highest priority, in any order. Segments are then prioritized by length.
Unacceptability	A segment s is unacceptable if it has an endpoint q with a Delaunay neighbor p such that $\|q-p\| < \|s\|$ and either p is a local feature size witness for s or p is a 1-feature size witness for s. A triangle t is unacceptable if it has a vertex q with Delaunay neighbor p such that $\|p-q\| < 2R_t$ and p does not lie in the face containing t.
Safety	It is not safe to split a triangle in face f if its circumcenter c will have a Delaunay neighbor q which is the endpoint of a segment s in face f and $\|c-q\| < \|s\|$.

but it is mainly used to simplify the arguments. It is likely that the same (or very similar) results hold using a more traditional priority queue.

The mesh resulting from Step 2b in our running example is given in Figure 4.15. Notice that this is the first step in which points are added in faces rather than just on segments.

First, the lower bound on segment length is shown.

Lemma 4.12. *Throughout Step 2b, the following estimate holds for any segment s:*

$$\min\left(\frac{1}{16}\,\mathrm{fs}_1(s), \frac{1}{4}\,\mathrm{lfs}(s)\right) \leq |s|.$$

Proof. This lemma is shown by induction. Lemma 4.11 implies that $|s| \geq \frac{1}{16}\,\mathrm{fs}_1(s)$ holds for all initial segments, so the base case holds. It must be shown that any new segment s which is the result of a split also satisfies the bound. A segment is only split if it is queued and segments are only queued if there is a nearby 1-feature size witness or local feature size witness. In the first case, an identical argument to that in Lemma 4.5 implies that $\frac{1}{4}\,\mathrm{fs}_1(s) \leq |s|$. In the second case, a very similar argument yields $\frac{1}{4}\,\mathrm{lfs}(s) \leq |s|$. □

The proof that segment lengths will bound local feature size from below requires a number of geometric facts. These are stated first.

Proposition 4.13. *Let t be a triangle, and let $x \in t$. Then there is a vertex q_t of t such that $|x - q_t| \leq R_t$.*

Proof. Let c_t be the circumcenter of triangle t. Observe that t is covered by the three diametral balls between each vertex and the circumcenter. See Figure 4.16. □

The next proposition ensures that if the nearest point on a face to a segment is in the interior of a face, then the nearest point on the segment to that face occurs as an endpoint of the segment.

Proposition 4.14. *Let s be a segment in a PLC. Then one of the following holds.*

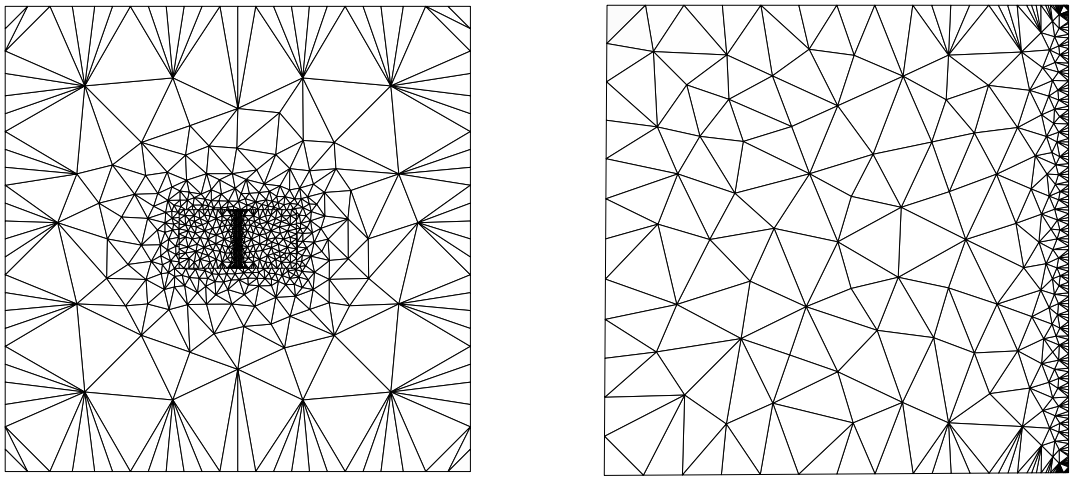


Figure 4.15: (Left) Example mesh following Step 2b. (Right) Enlarged mesh of one of the smaller squares.

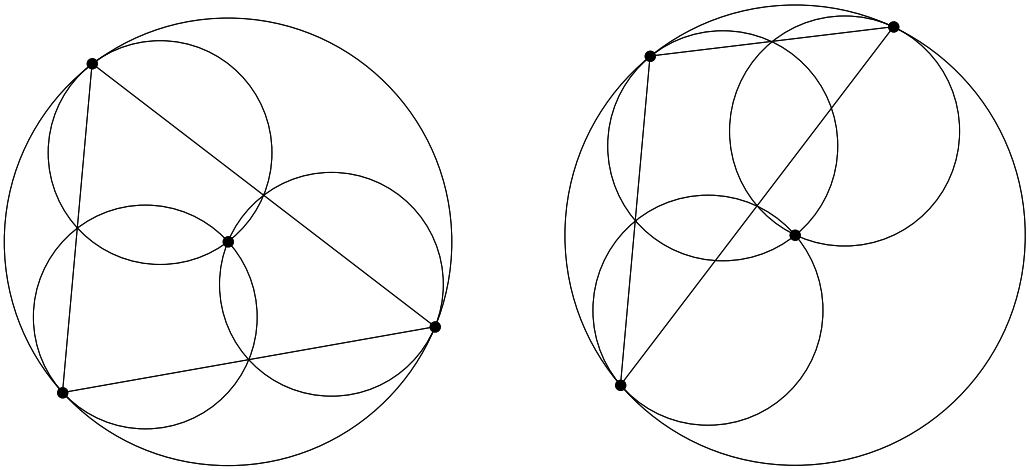


Figure 4.16: Given any triangle, the three diametral balls between vertices and the circumcenter cover the triangle.

- $\text{lfs}(s) = \text{lfs}_1(s)$.
- *There is some endpoint q of s and x in the interior of a disjoint face such that*

$$\text{dist}(q, x) = \text{lfs}(s),$$

and $\overline{qx}$ is orthogonal to the face containing x.

Proof. Suppose $\text{lfs}(s) \neq \text{lfs}_1(s)$. Then there exists a face f and point $x \in f$ such that f is disjoint from s and $\text{lfs}(s) = |x - y|$ for some $y \in s$. For any such f, x and y, $x \notin \partial f$ since then x would be contained in a segment of the PLC which is disjoint from s and thus x would be a witness that $\text{lfs}(s) = \text{lfs}_1(s)$. This means that $x - y$ is orthogonal to the face f. Further, either $x - y$ is orthogonal to s or s is parallel to some vector in the face f. In the former case, the proposition has been shown. In the latter, let P be the plane containing f. Then

$$\min_{x \in P} |x - y| = \min_{x \in P} |x - y_1|$$

holds for any $y, y_1 \in s$. Since

$$\arg\min_{x \in P} |x - y| \notin \partial f$$

for any $y \in s$, conclude that $\arg\min_{x \in P} |x - y| \in f$ for all $y \in s$. Thus y can be selected as an endpoint of s which completes the proof. □

The next proposition asserts a minimum circumradius on the Delaunay triangle containing a point x given that x belongs to an empty disk in the face.

Proposition 4.15. *Consider a set of* coplanar *vertices $\mathcal{P}$. Suppose ball $B(x_0, R)$ contains no vertices of $\mathcal{P}$. Consider $x \in B(x_0, R)$ and x in the convex hull of $\mathcal{P}$. Let t be a triangle in the Delaunay triangulation of $\mathcal{P}$ containing x. Then*

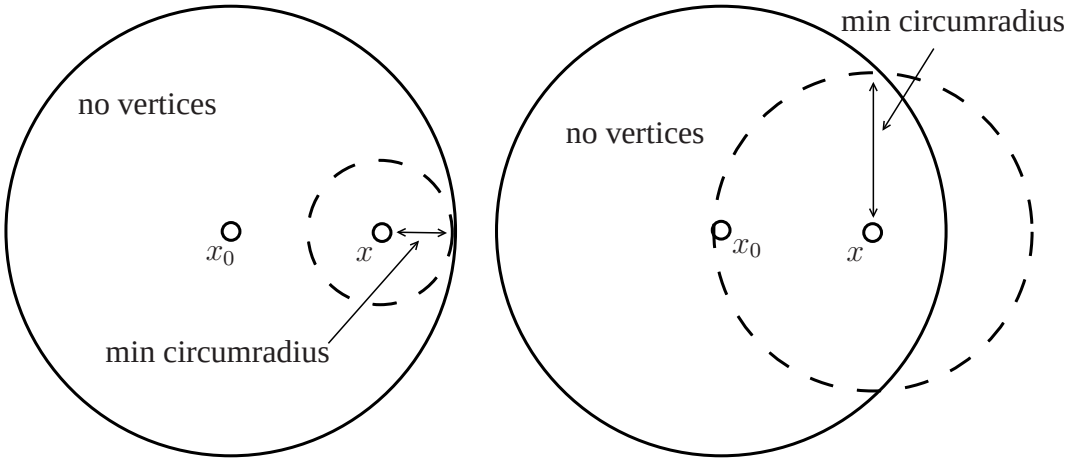
$$R_t \geq \sqrt{R^2 - |x - x_0|^2}.$$

Using the fact that $B(x, R - |x - x_0|) \subset B(x_0, R)$ only ensures that $R_t \geq R - |x - x_0|$. This bound will hold whenever x is in the circumdisk of t. The stronger bound comes from the fact that x is actually inside triangle t. This is depicted in Figure 4.17.

Proof. If $B(x_0, R) \subset B(t)$ or $B(x_0, R) = B(t)$, then $R_t \geq R$ and the result follows. Next, no triangle t exists such that $B(t) \subset B(x_0, R)$ since then $\partial B(t) \setminus B(x_0, R)$ contains at most one point and $B(x_0, R)$ contains no vertices of $\mathcal{P}$. In the remaining case, both $B(x_0, R) \setminus B(t)$ and $B(t) \setminus B(x_0, R)$ are nonempty. Let $\{p_1, p_2\} = \partial B(x, R') \cap \partial B(x_0, R)$ and let $s = \overline{p_1 p_2}$. We consider two cases depicted in Figure 4.18.

Case 1. s lies between x and x_0. Then by the Pythagorean theorem,

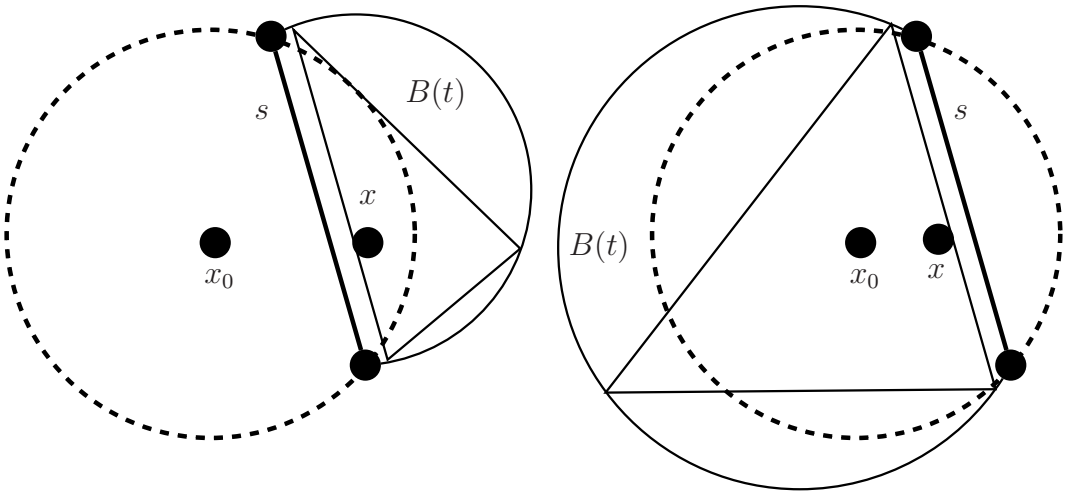
$$R^2 = \text{dist}(x_0, s)^2 + \frac{|s|^2}{4}.$$



(a) If point x is *contained in the circumcircle* of (Delaunay) triangle t and lies in disk which contains no vertices, then the circumradius of t is at least the distance from x to the boundary of the empty disk.

(b) If point x is *contained in* (Delaunay) triangle t and lies in an empty disk, then the lower bound on the circumradius of t is larger than the distance from x to the boundary of the empty disk.

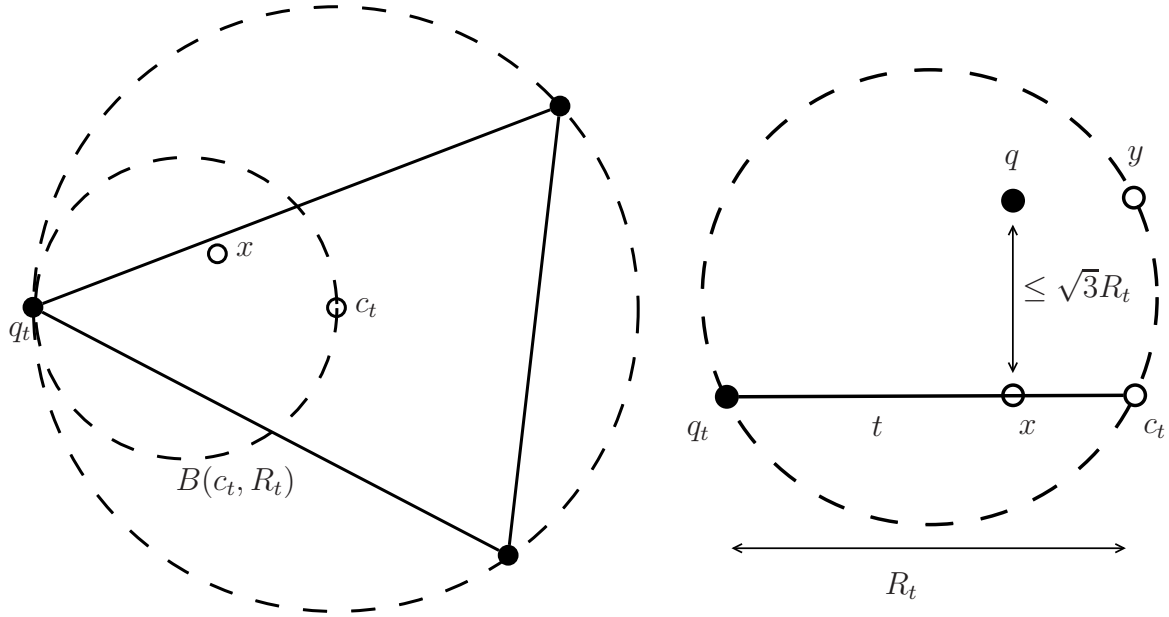
Figure 4.17: Diagram for Proposition 4.15.



(a) Case 1.

(b) Case 2.

Figure 4.18: Two cases in the proof of Proposition 4.15.



(a) Top view (b) Side view

Figure 4.19: Triangle t in Proposition 4.16

Applying the fact that $\text{dist}(x_0, s) \leq |x - x_0|$ leads to the inequality

$$R_t \geq \frac{|s|}{2} \geq \sqrt{R^2 - |x - x_0|^2}.$$

Case 2. s does not lie between x and x_0. Let L be the line which is parallel to s and passes through x_0. In this case, observe that

$$L \cap B(x_0, R) \subset B(t).$$

Thus $R_t \geq R \geq \sqrt{R^2 - |x - x_0|^2}$. □

Suppose that a vertex q is near a face f in some sense. Letting x be the projection of q onto the plane containing f, the next lemma ensures that if the triangle t (in the Delaunay triangulation of f) containing x is large enough, then one of the vertices of t has a nearby Delaunay neighbor (in the 3D Delaunay tetrahedralization) which is not in the face. In the algorithm, this will ensure that t was placed on the queue.

Proposition 4.16. *Let t be a Delaunay triangle in a face f. Let q be a vertex which is not in the face such that the nearest point to q on the plane containing t, denoted x, lies in t. If $|q - x| < \sqrt{3}R_t$, then there is a vertex of t, q_t, which has a Delaunay neighbor, p, such that p is not in the face containing t and $|q_t - p| < 2R_t$.*

Proof. Let q, t, x be as in the statement of the proposition. Let c_t be the circumcenter of t. Since x lies in t, by Proposition 4.13, there is a vertex of t, denoted q_t, such that $|x - q_t| \leq R_t$. See Figure 4.19. Let $y = c_t + q - x$. Then observe the following properties.

- $\partial B(\overline{q_t y}) \cap f$ is the diametral circle between c_t and q_t.
- $q \in B(\overline{q_t y})$.

Finally, applying Proposition 2.1, it follows that q_t must have a Delaunay neighbor p in $B(\overline{q_t y})$, and thus $|q_t - p| \leq |q_t - y| \leq 2R_t$. Moreover, p cannot be in the face f since the diametral circle of c_t and q_t must be empty since t is a Delaunay triangle in the face. □

With these geometric facts in place, we seek the bounds in Theorem 4.2, which are given in two lemmas.

Lemma 4.17. *Upon termination of Step 2b, each segment s satisfies*

$$|s| \leq \frac{5}{3} \operatorname{lfs}(s).$$

Proof. This inequality is shown by induction. Specifically, we show the following inductive hypothesis.

Inductive Hypothesis Let s be a segment such that $|s| > \frac{5}{3} \operatorname{lfs}(s)$. If s is not on the queue then there is some triangle t which is on the queue.

First, if the inductive hypothesis holds, then the desired bound holds at termination since whenever the desired bound fails, the queue is not empty. Next, suppose that s is some segment such that $|s| > \frac{5}{3} \operatorname{lfs}(s)$ and s is not queued. We will show that this implies that some triangle must be on the queue.

As edges have already been isolated from each other, the witness of the local feature size of s must be a face. Following Step 2a, $|s| \leq \frac{\operatorname{lfs}_1(s)}{2\sqrt{2}}$. Since splitting a segment decreases its length and cannot increase its local feature size, this bound will hold on all segments throughout the step. This ensures that no segment or input point can be the witness to the local feature size of s. Thus, there must be some face f such that $\operatorname{dist}(s, f) = \operatorname{lfs}(s)$. Using Proposition 4.14, conclude that there is some x in a face f and an endpoint q of s such that $\operatorname{lfs}(s) = |x - q|$, and the vector $q - x$ is orthogonal to the plane containing f.

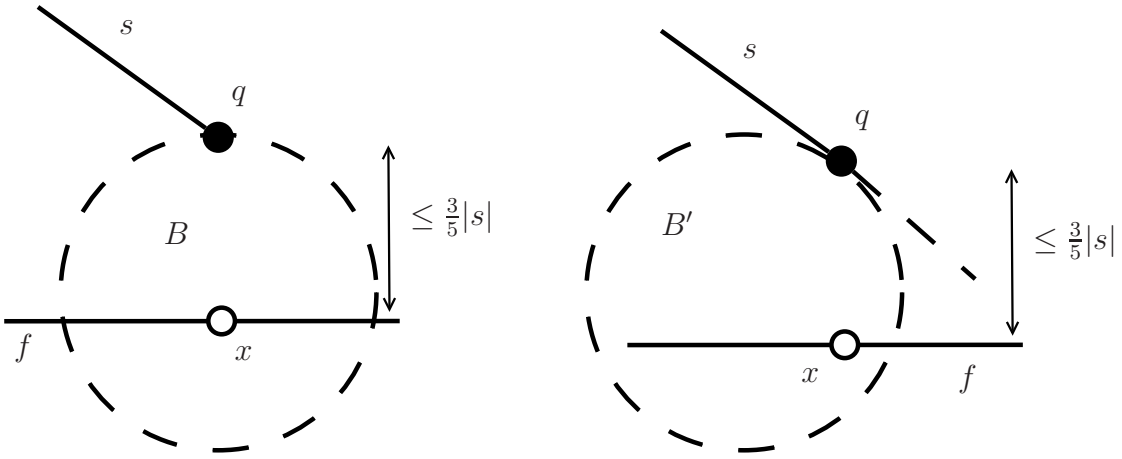
Let L denote the line containing s, P denote the plane containing f and π denote the projection function into P. Suppose there is a segment $s' \in \operatorname{Spind}(s)$ with endpoint q' which is closer to P than q. First, estimate the distance from x to the boundary of f, ∂f:

$$\begin{aligned} \operatorname{dist}(x, \partial f)^2 &= \operatorname{dist}(q, \partial f)^2 - |x - q|^2 \\ &\geq 8|s|^2 - \frac{9}{25}|s|^2. \end{aligned}$$

So $\operatorname{dist}(x, \partial f) \geq 2|s|$. Considering any point $p \in s'$,

$$|\pi(q') - x| \leq |q' - q| \leq 2|s|.$$

Thus $\pi(q') \in f$. This means that s and q can be replaced with s' and q' and the local feature size bound still fails. Thus without loss of generality, assume that s is the nearest segment in $\operatorname{Spind}(s)$ to P.



(a) Case 1 (b) Case 2

Figure 4.20: Delaunay neighbors to point q are considered in different balls in the the two different cases.

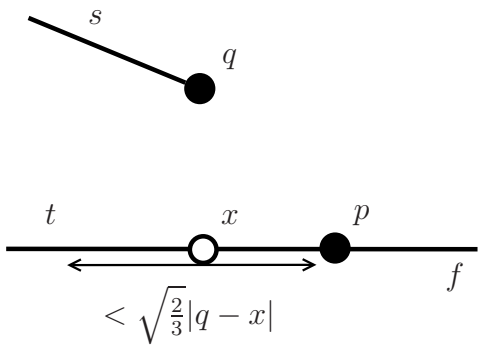


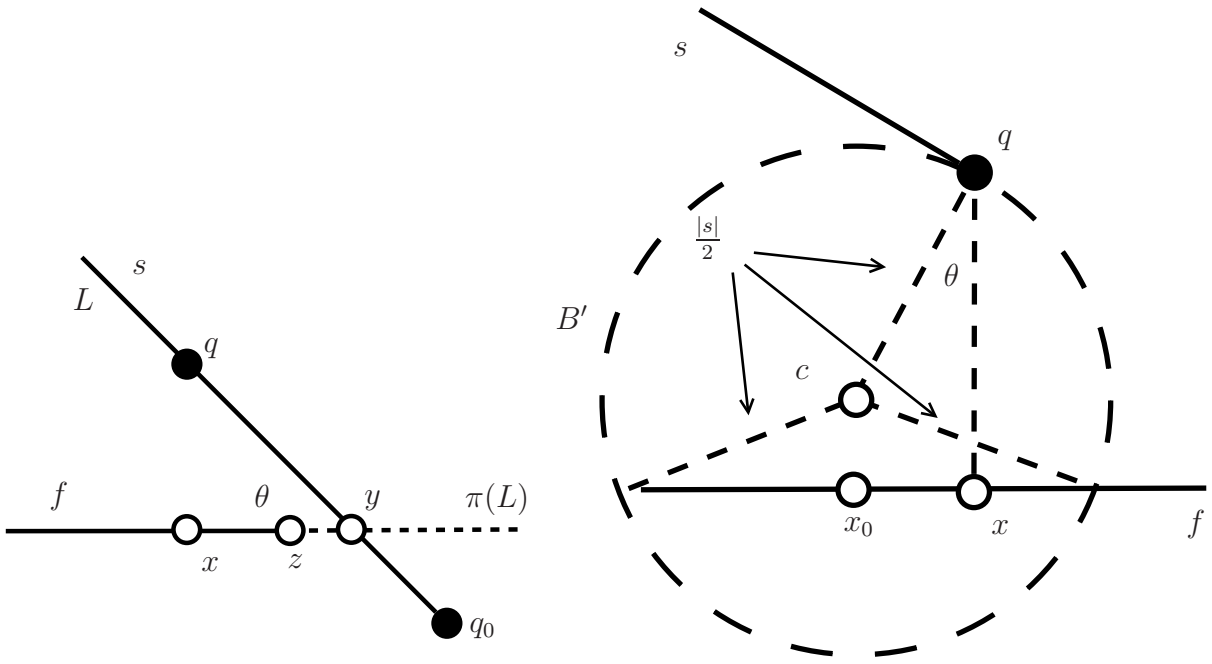
Figure 4.21: Diagram for Case 1.

In two cases, we show that the triangle t in f which contains x has been placed on the queue.

Case 1. Suppose that q is an input point. Now, let B be the ball of radius $\frac{|s|}{2}$ with q on the boundary and x on its diameter containing q as in Figure 4.20(a). The segment s is not on the queue, so q cannot have any Delaunay neighbors in B which witness the feature size of s. Since q is an input point and the nearest point on any segment containing q to face f, this means that B must be empty.

Proposition 4.15 implies that x belongs to some triangle in f with circumradius of at least $\sqrt{\frac{2}{3}}|q-x|$ as in Figure 4.21. Then applying Proposition 4.16 ensures that a vertex p of t must have a Delaunay neighbor which is not in the face at distance of at most $\sqrt{\frac{5}{3}}|q-x| < 2R_t$. Thus t has been queued at some step of the algorithm.

Case 2. Suppose that q is not an input point. Let q_0 be an input point on the segment containing s. First, claim that $|q_0 - q| \geq |s|$. If s is an end segment, this is trivial. If s is the



(a) The angle between the input segment containing s and f cannot be large.

(b) Estimating $|x - x_0|$.

Figure 4.22: Diagrams for Case 2.

subsegment of an end segment which existed at the end of Step 0, then this holds because s must be produced by a sequence of midpoint insertions. If s is a subsegment of a non-end segment which existed following Step 0, then q_0 is a 1-feature size witness for s and Step 2a ensures that $|s| \leq \frac{1}{2\sqrt{2}} \operatorname{fs}_1(s) < |q - q_0|$.

Next, consider the angle θ between L and $\pi(L)$ as in Figure 4.22(a). Using the fact that q is interior to an input segment, we will show that $\sin\theta \leq \frac{3}{5}$. Let $y = L \cap P$. If $\sin\theta > \frac{3}{5}$ then $|q - y| \leq |s|$ which means that y is contained in the input segment containing s and thus cannot be contained in f. This means that there is some point z on the segment $\overline{xy}$ contained in the boundary of f. Then the distance between z and q is less than $|s|$, meaning $\operatorname{lfs}_1(s) < |s|$. This violates the bound given in Step 2a which is maintained by the algorithm.

Let B' be a ball of radius $\frac{|s|}{2}$ which has a diameter with one endpoint at q and intersects $\pi(L)$ as in Figure 4.20(b). We assert that if B' is not empty, then the neighbor of q which lies in B' must be a local feature size witness for s. If s is a non-end segment, this is clear as B' only touches the line containing s at q. If s is an end segment, let q_0 be the input point which is an endpoint of s. The ball B' is below the cone formed by rotating L around the line containing q_0 and $\pi(q_0)$. Since q is the nearest point to P on the spindle of s, this implies that B' does not intersect any input segment containing q_0.

Since s is not queued and any point in B' would serve as an appropriate witness to cause s to be queued, B' must be empty.

Next we seek to apply Proposition 4.15 based on the fact that $B' \cap P$ is empty in f. Let c be the center of B' and let $x_0 = \pi(c)$. As seen in Figure 4.22(b), $|x - x_0| = \frac{|s| \sin\theta}{2}$ and the radius of $B' \cap P$ is

$$\sqrt{\frac{|s|^2}{4}\sin^2\theta - |q-x|^2 + |q-x||s|\cos\theta}.$$

Using Proposition 4.15, conclude that the triangle t containing x has circumradius of at least

$$R_t \geq \sqrt{|q-x||s|\cos\theta - |q-x|^2}.$$

Since $|q-x| < \frac{3}{5}|s|$ and $\cos\theta \geq \frac{4}{5}$,

$$\begin{aligned} R_t &\geq \sqrt{|q-x||s|\cos\theta - |q-x|^2} \\ &\geq |q-x|\sqrt{\frac{5}{3}\cdot\frac{4}{5} - 1} \\ &\geq \frac{|q-x|}{\sqrt{3}}. \end{aligned}$$

Now, by Proposition 4.16, there is a vertex of triangle t which has a Delaunay neighbor which is not in the face containing t and thus t has been queued.

In both cases, it was shown that t must have been put on the queue. If t is on the queue, then the inductive hypothesis holds. If the triangle queue is empty, deduce that t was processed and its circumcenter was rejected for being too close to a nearby edge based on the safety rule.

The circumcenter of t is only rejected if there was some segment $\hat{s}$ with endpoint $\hat{q}$, such that $|c_t - \hat{q}| < |\hat{s}|$ and $\hat{s}$ lies in the face containing t. Since face f is disjoint from the input feature containing s, this means that s must be a 1-feature size witness for $\hat{s}$ and vice versa. The following estimate on the distance between q and $\hat{q}$ then holds:

$$\begin{aligned} |q-\hat{q}|^2 &= |q-x|^2 + |x-\hat{q}|^2 \\ &\leq 3R_t^2 + (|c_t - \hat{q}| + |x - c_t|)^2 \\ &\leq 3R_t^2 + (|\hat{s}| + R_t)^2 \\ &\leq 7|\hat{s}|^2. \end{aligned}$$

Above, the fact $|\hat{s}| > |c_t - \hat{q}| \geq R_t$ was used to estimate R_t by $|\hat{s}|$. The second inequality holds since the circumdisk of t must be empty by the Delaunay property.

By the Lemma 4.10 (which is maintained throughout algorithm), $\text{dist}(s, \hat{s}) \geq 2\sqrt{2}|\hat{s}|$. This inequality contradicts the previous bound as $\sqrt{7} < 2\sqrt{2} = \sqrt{8}$.

Conclude that the inductive hypothesis holds and thus upon termination of the algorithm the upper bound on segment lengths holds. □

Also, it is important that this step maintains the estimate on the 1-feature size derived in Step 1b.

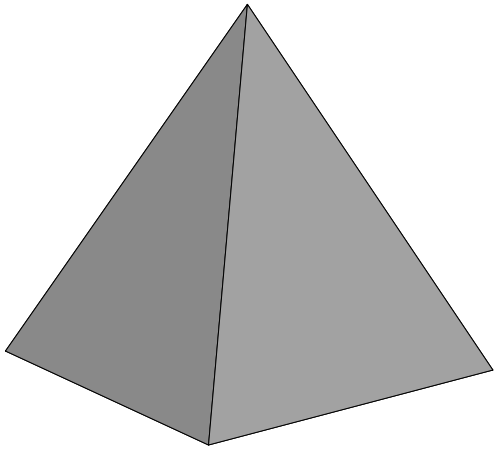


Figure 4.23: Initial PLC input for the pyramid example.

Lemma 4.18. *Upon termination of Step 2b, each segment s satisfies*

$$|s| \leq \sqrt{2}\,\mathrm{fs}_1(s).$$

This lemma is immediate for most of the segments in the mesh. Any end segment must satisfy this bound as $|s| \leq \frac{1}{2\sqrt{2}}\,\mathrm{fs}_1$ following Step 2a and splitting an end segment can only increase its 1-feature size. Similarly, for any segment which is a subsegment of a non-end segment which existed at the end of Step 1b, the same argument applies. This leaves only newly formed non-end segments which are subsegments of end segments of the mesh produced by Step 1b.

This proof is nearly identical to the proof of Lemma 4.9. In the base case, any segment which fails the bound must be queued since adjacent segments have the same length and thus no points on the same input segment can prevent the segment in question from being queued. In each step of the proof, nearby Delaunay neighbors of the endpoints of a segment are considered. In the Step 1b proof, either these neighbors are appropriate feature size witnesses to cause the segment to be queued, or they lie on an input segment. In Step 2b, this is still the case, due to the safety rule. This ensures that the endpoints of segment s will not have any Delaunay neighbors in any plane containing s within a distance of $|s|$.

4.2 Examples

The next three examples demonstrate some simple applications of the algorithm.

Example 4.2.1. The first example is a square pyramid shown in Figure 4.23. The mesh of the square base produced following each step of the algorithm can be seen in Figure 4.24. Similar output for one of the triangular sides is given in Figure 4.25.

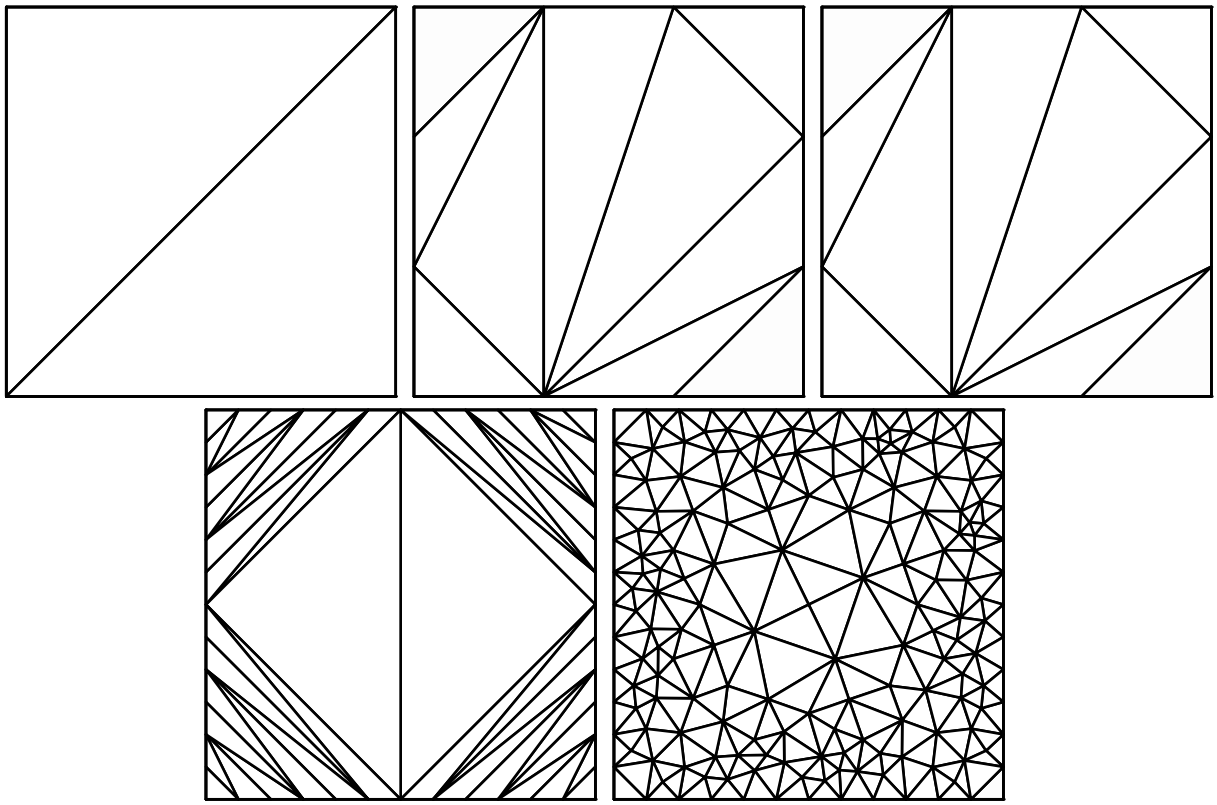


Figure 4.24: Base of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

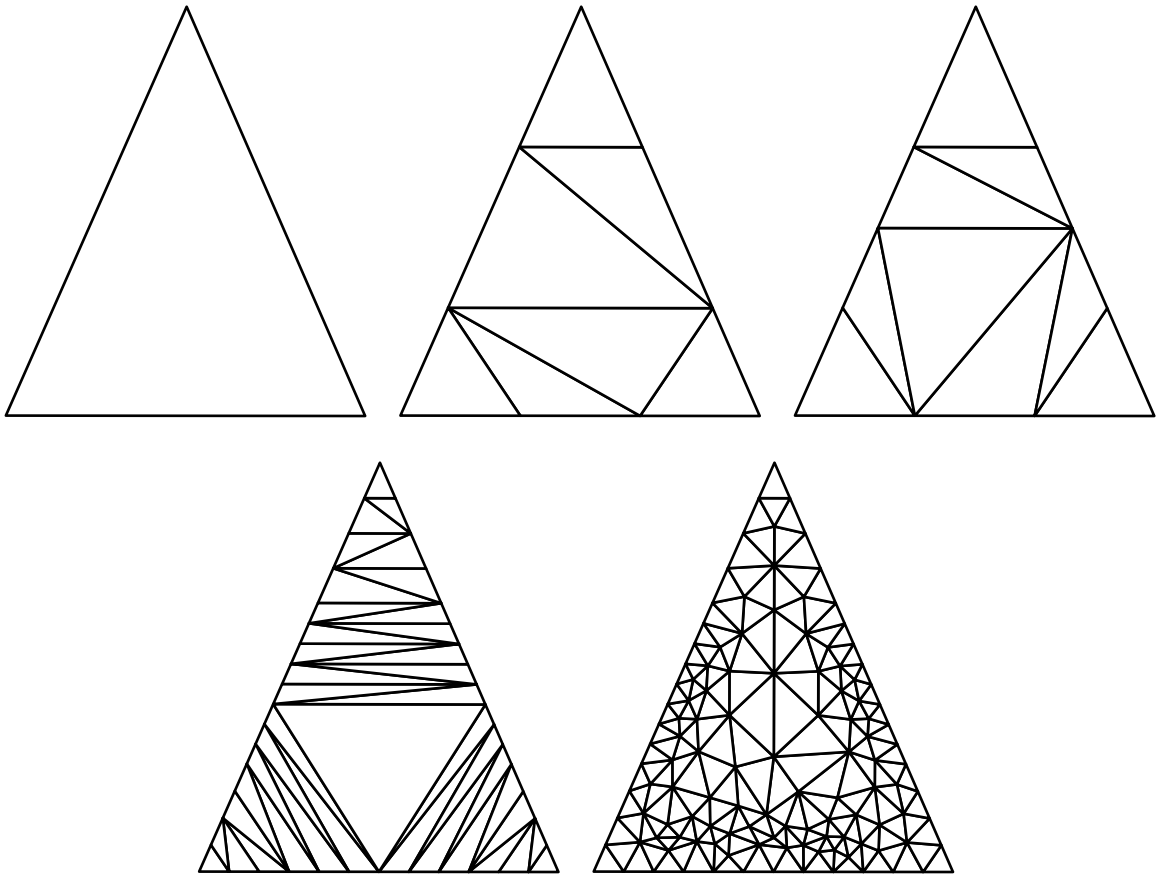


Figure 4.25: Side of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

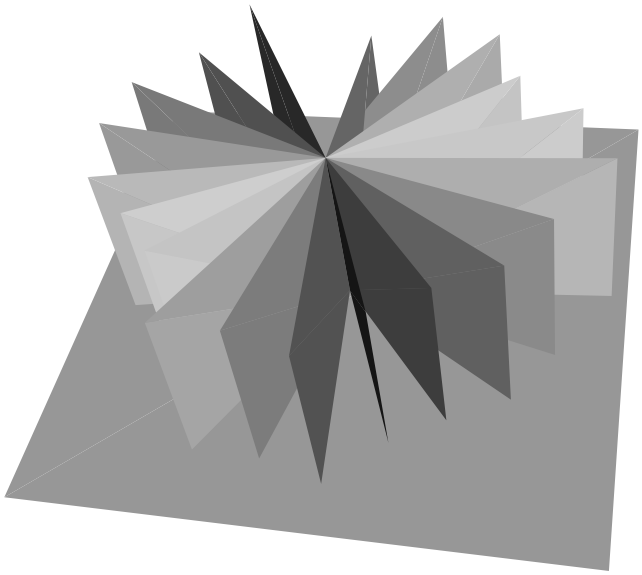


Figure 4.26: Wheel example: input PLC.

Example 4.2.2. This example consists of a wheel of 20 faces which lies slightly above a disjoint square as depicted in Figure 4.26. The mesh of the square base produced following each step of the algorithm can be seen in Figure 4.27. Similar output for one of the rectangular "spokes" of the wheel is given in Figure 4.28. Note that the algorithm still terminates even in the presence of acute angles in the input. The number of vertices in the mesh after each step is listed in Table 1.

Table 4.1: Number of points in the mesh following each step of the algorithm in Example 4.2.2.

Step	0	1a	1b	2a	2b
Vertices	72	202	518	2,051	11,351

Example 4.2.3. In the final example, we consider a PLC containing two non-convex faces shown in Figure 4.29. The refinement of one of these faces is shown in Figure 4.30.

In these examples, nearby edges typically cause more refinement than nearby faces. This is a result of Step 2a which causes segments to be split in fourths *after* they have been refined to realize fs_1. This can also be seen in Theorem 4.1 as each segment is guaranteed to have length of at least $\frac{1}{4}\,\text{lfs}(s)$ or $\frac{1}{16}\,\text{fs}_1(s)$. A small fs_1 does in practice lead to more refinement than simply a small lfs as was suggested by the constants in the proof.

The proof of Lemma 4.17 (and thus Theorem 4.2) uses Step 2a to ensure a bound on each segment's length by lfs_1. For some segments, this is an over-refinement since they were refined based on mfs_1 and not lfs_1. We continue to study an adaptive variant of Step 2a which attempts to only split segments in fourths when absolutely necessary.

In practice, the algorithm has been seen to terminate even after changing Step 2a to only split

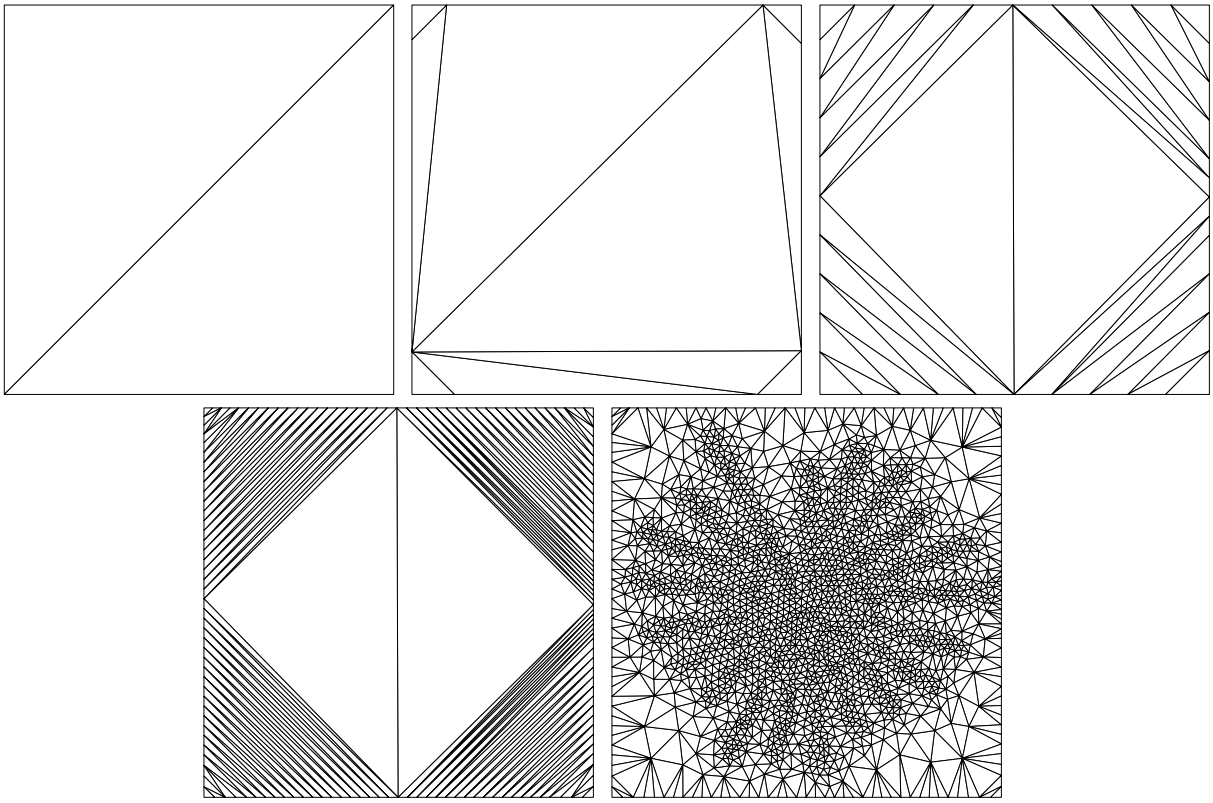


Figure 4.27: Base plane of the wheel example following steps 0, 1a, 1b, 2a, and 2b.

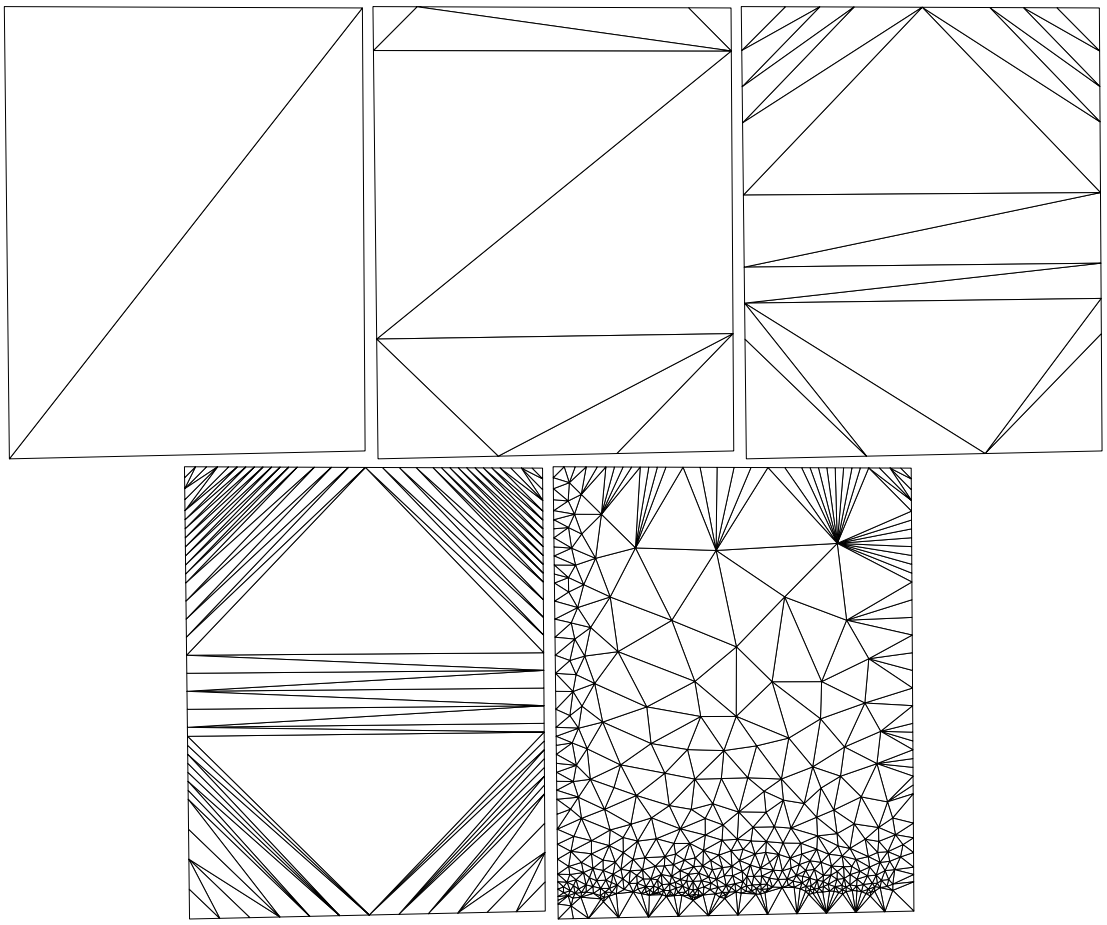


Figure 4.28: One “spoke” in the wheel example following steps 0, 1a, 1b, 2a, and 2b. The center of the wheel is at the bottom while the disjoint square is to the left of this face.

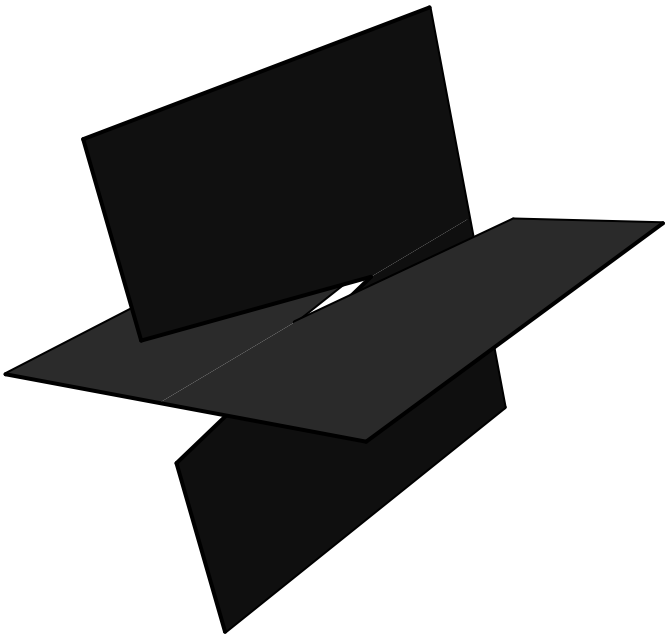


Figure 4.29: Example containing non-convex input faces.

segments in half (instead of fourths). This significantly reduces the output size (often by 50% or more in cases containing small input angles between faces). In further studies, we will seek to justify this modification of the algorithm in the proof or give a counterexample showing that the algorithm can fail without performing Step 2a as specified.

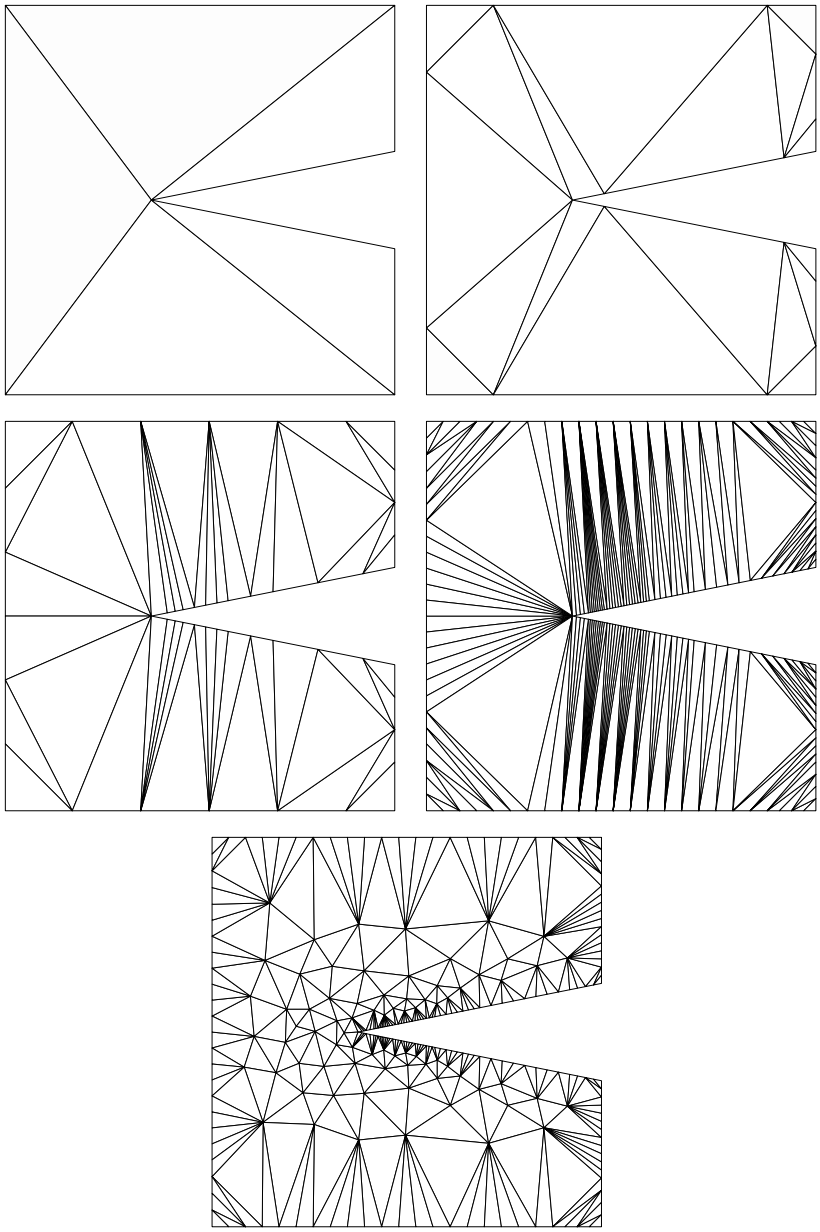


Figure 4.30: One of the faces in Example 4.2.3 following steps 0, 1a, 1b, 2a, and 2b.

Chapter 5

Quality, Conforming Triangulation

Acute input angles have long posed significant challenges to Delaunay refinement. In his original paper on Delaunay refinement, Ruppert immediately recognized this challenge and (inspired by previous work [3, 33]) suggests two related concepts for dealing with these issues: concentric shell splitting and corner lopping. Nearly all methods for performing Delaunay refinement on input with acute angles are based on these basic ideas. In this chapter, we formalize two closely related procedures for protecting acute input angles during Delaunay refinement and the necessary techniques needed to extend the standard analysis in Section 2.2.

Shewchuk gave the first Delaunay refinement algorithm guaranteed to terminate for general acute input [41]. Miller, Pav, and Walkington gave an alternative algorithm [30, 35] with a number of improved properties such as the elimination of any large angles in the mesh. However, no natural extension of these ideas to three dimensions has been found. Methods for protecting acute angles in three dimensions come in two flavors: "collars" and "intestines." The former generalizes a class of algorithms related to the Shewchuk's terminator algorithm and a 3D algorithm of Pav and Walkington [36] while the latter describes a class including the "second" (2D) Pav-Walkington algorithm (generalized in [37]) and the 3D refinement algorithm of Cheng and Poon [11]. For completeness and clarity the simplest collar and intestine protection procedures are described in 2D. This allows for a more natural extension to 3D in the next chapter.

Both approaches require estimates on local feature size at the input points of the mesh which can be determined via methods described in Chapter 3. Then protecting acute input angles involves two steps. First, an input PLC is augmented to ensure conformality of the resulting mesh near the acute angle. Second, Delaunay refinement is performed with an appropriate policy for accepting poor quality triangles near the acute angle. As the natural successor to Algorithm 3.1, these two steps will be labeled Step 1b and Step 2, respectively as in Algorithm 5.1.

Algorithm 5.1 Quality Refinement of Acute Input 2D

(Step 1b) Protect acute input angles.

(Step 2) Perform a protected version of Ruppert's algorithm.

5.1 Properties Of Local Feature Size

Before describing the algorithms and analysis, it is important to consider a few general facts about the changes in local feature size caused by certain refinements and augmentations of PLCs. In order to protect acute input features, we will augment the input PLC to produce a PLC or PSC for which conformality near acute input angles can easily be maintained. The next few results describe how the local feature size in the resulting PLC depends on the initial feature size and the smallest angle in the input.

Proposition 5.1. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ be a PLC or PSC and let $\bar{\mathcal{C}} = (\bar{\mathcal{P}}, \bar{\mathcal{S}})$ be a refinement of $\mathcal{C}$. Let $\bar{\mathcal{C}}_s$ be some (possibly trivial) subcomplex of $\bar{\mathcal{C}}$. If*

$$\operatorname{lfs}(y, \mathcal{C}) \leq K \operatorname{lfs}(y, \bar{\mathcal{C}}_s)$$

for all points y which belong to some feature of $\bar{\mathcal{C}}_s$ of dimension at most $\dim(\bar{\mathcal{C}}_s) - 1$, then

$$\operatorname{lfs}(x, \mathcal{C}) \leq (2K + 1) \operatorname{lfs}(x, \bar{\mathcal{C}}_s)$$

holds for all x.

Proof. Let x be any point. Let y be the nearest point on a feature of $\bar{\mathcal{C}}_s$ to x. Then

$$\begin{aligned} \operatorname{lfs}(x, \mathcal{C}) &\leq \operatorname{lfs}(y, \mathcal{C}) + |x - y| \\ &\leq K \operatorname{lfs}(y, \bar{\mathcal{C}}_s) + |x - y| \\ &\leq K \operatorname{lfs}(x, \bar{\mathcal{C}}_s) + (K + 1)|x - y| \\ &\leq (2K + 1) \operatorname{lfs}(x, \bar{\mathcal{C}}_s) \end{aligned}$$

Above, we have used the fact that $|x - y| \leq \operatorname{lfs}(x, \bar{\mathcal{C}}_s)$ since y is the nearest point to x on any feature of $\bar{\mathcal{C}}_s$. □

Using a nearly identical proof, we show a similar result involving the augmentation of a PLC or PSC with additional features.

Proposition 5.2. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ and $\hat{\mathcal{C}} = (\mathcal{P} \bigcup \hat{\mathcal{P}}, \mathcal{S} \cup \hat{\mathcal{S}})$ be PLCs or PSCs. Suppose that for all $y \in \hat{\mathcal{P}}$ or $y \in s \in \hat{\mathcal{S}}$, if*

$$\operatorname{lfs}(y, \mathcal{C}) \leq K \operatorname{lfs}(y, \hat{\mathcal{C}})$$

then for all x

$$\operatorname{lfs}(x, \mathcal{C}) \leq (4K + 3) \operatorname{lfs}(x, \hat{\mathcal{C}}).$$

Proof. Consider any point y contained in a feature of $\mathcal{C}$. Suppose $\operatorname{lfs}(y, \hat{\mathcal{C}}) < \operatorname{lfs}(y, \mathcal{C})$. This implies the existence of a point z either in $\hat{\mathcal{P}}$ or on a segment of $\mathcal{S}$ which is disjoint from the

feature containing y such that $|y - z| \leq \operatorname{lfs}(y, \hat{\mathcal{C}})$. Hence,

$$\begin{aligned}\operatorname{lfs}(y, \mathcal{C}) &\leq \operatorname{lfs}(z, \mathcal{C}) + |y - z| \\ &\leq K \operatorname{lfs}(z, \hat{\mathcal{C}}) + |y - z| \\ &\leq K \operatorname{lfs}(y, \hat{\mathcal{C}}) + (K+1)|y - z| \\ &\leq (2K+1) \operatorname{lfs}(y, \hat{\mathcal{C}}).\end{aligned}$$

Finally, we can apply Proposition 5.1 to extend this bound from points on features of $\hat{\mathcal{C}}$ to all points and get the desired estimate. □

In the next lemma, Proposition 5.1 will be applied to both the entire refined PLC $\bar{\mathcal{C}}$ as well as certain subcomplexes $\bar{\mathcal{C}}_s$ which are the PLCs containing only features which are contained in some segment s. Recall that α is defined to be the smallest angle between adjacent input segments.

Lemma 5.3. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ be a PLC with refinement $\bar{\mathcal{C}} = (\mathcal{P} \cup \bar{\mathcal{P}}, \bar{\mathcal{S}})$ such that*

$$\bar{\mathcal{P}} \subset \bigcup_{s \in \mathcal{S}} s^{\circ} \tag{5.1}$$

where s° is the relative interior of segment s. For a segment $s \in \mathcal{S}$, let $\bar{\mathcal{C}}_s$ be the PLC containing all features of $\bar{\mathcal{C}}$ which are contained in s. If for any segment $s \in \mathcal{S}$

$$\operatorname{lfs}(q, \mathcal{C}) \leq K \operatorname{lfs}(q, \bar{\mathcal{C}}_s)$$

holds for any vertex $q \in s \cap \bar{\mathcal{P}}$, then

$$\operatorname{lfs}(x, \mathcal{C}) \leq \left(\frac{4K+4}{\sin(\alpha)} + 4K + 3 \right) \operatorname{lfs}(x, \bar{\mathcal{C}})$$

holds for all x.

Proof. First, let $q_0 \in \mathcal{P}$ and let s be a segment containing q_0. Let q be the nearest vertex to q_0 on s. If $q \in \mathcal{P}$, then

$$\operatorname{lfs}(q_0, \mathcal{C}) \leq \operatorname{lfs}(q_0, \mathcal{C}_s) = \operatorname{lfs}(q_0, \bar{\mathcal{C}}_s).$$

Otherwise, $q \in \bar{\mathcal{P}}$ and it follows that

$$\begin{aligned}\operatorname{lfs}(q_0, \mathcal{C}) &\leq \operatorname{lfs}(q, \mathcal{C}) + |q - q_0| \\ &\leq K \operatorname{lfs}(q, \bar{\mathcal{C}}_s) + |q - q_0| \\ &\leq K \operatorname{lfs}(q_0, \bar{\mathcal{C}}_s) + (K+1)|q - q_0| \\ &\leq (2K+1) \operatorname{lfs}(q_0, \bar{\mathcal{C}}_s).\end{aligned} \tag{5.2}$$

Next, let u be a point (not necessarily a vertex in the mesh) contained in some segment of $\mathcal{S}$. The local feature size can be realized several ways. Let w be the point on a disjoint feature of $\bar{\mathcal{C}}$ which realizes the local feature size at u with respect to $\bar{\mathcal{C}}$.

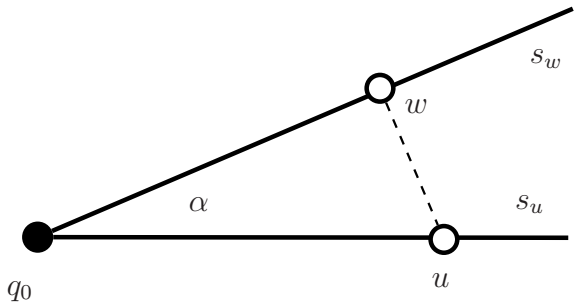


Figure 5.1: In Case 3 of Lemma 5.1, $\text{dist}(p, s_2) \geq \frac{|p-q_0|}{\sin(\alpha)}$.

Case 1. u and w lie on disjoint features of $\mathcal{C}$.

Then $\text{lfs}(u, \mathcal{C}) = \text{lfs}(u, \bar{\mathcal{C}})$.

Case 2. u and w are contained in the same segment of $s \in \mathcal{S}$.

In this case, apply Proposition 5.1 using (5.2) to get the desired inequality:

$$\text{lfs}(u, \mathcal{C}) \leq (4K+3)\,\text{lfs}(u, \bar{\mathcal{C}}_s) = (4K+3)\,\text{lfs}(u, \bar{\mathcal{C}}).$$

Case 3. u and w belong to adjacent segments in $\mathcal{C}$, denoted s_u and s_w.

Let q_0 denote the input point which lies at the intersection of s_u and s_w. Then,

$$\begin{aligned}
\text{lfs}(u, \mathcal{C}) &\leq \text{lfs}(q_0, \mathcal{C}) + |u - q_0| \\
&\leq (2K+1)\,\text{lfs}(q_0, \bar{\mathcal{C}}) + |u - q_0| \\
&\leq (2K+1)\,\text{lfs}(u, \bar{\mathcal{C}}) + (2K+2)|u - q_0| \\
&\leq \left(2K + 1 + \frac{2K+2}{\sin\alpha}\right) \text{lfs}(u, \bar{\mathcal{C}}).
\end{aligned}$$

As seen in Figure 5.1, we have used the fact that

$$\text{lfs}(u, \bar{\mathcal{C}}) \geq \text{dist}(u, s_w) \geq \frac{|u - q_0|}{\sin\alpha}.$$

Finally, applying Proposition 5.1 on the entire complex $\bar{\mathcal{C}}$ with constant K yields the desired result. □

Lemma 5.3 will surely not produce a sharp bound on the local feature size of the refined complex. However, this is not our purpose: the goal of these lemmas is to prevent the case explosion which occurs when attempting this direct analysis. In this direction, we note that to apply this lemma, estimates on the local feature size only need to be verified at a few vertices (the newly inserted ones) to assert estimates on the local feature size for any point. Lemma 5.3 does produce the correct scaling with respect to the smallest input angle α.

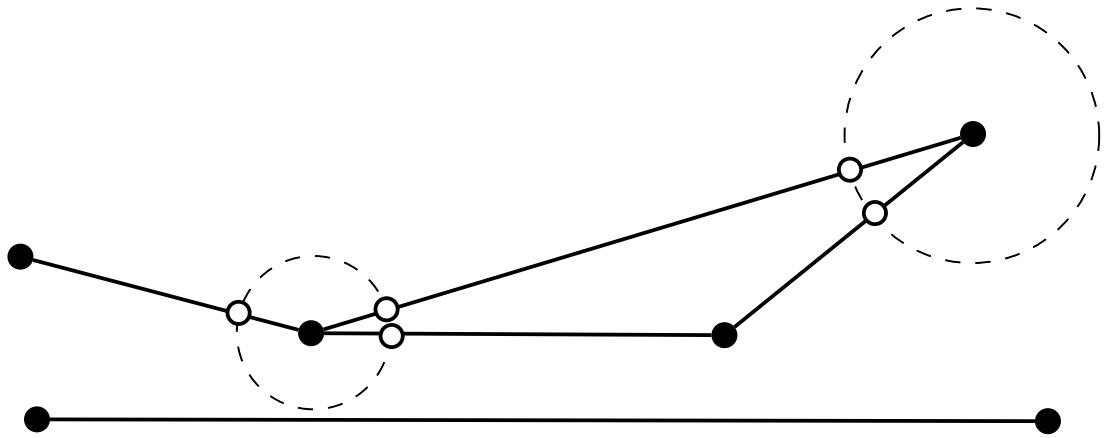


Figure 5.2: Disks around protected input points do not intersect features of the mesh disjoint to the input point.

5.2 Collar Protection Region

A collar protection region involves forming "collar" segments of equal length around each input point which ensures the input segments conform near this input point. Our resulting Delaunay refinement algorithm will then prevent the insertion of any vertices which encroach this collar region.

5.2.1 Step 1b

Protect acute input angles.

For each input point q_0 which is the vertex of an acute input angle, the collar is formed by splitting all segments containing q_0 at an equal distance d_q such that

$$b \operatorname{lfs}(q_0) \leq d_{q_0} \leq \min(c_0 \operatorname{lfs}_0(q_0), c_1 \operatorname{lfs}(q_0))$$

for some constants $b > 0$, $c_0 \in (0, .5)$ and $c_1 \in (0, 1)$. Algorithm 3.1 can be used to determine an acceptable distance. Figure 5.2 depicts an example of the points inserted during this step.

Each end segment containing the vertex of an acute input angle will be called a collar simplex and vertices inserted during this step are called collar vertices. See Figure 5.3.

First, we observe that the collar simplices are sufficiently far away from disjoint input features of $\mathcal{C}$.

Lemma 5.4. *For any input point $q_0 \in \mathcal{P}$,*

$$B(q_0, d_{q_0}) \bigcap B(q_0, d_{q_0'}) = \emptyset \text{ for all } \mathcal{P} \ni q_0' \neq q_0$$

and

$$B(q_0, d_{q_0}) \bigcap s = \emptyset \text{ for all segments } s \text{ disjoint from } q_0.$$

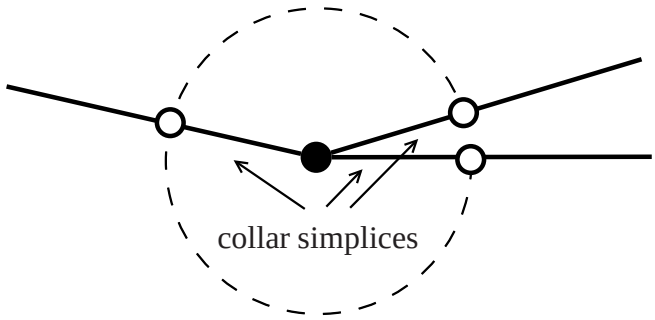


Figure 5.3: Definition of collar simplices

Proof. This follows directly from the restrictions on c_0 and c_1. □

Let $\bar{\mathcal{C}}$ denote the refined PLC obtained after inserting all of the collar vertices. The next lemma quantifies the relationship between the local feature size of $\bar{\mathcal{C}}$ and $\mathcal{C}$.

Lemma 5.5. *There exists $K > 0$ depending on only b and c_0 such that*

$$\text{lfs}(x, \bar{\mathcal{C}}) \leq \text{lfs}(x, \mathcal{C}) \leq \frac{K}{\sin(\alpha)} \text{lfs}(x, \bar{\mathcal{C}})$$

for all x.

Proof. Let s be an input segment, and let q be a collar vertex in s. Observe the following inequality:

$$\text{lfs}(q, \mathcal{C}) \leq \max\left\{\frac{1-b}{b}, \frac{1-b}{1-2c_0}\right\} \text{lfs}(q, \bar{\mathcal{C}}_s).$$

The desired inequality then follows from Lemma 5.3. □

5.2.2 Step 2

Perform a protected version of Ruppert's algorithm.

This step is the Delaunay refinement algorithm described in Algorithm 5.2. Each new end segment is "protected" during refinement: no vertices of the mesh will be allowed to enter the diametral ball of these segments. To ensure this, circumcenters which encroach these end segments will be rejected by the safety criteria of the algorithm. Lemma 5.4 ensures that no inserted midpoints encroach upon a collar simplex and thus the diametral disk of each collar simplex will be empty throughout the algorithm.

The termination of the algorithm and properties of the resulting mesh are described in Theorems 5.6 and 5.7. The first theorem ensures that the algorithm terminates and the resulting mesh is graded to the feature size while the second theorem asserts that the mesh conforms to the input PLC and specifies which triangles may have poor quality.

Algorithm 5.2 2D Delaunay Refinement With Collar

Action	Insert the circumcenter of a simplex.
Priority	Encroached segments are given higher priority than poor quality triangles.
Unacceptability	Any segment with a non-empty diametral disk is unacceptable. Any triangle with raduis-edge ratio less than τ is also unacceptable.
Safety	Collar simplices are not safe to split.

Theorem 5.6. *There exists $C > 0$ depending only upon τ, α, b, and c_0 such that for each vertex q inserted in the mesh,*

$$\text{lfs}(q) \leq Cr_q.$$

Proof. The analysis of Ruppert's algorithm (Theorem 2.2) applies with respect to the protected complex $\bar{\mathcal{C}}$ since end segments are never split (which follows from Proposition 5.4) and thus no segment is encroached by a vertex on an adjacent segment of $\bar{\mathcal{C}}$. This asserts the existence of C depending only upon τ such that

$$\text{lfs}(q, \bar{\mathcal{C}}) \leq Cr_q.$$

The proof is then completed by applying the estimate in Lemma 5.5 to convert the estimate on the local feature size with respect to $\bar{\mathcal{C}}$ into an estimate on the local feature size with respect to $\mathcal{C}$. □

Remark. While the constant in the previous theorem does not depend on c_1, the restiction that $c_1 < 1$ is important to ensure the proof is valid: otherwise, it would be possible for vertices to be inserted on segments which encroach upon collar simplices and the proof would not hold.

Theorem 5.7. *The resulting Delaunay triangulation conforms to the input. The circumcenter of any remaining poor quality triangles lies in the diametral disk of a collar simplex.*

Proof. No vertex is inserted which encroaches a collar simplex so all collar simplices conform in the resulting mesh. All unacceptable (and thus non-collar) segments are queued for splitting and none are rejected by the safety rule, thus in the final mesh all segments conform to the input. □

5.3 Intestine Protection Region

The intestine protecting region is more complex than the collar approach, but yields the added result that no triangles in the resulting mesh have angles larger than $\pi - 2\kappa$, where $\kappa := \sin^{-1}\left(\frac{1}{2\tau}\right)$ is the minimum angle corresponding to the radius-edge threshold τ. Again, we will insert additional points in Step 1b and perform the appropriate Delaunay refinement in Step 2.

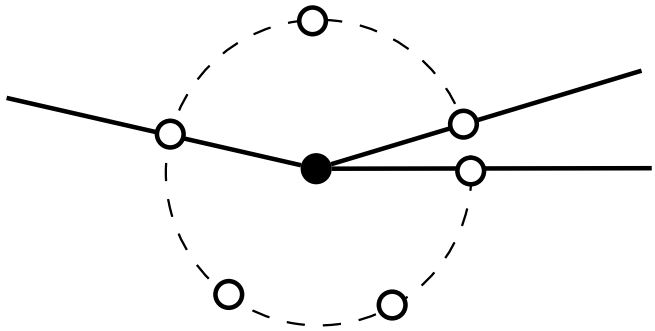


Figure 5.4: Vertices inserted for intestine protecting region.

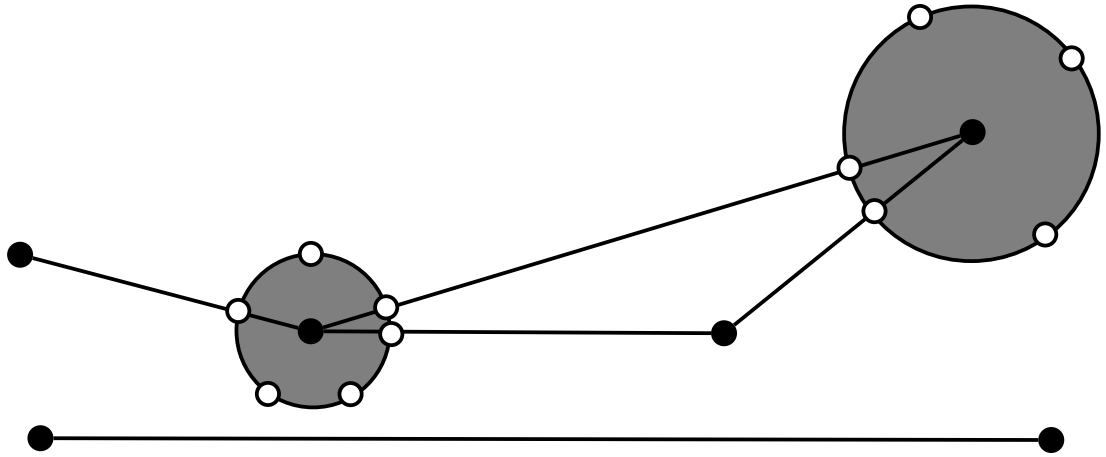


Figure 5.5: Consider the new PSC formed by adding protecting circles to the input PLC.

5.3.1 Step 1b

Protect acute input angles.

As with the collar, for each input point q_0 which is the vertex of an acute input angle, consider distance d_{q_0} such that

$$b \operatorname{lfs}(q_0) \leq d_{q_0} \leq \min(c_0 \operatorname{lfs}_0(q_0), c_1 \operatorname{lfs}(q_0))$$

with $b > 0$, $c_0 \in (0, .5)$ and $c_1 \in (0, 1)$.

For each input vertex q_0 at an acute input angle, we will split all input segments at a distance d_{q_0} from point q_0. Additionally, consider the circle centered at q_0 of radius d_{q_0} and add points to ensure that no arc of this circle is larger than $\frac{\pi}{2}$. This ensures that the diametral ball of each arc of the circle does not contain q_0 and requires at most 3 additional vertices per input vertex. An example of this construction is shown in Figure 5.4.

We will now consider a piecewise smooth complex (PSC) $\hat{\mathcal{C}}$ defined by the input PLC, vertices inserted during Step 1b, and the boundary arcs of each disk $B(q, d_q)$ as depicted in Figure 5.5.

Lemma 5.8. *Let $\hat{\mathcal{C}}$ be the PSC derived from PLC $\mathcal{C}$ as described above. Then there exists $K > 0$ depending only on b, c_0, and c_1 such that for all x*

$$\text{lfs}(x, \hat{\mathcal{C}}) \leq \text{lfs}(x, \mathcal{C}) \leq \frac{K}{\sin \alpha} \text{lfs}(x, \hat{\mathcal{C}}).$$

Proof. The first inequality is immediate from the definition of local feature size: adding additional features and refining existing ones only decreases the local feature size. The second inequality will be shown by applying Proposition 5.2 to $\bar{\mathcal{C}}$, the complex with just the initial collar vertices inserted. Let $y \in \partial B(q, d_q)$. Then,

$$\text{lfs}(y, \bar{\mathcal{C}}) \leq \frac{2}{1 - c_1} \text{lfs}(y, \bar{\mathcal{C}}).$$

Combining the conclusion of Proposition 5.2 with Lemma 5.5 yields the result. □

5.3.2 Step 2

Perform a protected version of Ruppert's algorithm.

Now Ruppert's algorithm is performed outside of $\bigcup_{q_0} B(q_0, d_{q_0})$ and each of the boundary arcs of any disk $B(q_0, d_{q_0})$ is protected by the diametral disk of its endpoints. This is described completely in Algorithm 5.3.

Algorithm 5.3 2D Delaunay Refinement With Intestine

Action	Insert the circumcenter of a triangle. Insert the midpoint of a segment or arc.
Priority	Encroached segments and arcs are given higher priority than poor quality triangles.
Unacceptability	Any segment or arc with a non-empty diametral disk is unacceptable. Any triangle with radius-edge ratio larger than τ is also unacceptable.
Safety	All simplices and arcs are safe to split.

Ruppert's algorithm can be generalized to piecewise smooth input complexes [4, 7, 37]. The analysis of Pav and Walkington applies to Algorithm 5.3, however that theory involves a trade-off between the maximum total variation in orientation of the curves in the input and the smallest allowable radius-edge threshold of the refinement. This application leads to two choices:

- Use the input as specified and select a sufficiently large τ to ensure termination.
- Pick any $\tau > \sqrt{2}$. First split smooth input features to have a sufficiently small total variation in orientation based on the τ value selected and then perform Algorithm 5.3.

However, we will show that this trade-off is unnecessary. Algorithm 5.3 will terminate and produce a well-graded, conforming Delaunay triangulation for any input complex previously

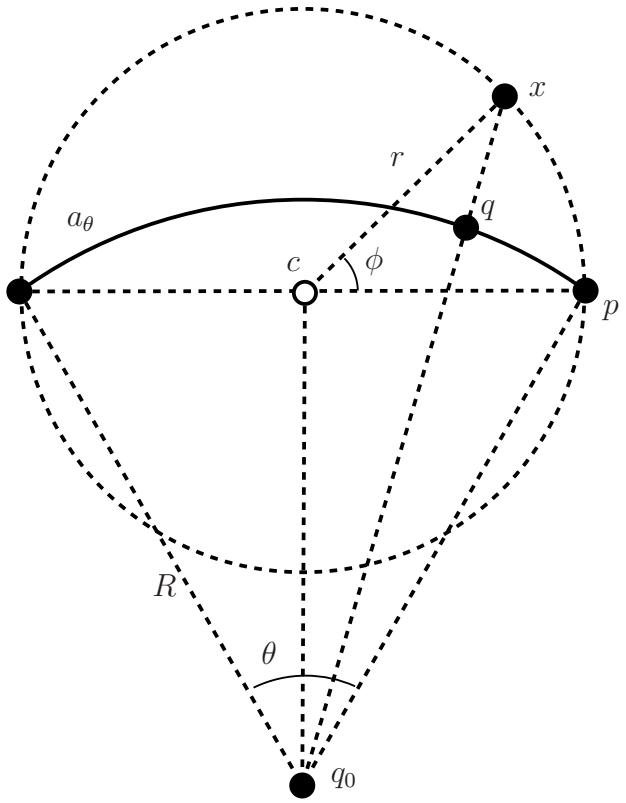


Figure 5.6: Configuration in Proposition 5.9.

described (such that all input arcs have total variation in orientation of at most $\frac{\pi}{2}$) and any radius-edge threshold $\tau > \sqrt{2}$. This will require a more careful analysis, which is centered around the following technical lemma.

Proposition 5.9. *Let $\theta \in (0, \frac{\pi}{2}]$ and let a_θ be the arc of a circle $\partial B(q_0, R)$ which subtends an angle θ. Let $x \in \partial B(a_\theta) \setminus B(q_0, R)$, let p be the nearest endpoint of a_θ to x and let q be the projection of x onto $\partial B(a_\theta)$. For any $\tau > \sqrt{2}$ there exists $\beta_\tau^* > 0$, independent of θ, such that if*

$$\frac{|x-q|}{R} \leq \beta_\tau^*,$$

then

$$\frac{|x-p|}{|x-q|} \leq \tau.$$

Remark. The most important feature of this lemma is that β_τ^* is independent of θ. This is essential in the proof of termination of the Delaunay refinement algorithm for any $\tau > \sqrt{2}$ without a coupled restriction based on the total variation in orientation of the input curves.

Proof. Let c be the center of $B(a_\theta)$ and let ϕ denote the angle between segments $\overline{pc}$ and $\overline{xc}$ as shown in Figure 5.6. Let r denote the radius of $B(a_\theta)$.

Applying the fact that $r = R\sin\left(\frac{\theta}{2}\right)$ gives

$$|x-p| = 2r\sin\left(\frac{\phi}{2}\right) = 2R\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right). \tag{5.3}$$

Using $|c-q_0| = R\cos\left(\frac{\theta}{2}\right)$ and the law of cosines gives the next sequence of equalities:

$$\begin{aligned}(|x-q|+R)^2 &= r^2 + \left(R\cos\left(\frac{\theta}{2}\right)\right)^2 - 2rR\cos\left(\frac{\theta}{2}\right)\cos\left(\phi+\frac{\pi}{2}\right)\\ &= \left(R\sin\left(\frac{\theta}{2}\right)\right)^2 + \left(R\cos\left(\frac{\theta}{2}\right)\right)^2 - 2R^2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\phi+\frac{\pi}{2}\right)\\ &= R^2 + R^2 2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\phi\\ &= R^2\left(1+\sin\theta\sin\phi\right).\end{aligned}$$

Rearranging leads to an expression for $|x-q|$:

$$|x-q| = R\left(\sqrt{1+\sin(\theta)\sin(\phi)}-1\right). \tag{5.4}$$

Combining (5.3) and (5.4) gives

$$\frac{|x-p|}{|x-q|} = \frac{2\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right)}{\sqrt{1+\sin(\theta)\sin(\phi)}-1} = \frac{\sqrt{1+\sin(\theta)\sin(\phi)}+1}{2\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right)}. \tag{5.5}$$

Let $\beta := \frac{|x-q|}{R}$ and $\gamma := \sqrt{2\beta+\beta^2}$. Rearranging terms in (5.4) yields

$$\gamma^2 = 2\beta+\beta^2 = \sin(\theta)\sin(\phi). \tag{5.6}$$

This implies that $\gamma \geq \sin\phi$ or $\gamma \geq \sin\theta$. These two possibilities are handled in two cases. However, since (5.5) is symmetric in the variables ϕ and θ, the argument is identical in each case. Thus, we consider only the case $\gamma \geq \sin\phi$. Then,

$$\sqrt{1+\sin(\theta)\sin(\phi)} = \sqrt{1+\gamma^2} \leq 1+\frac{\gamma^2}{2},$$

and

$$\begin{aligned}\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) \geq \cos\left(\frac{\pi}{4}\right)\cos\left(\frac{\sin^{-1}\gamma}{2}\right) &\geq \left(\frac{1}{\sqrt{2}}\right)\left(1-\frac{\left(\sin^{-1}\gamma\right)^2}{8}\right)\\ &\geq \left(\frac{1}{\sqrt{2}}\right)\left(1-\frac{\pi^2\gamma^2}{32}\right).\end{aligned}$$

The final inequality results from $\sin^{-1}\gamma \leq \frac{\pi}{2}\gamma$ which requires that $\gamma \leq 1$. To ensure this, we will seek $\beta_\tau^* \leq \frac{1}{3}$. Substituting these estimates into (5.5) gives

$$\frac{|x-p|}{|x-q|} \leq \sqrt{2}\left(\frac{1+\gamma^2/4}{1-\gamma^2\pi^2/32}\right).$$

Let

$$\gamma^* = 4\sqrt{\frac{2(\tau - \sqrt{2})}{8\sqrt{2} + \pi^2\tau}},$$

and select

$$\beta_\tau^* = \frac{1}{3}\min\left(1, (\gamma^*)^2\right).$$

Substitution yields that if $\frac{|x-q|}{R} \leq \beta_\tau^*$, then

$$\frac{|x-p|}{|x-q|} \leq \tau. \qquad \square$$

Now, we are prepared to prove that Algorithm 5.3 terminates and that the output mesh is well graded.

Theorem 5.10. *There exists $C > 0$ depending on τ, α, c_0, c_1, and b such that for each vertex q inserted in the mesh,*

$$\text{lfs}(q) \leq Cr_q.$$

Proof. Recall that $\hat{\mathcal{C}}$ denotes the PSC which includes the original input with the protecting circles around each input vertex which are split into arcs subtending at most $\frac{\pi}{2}$. Lemma 5.8 ensures that it is sufficient to prove the theorem considering local feature size with respect to $\hat{\mathcal{C}}$ rather than $\mathcal{C}$. So, as in the proof of Theorem 2.2, we will show inductively that

$$\text{lfs}(q, \hat{\mathcal{C}}) \leq \begin{cases} r_q & \text{if } q \text{ is an input point,} \\ C_1 r_q & \text{if } q \text{ is a segment or arc midpoint,} \\ C_2 r_q & \text{if } q \text{ is a circumcenter.} \end{cases}$$

Cases 1 through 3 in Theorem 2.2 also apply in the same fashion as before. However, there are additional cases which must be considered. These cases involve showing the estimate $\text{lfs}(q) \leq C_1 r_q$ for a vertex q which is inserted as a midpoint of some circular arc a in the refined complex. These cases are distinguished by the type of point x which is the nearest neighbor to q when q is inserted.

Case 4. Vertex q is the midpoint of some arc and the nearest neighbor to q lies on an input feature (in $\hat{\mathcal{C}}$) which is disjoint from a.

Then $\text{lfs}(q, \hat{\mathcal{C}}) \leq |q - x| = r_q$ since this vertex must be on a disjoint feature (with respect to $\hat{\mathcal{C}}$ because $\hat{\mathcal{C}}$ is non-acute). This yields the (previously required) condition that $C_1 \geq 1$.

Case 5. Vertex q is the midpoint of some arc and the nearest neighbor to q is an endpoint of a.

Let p be an endpoint of the arc a and let y be a point encroaching a which caused a to be split. Let c be the circumcenter of $B(a)$.

If y lies on an input feature, then

$$\text{lfs}(q, \hat{\mathcal{C}}) \leq |q - y| \leq |q - c| + |c - y| \leq 2|p - c| \leq 2|p - q| \leq 2r_q. \tag{5.7}$$

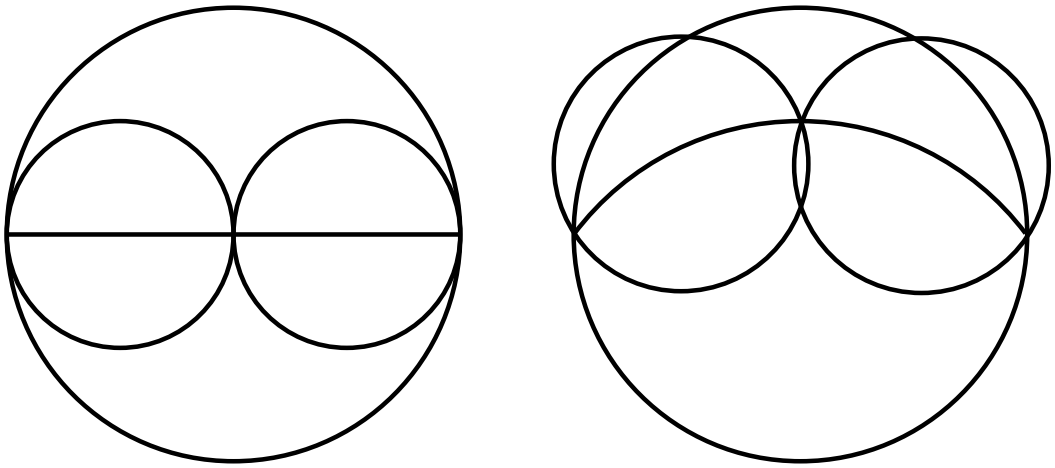


Figure 5.7: Diametral balls nest properly when segments are split, while chordal balls do not.

Thus it is sufficient to require that

$$C_1 \geq 2. \tag{5.8}$$

Otherwise, y is a (possibly rejected) circumcenter of a poor quality triangle.

$$\begin{aligned}
\text{lfs}(q) &\leq \text{lfs}(y) + |q-y| \\
&\leq C_2 r_y + |q-y| \\
&\leq \sqrt{2} C_2 |p-c| + |q-y| \\
&\leq \sqrt{2} C_2 |p-q| + |q-y| \\
&\leq (\sqrt{2} C_2 + 2) r_q
\end{aligned}$$

The fact that $|q-y| \leq 2|q-p|$ follows from the same reasoning as in (5.7). This gives a restriction that

$$C_1 \geq \sqrt{2} C_2 + 2 \tag{5.9}$$

which is strictly stronger than (5.8).

Case 6. Vertex q is the midpoint of some arc and the nearest neighbor to q is a circumcenter x in the mesh.

Let arc a lie on a circle centered at input point q_0. Unlike the case of straight line input, an arc may be encroached by a circumcenter in the mesh which was inserted and did not yield to the (larger) arc that was protected at the time. This can occur because the protected chordal balls do not nest when splitting arcs as seen in Figure 5.7. Let a_θ be the arc containing a in the mesh **when x was inserted** into the mesh. So, $x \in B(q, |q-p|) \setminus B(a_\theta)$ where p is an endpoint of a. Let a_θ be the nearest endpoint of a_θ to x and let θ be the angle subtended by a_θ.

Let $\beta = \frac{|x-q|}{|q-q_0|}$. Now consider two cases depending on the size of β. Let $\beta^*_{\frac{\sqrt{2}+\tau}{2}}$ be the constant given in Proposition 5.9.

First, suppose that $\beta > \beta^*_{\frac{\sqrt{2}+\tau}{2}}$. Now, we estimate the local feature size at q using the Lipschitz property and the associated input point q_0:

$$\begin{aligned}\text{lfs}(q, \hat{\mathcal{C}}) &\leq |q - q_0| + \text{ lfs}(q_0, \hat{\mathcal{C}}) \\ &\leq 2|q - q_0| \\ &\leq \frac{2}{\beta^*_{\frac{\sqrt{2}+\tau}{2}}}|x - q| \\ &\leq \frac{2}{\beta^*_{\frac{\sqrt{2}+\tau}{2}}} r_q.\end{aligned}$$

We must require that

$$C_1 \geq \frac{2}{\beta^*_{\frac{\sqrt{2}+\tau}{2}}}. \tag{5.10}$$

Next, suppose that $\beta \leq \beta^*_{\frac{\sqrt{2}+\tau}{2}}$. We seek to apply Proposition 5.9 and assert that

$$\frac{|x - p|}{|x - q|} \leq \frac{\sqrt{2} + \tau}{2}.$$

However, x is not necessarily on the boundary of $B(a)$ and q is not necessarily the projection of x onto a. If $\frac{|x-p|}{|x-q|} \leq 1$ the desired estimate holds. Otherwise, let q' be the projection of x onto a_θ, and let $x' = \overline{q'x} \cap \partial B(a_\theta)$ as shown in Figure 5.8. Then,

$$\frac{|x - p|}{|x - q|} \leq \frac{|x - p|}{|x - q'|} \leq \frac{|x' - p|}{|x' - q'|}.$$

Moreover,

$$\frac{|x' - q'|}{|q' - q_0|} \leq \frac{|x - q|}{|q - q_0|} \leq \beta^*_{\frac{\sqrt{2}+\tau}{2}}$$

and thus applying Proposition 5.9 (using x' and q') implies that

$$\frac{|x - p|}{|x - q|} \leq \frac{|x' - p|}{|x' - q'|} \leq \frac{\sqrt{2} + \tau}{2}.$$

Now the local feature size can be estimated:

$$\begin{aligned}\text{lfs}(q, \hat{\mathcal{C}}) &= |x - q| + \text{ lfs}(x, \hat{\mathcal{C}}) \\ &\leq |x - q| + C_2 r_x \\ &\leq |x - q| + C_2|x - p| \\ &\leq |x - q| + C_2 \frac{\sqrt{2} + \tau}{2}|x - q| \\ &= \left(1 + C_2 \frac{\sqrt{2} + \tau}{2}\right) r_q.\end{aligned}$$

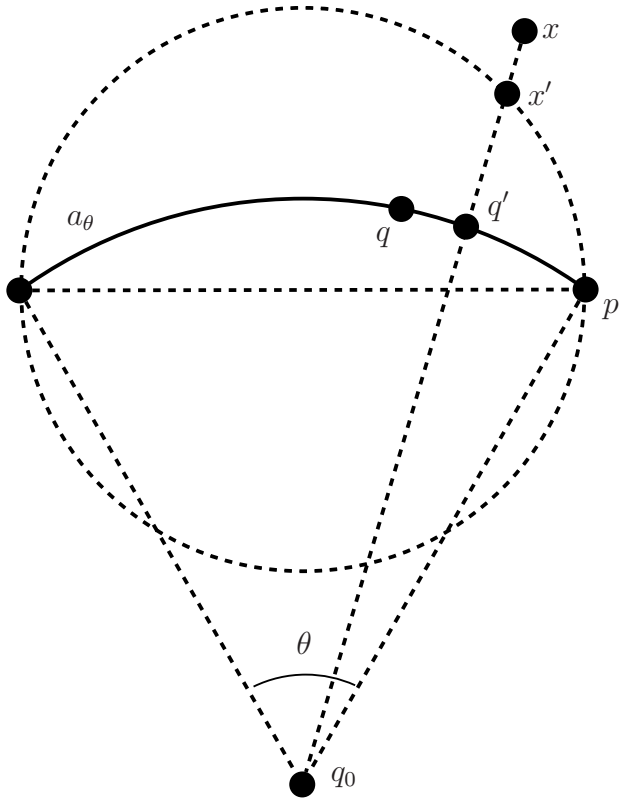


Figure 5.8: Diagram for Theorem 5.10.

Then, we require $C_1 \geq 1 + C_2 \frac{\sqrt{2}+\tau}{2}$. This restriction and several previous ones, (2.2), (5.8), and (5.9), can all be satisfied if

$$C_1 \geq 2 + C_2 \frac{\sqrt{2}+\tau}{2}. \tag{5.11}$$

Thus, it remains to find constants satisfying the three remaining conditions: (2.1), (5.10), and (5.11). A valid choice of C_1 and C_2 is given below:

$$C_2 := \max\left(\frac{4+2\tau}{\tau-\sqrt{2}}, \frac{4-2\beta^*_{\frac{\sqrt{2}+\tau}{2}}}{\beta^*_{\frac{\sqrt{2}+\tau}{2}}(\sqrt{2}+\tau)}\right)$$

$$C_1 := 1 + C_2 \frac{\sqrt{2}+\tau}{2}.$$

This completes the proof. □

Theorem 5.11. *The resulting Delaunay triangulation conforms to the input. Any remaining poor quality triangles are inside $B(q_0, d_{q_0})$ for some input point q_0. The resulting triangulation contains no angles larger than $\pi - 2\kappa$.*

Proof. These properties are immediate from the definition of the Delaunay refinement algorithm. Note that no triangles contain large angles since all Delaunay triangles inside the protected disks are acute although they can be arbitrarily small if the input contains small angles. □

Chapter 6

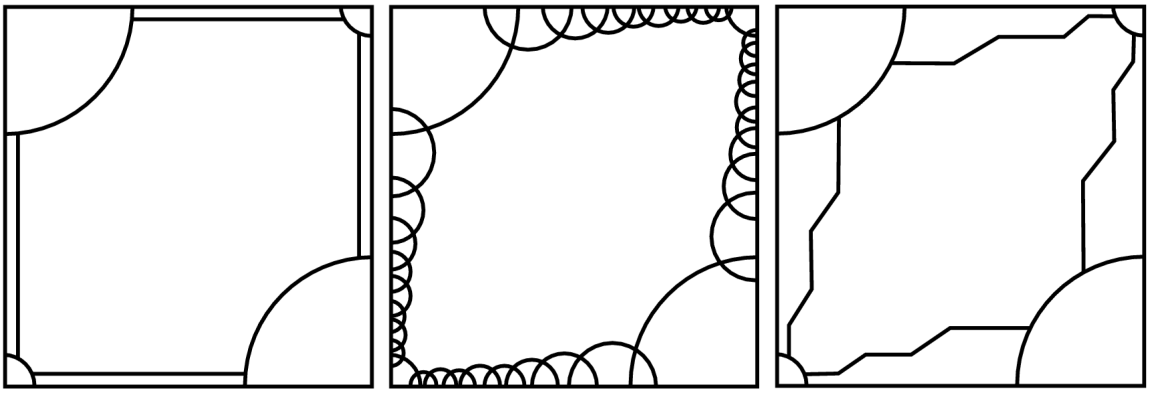
Quality, Conforming Tetrahedralization

Generating conforming meshes of an arbitrary PLC in 3D is substantially more complex than the 2D analog. This stems from the fact that only segments near input points need to be protected in 2D while in 3D a consistent strategy must be employed along edges to ensure the conformity of all adjacent faces.

The first algorithm for computing conforming Delaunay tetrahedralizations of a general 3D PLC was described by Murphy, Mount and Gable [34]. In addition to protecting spheres around input points, this algorithm selects a protection size for each input segment based on the minimum local feature size along the segment and determines a rectangular buffer region in each adjacent face to be triangulated identically. Cohen-Steiner, Colin de Verdière, and Yvinec [17] developed an alternative approach based on packing spheres around segments with the improvement that the size of the protected region is proportional to the 1-feature size at nearby points on a segment. This allows the size of the protection region to vary along segments with the local feature size. These first two algorithms describe "collar" type protection schemes but were not integrated with quality Delaunay refinement. The first algorithm for quality Delaunay refinement of arbitrary input in 3D was given by Cheng and Poon [10, 11]. They proposed an "intestine" based protection scheme based on a sphere packing over the 1-skeleton which is similar to one used by Cohen-Steiner, Colin de Verdière, and Yvinec. Pav and Walkington [36] also designed a collar based approach for quality mesh generation. This approach had the advantage that it was not necessary to pre-compute local feature size before performing the refinement.

Of the algorithms discussed, only one has been implemented. The algorithm of Cohen-Steiner, Colin de Verdière, and Yvinec was implemented and several examples of complex conforming Delaunay triangulations are given. The others have not been implemented and implementation would be a substantial task for any of them. Other algorithms for implementing Delaunay refinement in 3D which allow acute angles rely on constrained Delaunay tetrahedralization [43, 44] or weighted Delaunay refinement [9].

We will given an alternative protection procedure which is a graded to 1-feature size. In faces near edges, rectangles (or trapezoids in certain cases) are protected rather than spheres and the resulting mesh can be viewed as a 1-feature size graded version of the original Murphy, Mount



(a) Protection region in [34] (b) Protection region in [10, 17, 36] (c) New protection region

Figure 6.1: Different Collar Protection Strategies

and Gable protection procedure. Figure 6.1 depicts these different protection strategies. This protection strategy can be used in both collar and intestine type Delaunay refinement procedures. In each case, there are two steps, Step 2c and Step 3. The first involves inserting points to build the initial protection region, while the second is a Delaunay refinement algorithm which is designed to preserve conformity of the protected region. The numbering of these steps (2c and 3) is due to the fact that they naturally follow Algorithm 4.1.

Algorithm 6.1 Quality Refinement of Acute Input 3D

(Step 2c) Protect acutely adjacent input features.

(Step 3) Perform a protected quality Delaunay refinement.

6.1 Properties of Local Feature Size

Several of the results in Section 5.1 are generalized to 3D. Proofs are only given when the 2D proofs cannot be immediately extended to the 3D case. First are the extensions of Proposition 5.1 and Proposition 5.2.

Proposition 6.1. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ be a PLC and let $\bar{\mathcal{C}} = (\bar{\mathcal{P}}, \bar{\mathcal{S}}, \bar{\mathcal{F}})$ be a refinement of $\mathcal{C}$. Let $\bar{\mathcal{C}}_s$ be some (possibly trivial) subcomplex of $\bar{\mathcal{C}}$. If*

$$\operatorname{lfs}(y, \mathcal{C}) \leq K \operatorname{lfs}(y, \bar{\mathcal{C}}_s)$$

for all points y which belong to some feature of $\bar{\mathcal{C}}_s$ of dimension at most $\dim(\bar{\mathcal{C}}_s) - 1$, then

$$\operatorname{lfs}(x, \mathcal{C}) \leq (2K + 1) \operatorname{lfs}(x, \bar{\mathcal{C}}_s)$$

holds for all x.

Proposition 6.2. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ and let $\hat{\mathcal{C}} = (\mathcal{P} \cup \hat{\mathcal{P}}, \mathcal{S} \cup \hat{\mathcal{S}}, \mathcal{F} \cup \hat{\mathcal{F}})$ be a piecewise smooth complex. Suppose that for all $y \in \hat{\mathcal{P}}$, $y \in s \in \hat{\mathcal{S}}$, or $y \in f \in \hat{\mathcal{F}}$,*

$$\operatorname{lfs}(y, \mathcal{C}) \leq K \operatorname{lfs}(y, \hat{\mathcal{C}})$$

then for all x

$$\operatorname{lfs}(x, \mathcal{C}) \leq (4K + 3) \operatorname{lfs}(x, \hat{\mathcal{C}}).$$

Lemma 5.3 will have two analogous 3D versions. The first corresponds to a refinement of the segments of a PLC and follows an identical proof as the 2D version. The second involves the refinement of faces of the PLC.

Lemma 6.3. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ be a PLC with refinement $\bar{\mathcal{C}} = (\mathcal{P} \cup \bar{\mathcal{P}}, \bar{\mathcal{S}}, \mathcal{F})$ such that*

$$\bar{\mathcal{P}} \subset \bigcup_{s \in \mathcal{S}} s^{\circ} \tag{6.1}$$

where s° is the relative interior of segment s. For a segment $s \in \mathcal{S}$, let $\bar{\mathcal{C}}_s$ be the PLC containing all features of $\bar{\mathcal{C}}$ which are contained in s. If for any segment $s \in \mathcal{S}$

$$\operatorname{lfs}(q, \mathcal{C}) \leq K \operatorname{lfs}(q, \bar{\mathcal{C}}_s)$$

holds for any vertex $q \in s \cap \bar{\mathcal{P}}$, then

$$\operatorname{lfs}(x, \mathcal{C}) \leq \left(\frac{4K + 4}{\sin \alpha_1} + 4K + 3 \right) \operatorname{lfs}(x, \bar{\mathcal{C}})$$

holds for all x.

Lemma 6.4. *Let $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ be a PLC with refinement $\bar{\mathcal{C}} = (\mathcal{P} \cup \bar{\mathcal{P}}, \mathcal{S} \cup \bar{\mathcal{S}}, \bar{\mathcal{F}})$ such that*

$$\bar{\mathcal{P}} \subset \bigcup_{f \in \mathcal{F}} f^{\circ}$$

and for all $\bar{s} \in \bar{\mathcal{S}}$,

$$\bar{s} \subset \bigcup_{f \in \mathcal{F}} f^{\circ}$$

where f° is the relative interior of face f. For a face $f \in \mathcal{F}$, let $\bar{\mathcal{C}}_f$ be the PLC containing all features of $\bar{\mathcal{C}}$ which are contained in f. If for any face $f \in \mathcal{F}$

$$\operatorname{lfs}(y, \mathcal{C}) \leq K \operatorname{lfs}(y, \bar{\mathcal{C}}_f)$$

holds for any $y \in f$ which belongs to $\bar{\mathcal{P}}$ or a segment of $\bar{\mathcal{S}}$, then

$$\operatorname{lfs}(x, \mathcal{C}) \leq \left(\frac{4K + 4}{\sin \alpha_2} + 4K + 3 \right) \operatorname{lfs}(x, \bar{\mathcal{C}})$$

holds for all x.

The proof of Lemma 6.4 is also very similar to the proof of Lemma 5.3. Below, the key differences are described.

Proof. The first step is to show that

$$\operatorname{lfs}(y, \mathcal{C}) \leq (2K+1) \operatorname{lfs}(y, \bar{\mathcal{C}}_f)$$

for any $y \in \mathcal{P}$ and $y \in s \in \mathcal{S}$. This is a result of the same argument seen previously. Following the proof of Lemma 5.3, consider any point u contained a face of $\mathcal{F}$ and let w be the point on a feature of $\bar{\mathcal{C}}$ which is disjoint from the feature containing u which realizes the local feature size. This gives three cases:

1. u and w lie on disjoint features of $\mathcal{C}$.
2. u and w are contained in a single face of $\mathcal{C}$.
3. u and w belong to adjacent faces if $\mathcal{C}$.

The first three cases have identical proofs to those of Lemma 5.3 with one exception: the minimum angle between face α_2 is used in the third case. □

6.2 Collar Protection Region

6.2.1 Step 2c

Protect acutely adjacent input features.

Fix $b > 0$ and $c < 1$. Given a PLC $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$, we will assume that we have a refinement $\mathcal{C}_1 = (\mathcal{P}_1, \mathcal{S}_1, \mathcal{F})$ such that

(H1) $(\mathcal{P}' \setminus \mathcal{P}) \setminus (\cup_{s \in \mathcal{S}} s) = \emptyset$,

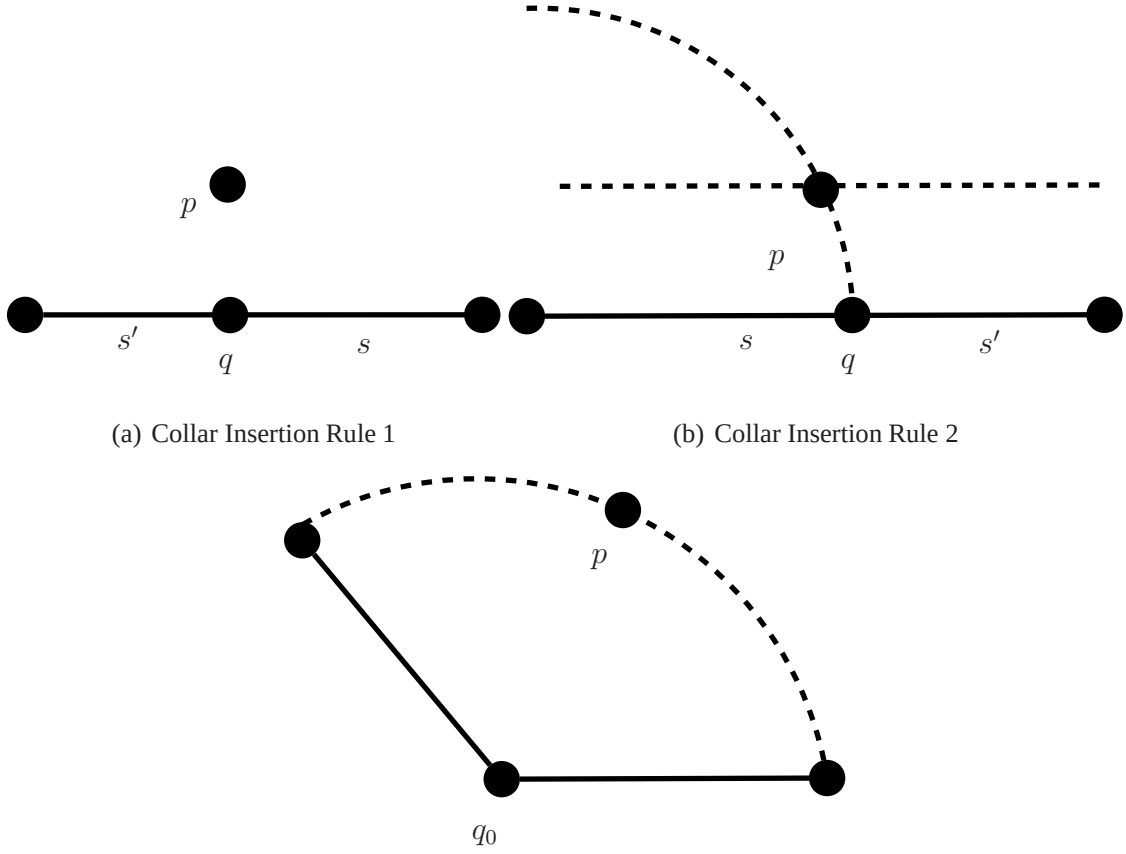
(H2) for all $s' \in \mathcal{S}'$, $b \cdot \min(\operatorname{fs}_1(s), \operatorname{lfs}(s)) < |s| < c \cdot \min(\operatorname{fs}_1(s), \operatorname{lfs}(s))$, and

(H3) all adjacent end segments have equal length.

The purpose of these requirements is to ensure that all subsegments have lengths comparable to the appropriate feature size. This is important since segment length will be used to determine the size of the collar region produced. If this size is not sufficiently small, different sections of the collar may be "tangled".

A suitable refinement can be computed by performing Algorithm 4.1 in Chapter 4 and only considering points which lie on the 1-skeleton of the input. This algorithm yields the desired upper bound with $c = \frac{5}{3}$. Since we will require $c < 1$, this requires an additional split of each segment and potentially some simple local clean up of new non-end segments.

The collar is formed by inserting points in each face according to the following rules which are depicted in Figure 6.2.



(a) Collar Insertion Rule 1

(b) Collar Insertion Rule 2

(c) Collar Insertion Rule 3

Figure 6.2: Collar insertion rules

1. If s and s' are adjacent non-end segments which meet at point q, then a point p is inserted at distance $\frac{\max(|s|,|s'|)}{2}$ from q, in any direction into the face perpendicular to s.
2. If s is an end segment and s' is an adjacent non-end segment, both containing q, then insert p at the intersection of any line parallel to s in the face at distance $\frac{|s'|}{2}$ away from s and on the circle of radius $|s|$ around the input point on s.
3. Given any input point q_0 in the face, insert collar points in the face such that the circle of radius $N(q_0)$ around q_0 has no arcs of angle larger than $\frac{\pi}{2}$.

Table 6.1 contains a list of objects defined to describe the collar based on the vertices inserted during this step. These objects are depicted in Figure 6.3. Figure 6.4 gives an example of collar simplices for a simple face. Following the insertion of the collar vertices, the resulting Delaunay tetrahedralization satisfies a number of properties given in the following lemma.

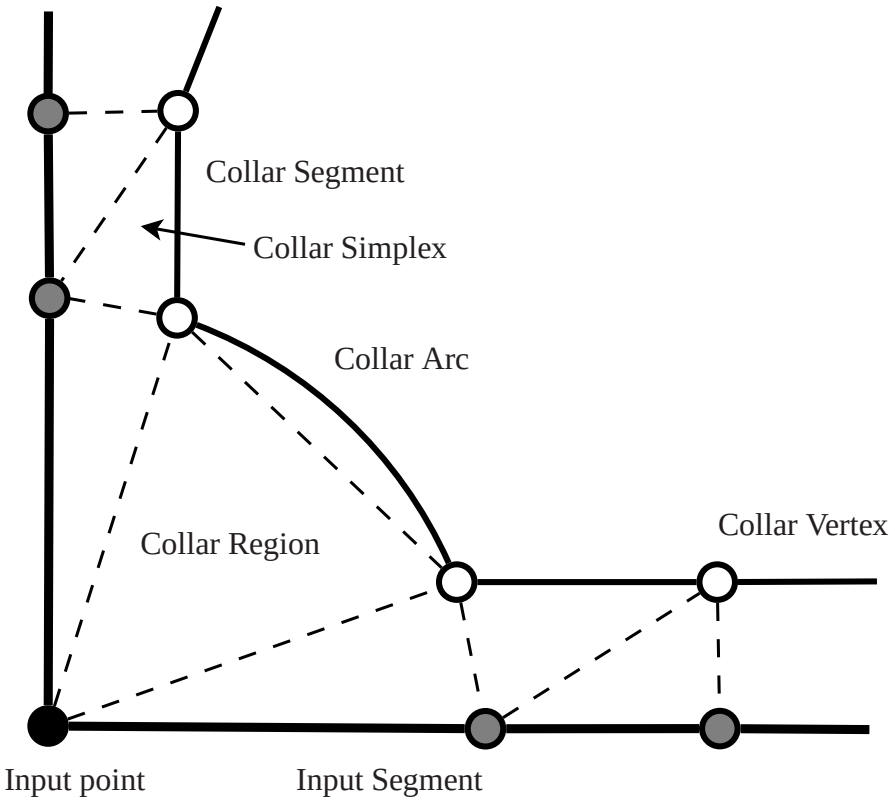


Figure 6.3: Collar Definitions. Black vertices are input vertices, gray vertices are those inserted before Step 2c and the white vertices are those which are inserted during Step 2c.

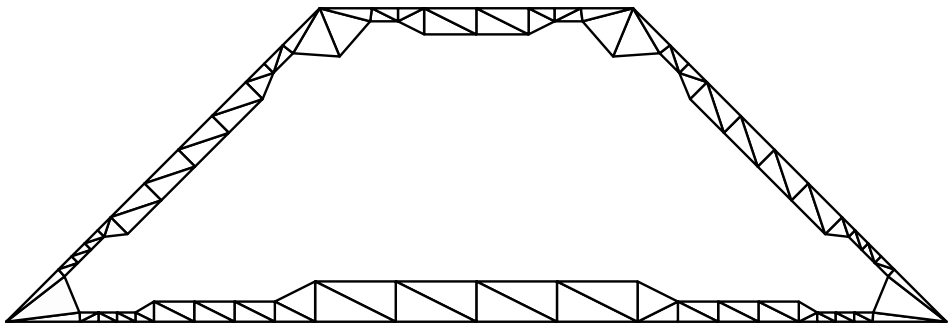


Figure 6.4: Collar simplices in a face.

Table 6.1: Defintions of collar terms.

Collar Vertex	Any vertex inserted during Step 2c.
Collar Segment	Segment between collar vertices which correspond to adjacent vertices on an input segment.
Collar Arc	Arc between adjacent collar vertices corresponding to the same input vertex.
Collar Region	Region between input segments and collar segments and arcs.
Collar Simplex	Any simplex in the Delaunay triangulation of the face which lies inside collar region.

Lemma 6.5. *After Step 2c, the following properties hold.*

I All adjacent collar segments meet at non-acute angles.

II In each face, the diametral disk of each collar segment contains no points of $\mathcal{P}'$.

III The circumball of any collar simplex contains no points of $\mathcal{P}'$.

IV The circumball of any collar simplex does not intersect any disjoint faces or segments.

Proof. Property I follows immediately from the construction. Properties II and IV result from the local feature size bound which we assume the input satisfies. Finally, Property III results from the fact the collar in a face is formed based only upon lengths of subsegments in the associated input segment. Property IV follows from the assumption (H2) on the input complex. □

Since the circumball of each collar simplex is empty, this ensures that the collar simplices conform to the input. Collar segments meet non-acutely and thus the complement of the collar region in each face is well-suited for Ruppert's algorithm. The final property is needed to guarantee that subsequent points inserted in the mesh for conformity will not encroach disjoint collar simplices.

The collar divides each face into two regions: the collar region and the non-collar region. This defines a new piecewise *smooth* complex (since the arcs around input points are curved). The next lemma asserts that this augmented complex preserves the initial local feature size, up to a factor depending on the smallest angles in the input. Recall that α_1 is the smallest angle between an input segment and another adjacent input feature in the mesh and α_2 is the smallest angle between adjacent input faces.

Lemma 6.6. *Let $\bar{C}$ be the PSC consisting of each face divided into the collar region and the non-collar region and requiring all collar segments and collar arcs. Then there exists a constant $k > 0$ depending only upon b and c such that*

$$\text{lfs}(x, \mathcal{C}) \geq \text{lfs}(x, \bar{\mathcal{C}}) \geq k \sin\alpha_1 \sin\alpha_2 \, \text{lfs}(x, \mathcal{C}).$$

Proof. The first inequality is immediate for any refinement. For the second inequality, consider an additional intermediate PLC, $\mathcal{C}_0$. Let $\mathcal{C}_0 = (\mathcal{P}_0, \mathcal{S}_0, \mathcal{F})$ where $\mathcal{P}_0$ contains $\mathcal{P}$ and all vertices

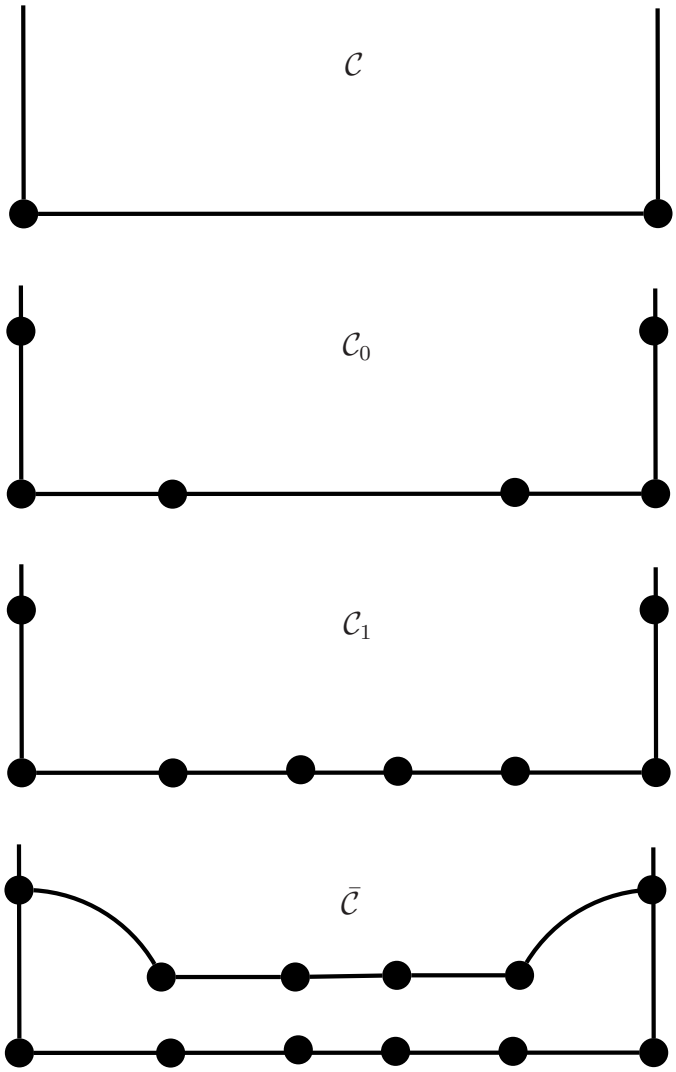


Figure 6.5: Intermediate PLCs described in Lemma 6.13 for a partial example face.

of $\mathcal{P}_1$ which are adjacent to some input vertex in $\mathcal{P}$ on a segment of $\mathcal{S}_1$. Recall that $\mathcal{P}_1$ is the refined PLC which satisfies the assumptions H1-3. Then $\mathcal{S}_0$ is the refined segments of $\mathcal{S}$ based on the vertices in $\mathcal{P}_0$. Figure 6.5 depicts the various intermediate PLCs which have been defined. The result will follow by arguing the following list of inequalities.

$$\text{lfs}(x, \mathcal{C}_0) \geq k_0 \sin \alpha_1 \, \text{lfs}(x, \mathcal{C}) \tag{6.2}$$

$$\text{lfs}(x, \mathcal{C}_1) \geq k_1 \, \text{lfs}(x, \mathcal{C}_0) \tag{6.3}$$

$$\text{lfs}(x, \bar{\mathcal{C}}) \geq k_2 \sin \alpha_2 \, \text{lfs}(x, \mathcal{C}_1) \tag{6.4}$$

Lemma 6.3 implies (6.2) and Lemma 6.4 implies (6.4). Finally (6.3) follows from the fact that no acutely adjacent features are refined between $\mathcal{C}_0$ and $\mathcal{C}_1$. □

6.2.2 Step 3

Perform a protected quality Delaunay refinement.

In the final step, the volume mesh is refined based on both quality and conformity criteria using Ruppert's algorithm. Similar to the non-acute case, any maximum radius-edge threshold $\tau > 2$ can be selected for determining poor quality tetrahedra. The Delaunay refinement algorithm is specified in Algorithm 6.2.

Algorithm 6.2 3D Delaunay Refinement With Collar

Action	When a simplex is processed, its circumcenter is inserted, unless it encroaches upon the diametral ball of a lower dimensional simplex, collar segment, or collar arc. In this case, the encroached simplex is queued for splitting. When an collar segment or collar arc is processed, insert the midpoint.
Priority	Collar segments and arcs are given the highest priority. Simplices are prioritized by dimension with lower dimensional simplices processed first.
Unacceptability	A simplex, collar segment or collar arc is unacceptable if it has a nonempty circumsphere. A tetrahedron is unacceptable if its radius-edge ratio is larger than τ.
Safety	It is not safe to split any collar simplex (either triangles in input faces or subsegments of input segemtns).

The key difference between Algorithm 6.2 and the 3D version of Ruppert's algorithm is the safety criteria. This prevents acutely adjacent faces from preventing termination of the algorithm.

For collar arcs, note that the arc midpoint (rather than the midpoint between the two ends of the arc) is inserted. See Figure 6.6. This procedure was also used in the 2D intestine protection strategy and will lead to very similar analysis.

During the algorithm, it is important to ensure that the properties of the collar in Lemma 6.12 continue to hold while allowing refinement of the non-collar region of each face to create a conforming mesh. In the 2D collar protection procedure, the protected collar simplices (the end segments) never change during Algorithm 5.2. In 3D however, the set of collar simplices does change. This occurs when the standard Delaunay refinement algorithm seeks to insert a vertex in a face that encroaches upon a collar segment or collar arc. Instead of adding this encroaching vertex, this collar segment or arc is split. This new vertex is considered a collar vertex and the collar segment or arc is broken into two new collar segments or arcs. The collar *region* has not changed but the set of collar *simplices* has changed. Further, this new vertex may encroach upon the circumball of another collar simplex in an adjacent face. In this face, the collar segment associated with this encroached circumball is also split so that the collar simplices on adjacent faces again "line up." So conformity of the mesh is maintained by only splitting the collar

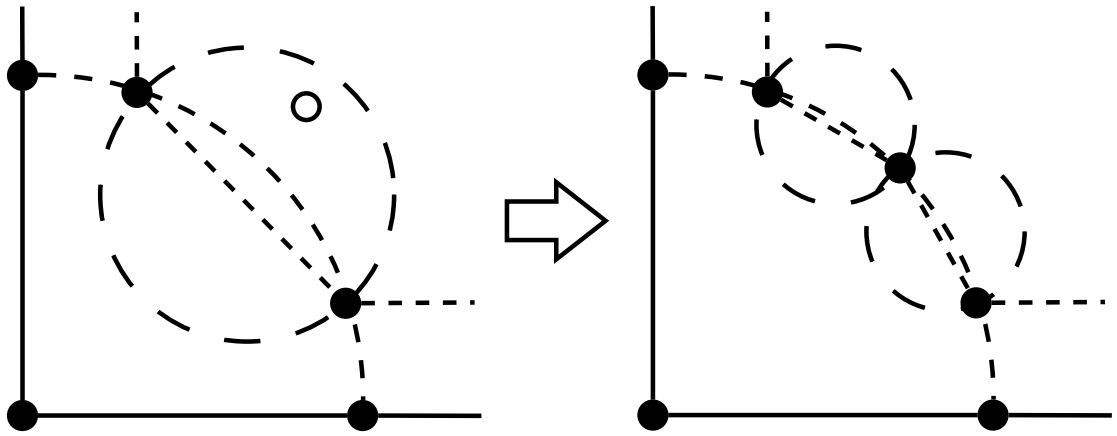


Figure 6.6: A collar segment associated with an input point is split. When a proposed point (denoted by the empty dot) encroaches a collar segment, the segment is split by adding the midpoint of the arc of a circle around the input point.

segments and thus the algorithm never attempts to insert the circumcenter of an encroached collar simplex. This is exactly the purpose of the collar construction: avoid the cascading encroachment sequence between adjacent faces associated with inserting the circumcenters of triangles near adjacent input angles.

Now, we highlight two propositions which are necessary to ensure the correctness of the algorithm. Both follow from the Delaunay property. The first is given in [29] (which gave a *complete* description of the 3D Delaunay refinement algorithm for non-acute input) and provides the natural analogy to the fact that when splitting a segment, the diametral balls of the new subsegments are contained in the diametral ball of the old segment.

Proposition 6.7. *[29, Lemma 4.5] Let $\mathcal{T}$ be the Delaunay triangulation of a planar face F such that the circumball of each bounding segment is empty. Let $\mathcal{B}_1$ be the union of the circumballs of all bounding segments of F and $\mathcal{B}_2$ be the union of the circumballs of all triangles in F.*

Let $p \in F \setminus \mathcal{B}_1$. If $\mathcal{T}'$ is the Delaunay triangulation of the face resulting from the addition of p, define $\mathcal{B}'_2$ to be the union of the circumballs of triangles in the resulting Delaunay triangulation. Then,

$$\mathcal{B}'_2 \subset \mathcal{B}_1 \cup \mathcal{B}_2.$$

This fact is important in ensuring that when additional collar vertices are inserted the resulting collar simplices are not encroached by other vertices added as circumcenters of poor quality triangles. Collar simplices can be encroached by collar vertices in adjacent faces but these encroachments are removed once encroachment of collar segments in adjacent faces are identical (and thus termination can be ensured).

The next proposition motivates the idea behind protecting the collar segments. This ensures that whenever a triangle in a face is processed on the refinement queue, either the circumcenter is valid to be inserted into the mesh or it encroaches a collar segment, which will be queued

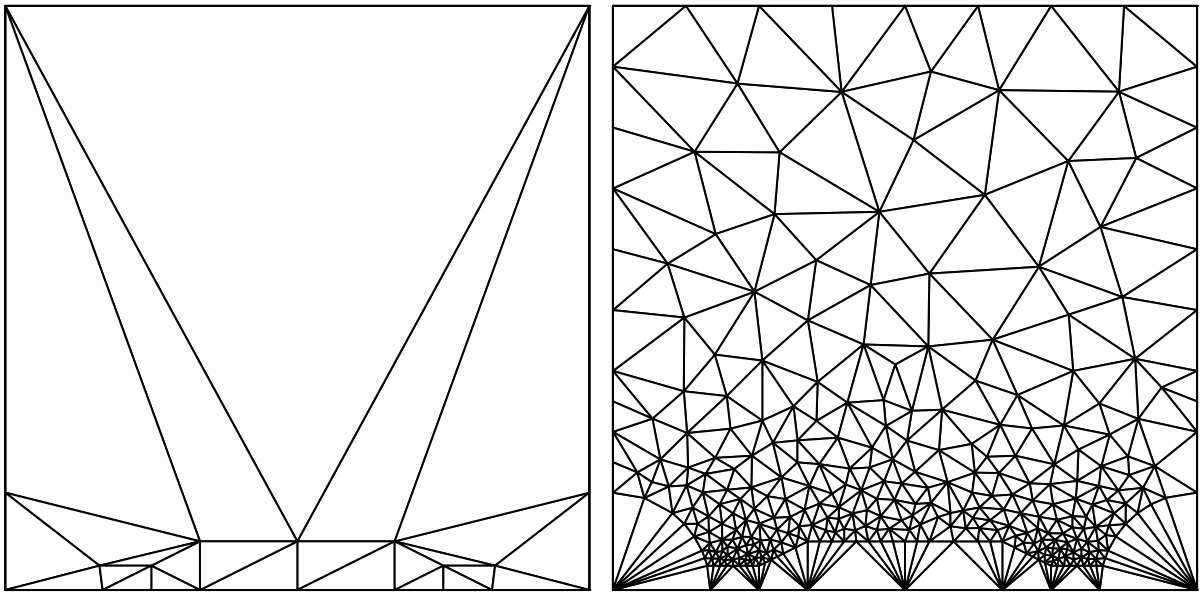


Figure 6.7: Refinement of the non-collar region of a face.

for splitting. This is necessary for ensuring conformity of the mesh upon termination of the algorithm.

Proposition 6.8. *Let A be the set of vertices on the boundary of some face F and let A' be a set of vertices inside F. Suppose that for each boundary segment of F there is a circle through the end points of the segment which does not contain any vertices of $A \cup A'$ in its interior. Then the Delaunay triangulation of $A \cup A'$ conforms to F. Moreover, for any Delaunay triangle t in the interior of F, the circumcenter of t either lies inside F or inside the empty disk associated with a boundary segment.*

Now we can state the key properties which hold throughout the algorithm whenever there are no unacceptable collar segments on the queue to split.

Lemma 6.9. *Whenever the queue of unacceptable simplices does not contain any collar segments and collar arcs, the following properties hold.*

I The circumball of any collar element contains no vertices in $\mathcal{P}'$.

II Adjacent collar segments meet at non-acute angles.

Proof. By applying Lemma 6.7 to the collar region of each face, conclude that no circumcenter can encroach a collar simplex. Property I then follows because a collar simplex can only be encroached by a vertex on an adjacent collar segment which also encroaches the corresponding collar segment in the face with the original simplex.

Property II results from the fact that nearly formed collar arcs meet at an angle of at least $\frac{\pi}{2}$ since the original arcs were split to subtend angles of at most $\frac{\pi}{2}$. □

Theorem 6.10. *For any $\tau > 2$, there exists $C > 0$ depending only upon τ, α, b, and c such that*

for each vertex q inserted in the mesh,

$$\text{lfs}(q) \leq Cr_q.$$

Proof. Lemma 6.13 ensures that it is sufficient to prove the inequality

$$\text{lfs}(q, \bar{\mathcal{C}}) \leq Cr_q.$$

This standard proof of termination of Ruppert's algorithm (i.e. the 3D analog to the proof of Theorem 2.2) is used in conjunction with the techniques in the proof of Theorem 5.10 to handle the collar arcs around input points.

Acute angles usually cause the the standard proof to fail. This failure occurs when a simplex (a Delaunay triangle in a face or a subsegment of an input segment) is encroached by a vertex on an adjacent input feature. Lemma 6.9 ensures that this does not occur: collar simplices are never split. If a collar segment is split due to a vertex on another collar segment, this vertex may lie on an adjacent input feature but these collar segments are disjoint features in the PLC $\bar{\mathcal{C}}$ ensuring that the original proof of termination for Delaunay refinement holds. □

Theorem 6.11. *The resulting Delaunay tetrahedralization conforms to the input. Any remaining poor quality tetrahedra encroach collar simplices.*

Proof. These properties are immediate from the description of the algorithm (which ensures remaining poor quality tetrahedra failed the safety criteria) and Lemma 6.9 (which ensures that collar simplices conform to the input). □

6.3 Intestine Protection Region

The intestine approach for protecting acute input angles mirrors that in 2D described in Section 5.3. Smooth features will be added to the input to isolate all input segments and vertices (or at those contained in acutely adjacent features) from the region to be refined for quality.

6.3.1 Step 2c

The vertices and features which are added to the mesh in this step are a superset of those added in Step 2c of the collar approach (the PSC $\bar{\mathcal{C}}$). In addition to the collar vertices, the following features are added to the mesh forming a new complex $\hat{\mathcal{C}}$.

- For each input vertex q_0 which belongs to some segment, let d_{q_0} be the length of all segments containing q_0. Then $\hat{\mathcal{C}}$ includes $\partial B(q_0, d_{q_0})$.
- For each collar segment s, let c be the surface of revolution produced by revolving segment s about its associated input segment. The features s and ∂s are included in $\hat{\mathcal{C}}$.

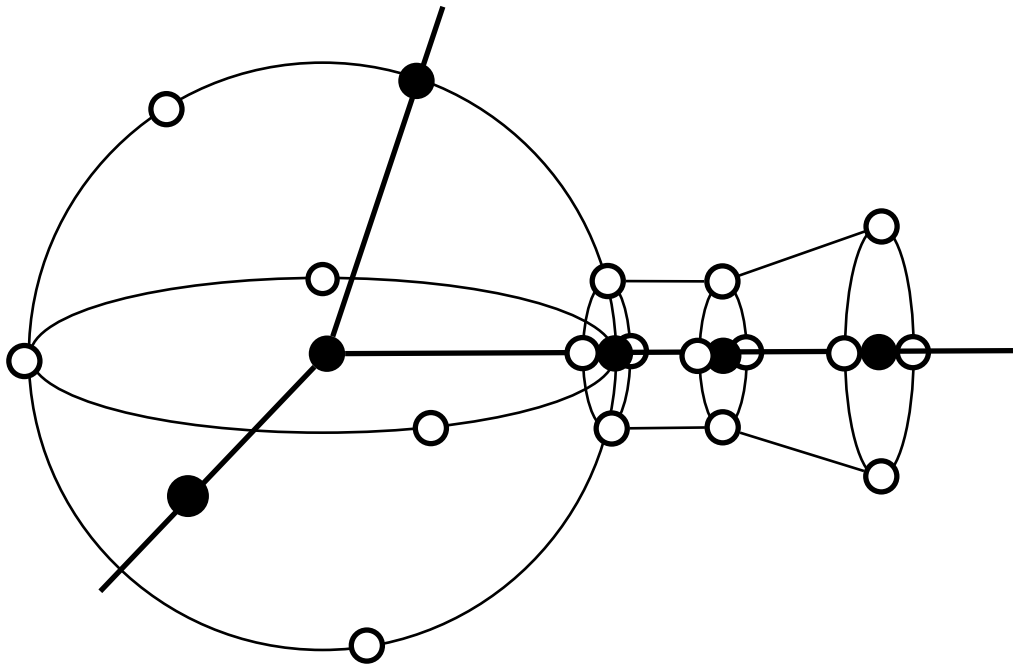


Figure 6.8: Intestine Protection Region

The region inside each sphere and cylindrical surface added to the mesh will be called the intestine region and the remaining volume is called the non-intestine region. This construction is designed to ensure the following fact.

Lemma 6.12. *The non-intestine region of the PSC $\hat{\mathcal{C}}$ contains no acute angles between features.*

This lemma is necessary to ensure that the usual proof of termination and grading will apply to Delaunay refinement in the non-intestine region. However, some issues must be resolved concerning the applicability of this proof to curved input features.

Lemma 6.13. *There exists a constant $k > 0$ depending only upon b and c such that,*

$$\text{lfs}(x, \mathcal{C}) \geq \text{lfs}(x, \hat{\mathcal{C}}) \geq k \, \text{lfs}(x, \bar{\mathcal{C}})$$

where $\bar{\mathcal{C}}$ is the PSC containing the collar.

Proof. This follows from Proposition 6.2. □

6.3.2 Step 3

We now consider two different approaches to performing a quality refinement of the non-intestine region. The first is to perform the usual Delaunay refinement and split smooth surfaces by projecting the circumcenter of any Delaunay triangle in the face to the surface. This is described in Algorithm 6.3. This approach suffers from one minor drawback: the Delaunay tetrahedralization inside the cylindrical regions of the intestine may not conform to the input. Eliminating

this issue, the second approach is to impose more structure on the refinement of these cylindrical regions. This algorithm is described in Algorithm 6.4.

Algorithm 6.3 3D Delaunay Refinement With Intestine - Unstructured

Action	When a simplex is processed, the projection of its circumcenter onto the input feature containing the simplex is inserted, unless it encroaches upon the diametral ball of a lower dimensional feature. In this case, the encroached item is queued for splitting.
Priority	Items are prioritized by dimension, with lower dimensional items processed first.
Unacceptability	A simplex in the non-intestine region is unacceptable if it has a nonempty circumsphere. A tetrahedron is unacceptable if its radius-edge ratio is larger than τ.
Safety	All simplices are safe to split.

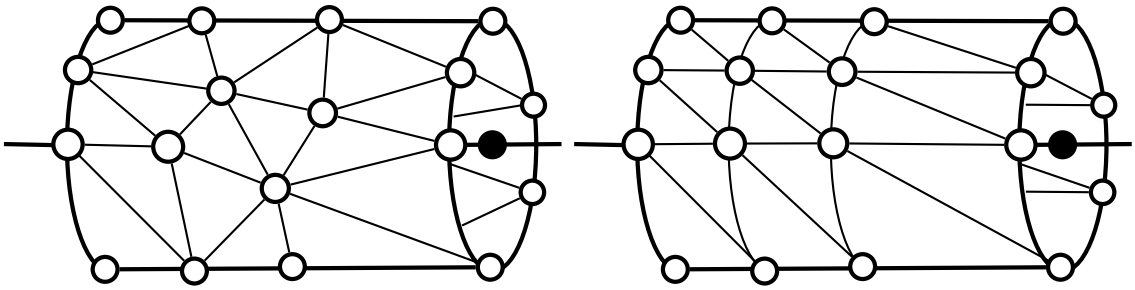
Algorithm 6.4 3D Delaunay Refinement With Intestine - Structured

Action	If a protected triangle on the boundary of a cylindrical region is to be split, add a required circle in the cylinder between the two circles in the PSC associated with the triangle to be split. Add vertices at the intersection of this circle and every face of the original PLC that it intersects and then insert additional vertices so that no arc of the circle is larger than $\frac{\pi}{2}$. When any other simplex is processed, the projection of its circumcenter onto the input feature containing the simplex is inserted, unless it encroaches upon the diametral ball of a lower dimensional feature. In this case, the encroached item is queued for splitting.
Priority	Items are prioritized by dimension, with lower dimensional items processed first.
Unacceptability	A simplex in the non-intestine region is unacceptable if it has a nonempty circumsphere. A tetrahedron is unacceptable if its radius-edge ratio is larger than τ.
Safety	All simplices are safe to split.

Figure 6.9 shows the difference between the refinement around required cylindrical surfaces of the two algorithms.

Algorithm 6.14 produces a conforming Delaunay tetrahedralization of the input. This is shown in the next theorem.

Theorem 6.14. *Upon termination of Algorithm 6.4, the resulting Delaunay tetrahedralization conforms to the input. All tetrahedra outside the intestine region have radius-edge ratio less*



(a) Unstructured Approach of Algorithm 6.3 (b) Structured Approach of Algorithm 6.4

Figure 6.9: Refinement of cylindrical surfaces around the intestine.

than τ.

Proof. The important task is to verify that the Delaunay tetrahedralization conforms to the input inside the intestine region. Because each triangle on the boundary of the collar is protected, no vertex outside of the intestine region can lie in the diametral ball of a required face. By constructing all vertices on cylinders around concentric rings, no vertex on the boundary of the cylinder can encroach upon the diametral ball of a required face either. □

For Algorithm 6.3, the argument above does not hold: a vertex on the boundary of a required cylindrical surface may encroach upon the diametral ball of a required face and this face may not conform to the Delaunay tetrahedralization in the resulting mesh. However, a simple conforming (but not Delaunay) tetrahedralization of the intestine region does exist. The spheres around input vertices are tetrahedralized using the Delaunay tetrahedra. For the cylindrical sections, let p_1 and p_2 be the endpoints of the corresponding input segment. The tetrahedralization is produced with two types of tetrahedra.

- For any Delaunay triangle t on the boundary of the cylinder, include the tetrahedron with base t and vertex at p_1.
- For any Delaunay segment s on the circle around p_2, include the tetrahedra with vertices p_1, p_2 and the endpoints of s.

These tetrahedra are depicted in Figure 6.10. For Algorithm 6.3, a weaker theorem holds.

Theorem 6.15. *Upon termination of Algorithm 6.3, in the non-intestine region the resulting Delaunay tetrahedralization conforms to the input and all tetrahedra ratio less than* τ*. There exists a matching tetrahedralization of the intestine region which conforms to the input PLC.*

We leave the analogous theorem to Theorem 5.10 as a topic of further research.

Conjecture 6.16. *For either Algorithm 6.3 or Algoithm 6.4 and for any* $\tau > 2$*, there exists* $C > 0$ *depending only upon* τ*,* α*,* b*, and* c *such that for each vertex* q *inserted in the mesh,*

$$\text{lfs}(q) \leq Cr_q.$$

To prove this theorem, it is important to carefully generalize the analysis in Case 6 of the proof of Theorem 5.10. In the process, this should yield a condition on input surfaces similar to

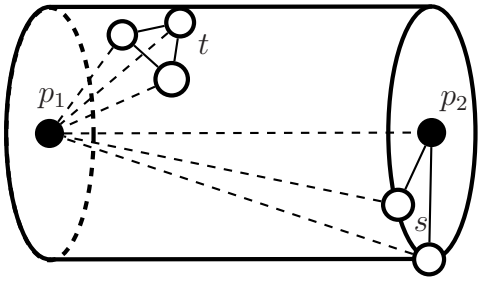


Figure 6.10: Two types of tetrahedra are used inside the intestine region when constructing a conforming tetrahedralization following Algorithm 6.3. This tetrahedralization may not be Delaunay.

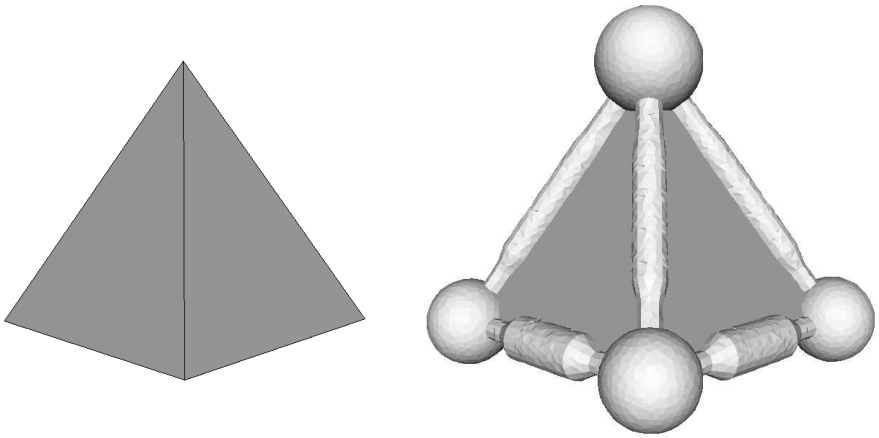
the requirement that arcs initially subtend angles no larger than $\frac{\pi}{2}$. A condition of this type which matches the number of vertices used in practice has not been found.

6.4 Examples

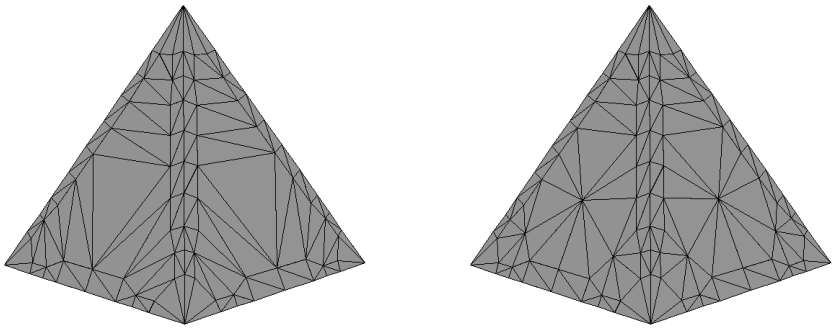
Algorithm 6.3 (rather than Algorithm 6.4) has been implemented and will be referred to as the intestine approach in the examples below.

Example 6.4.1. Figure 6.11 demonstrates both protection strategies on the pyramid PLC input (previously used in Example 4.2.1). Figure 6.12 shows the refinement of a single face of the pyramid during this procedure using the collar. In the face, the intestine approach yields a very similar mesh.

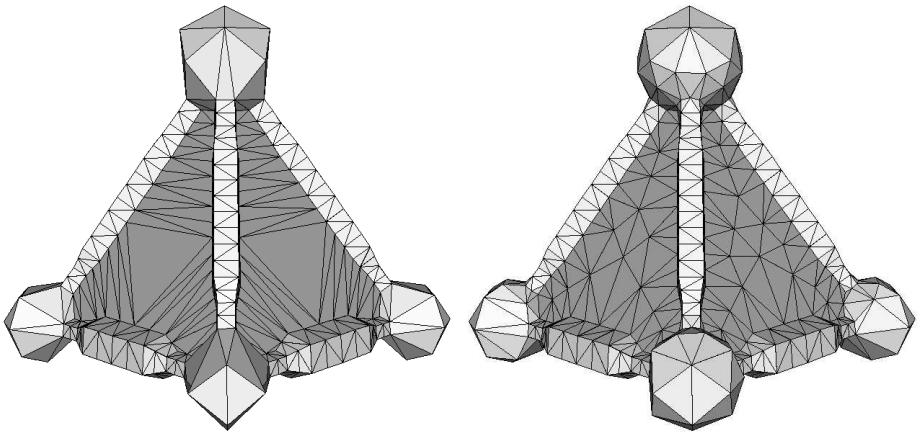
Example 6.4.2. Figure 6.13 demonstrates this algorithm on the wheel PLC from Example 4.2.2. In this example, the only segment that needs to be protected is the segment at the center of the wheel: this is the only segment which is contained in multiple faces. These meshes were produced using the value $\tau = 2.3$.



(a) Initial pyramid PLC and augmented PSC with intestine protection region.



(b) Refinement of the PSC with collar following steps 2c and 3 using the collar protection scheme.



(c) Refinement of the PSC with intestine following steps 2c and 3.

Figure 6.11: Refinement of a simple pyramid.

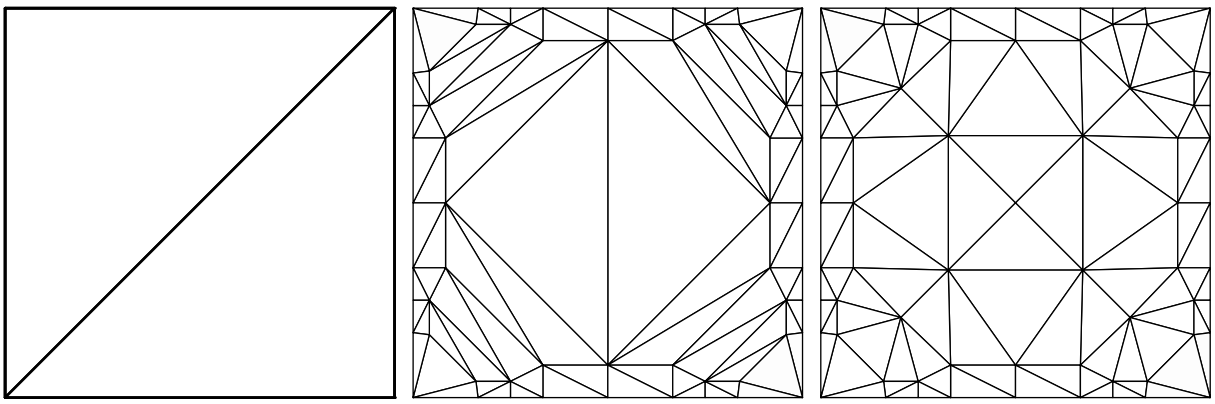
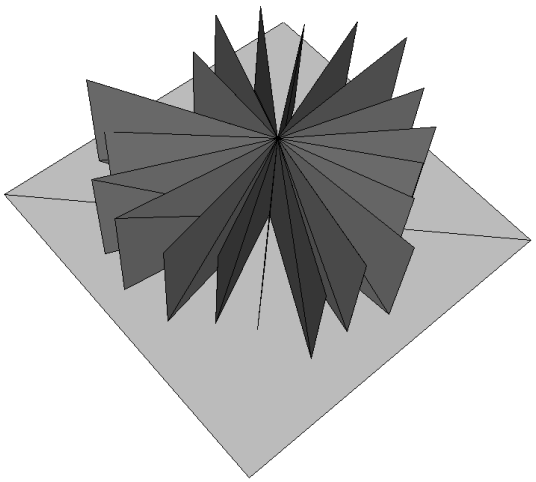
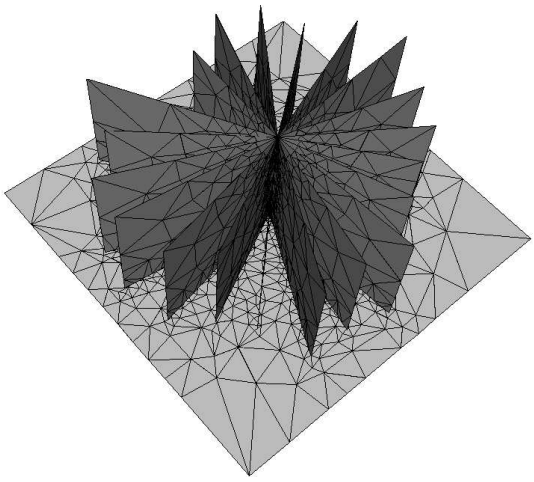


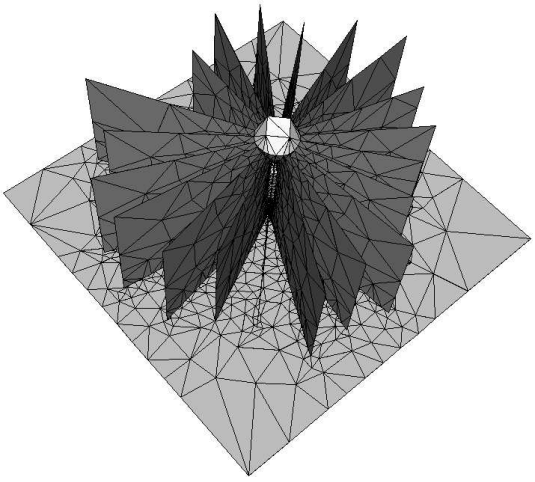
Figure 6.12: Base of the pyramid: initial triangulation, triangulation following Step 2c and final triangulation.



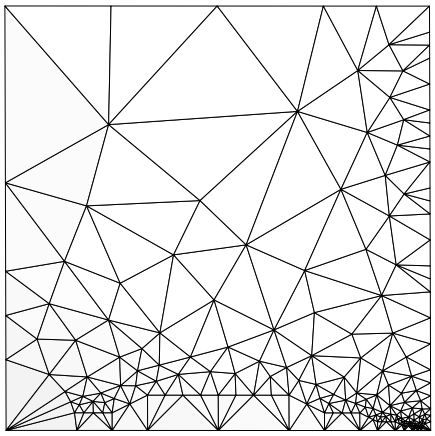
(a) Input PLC



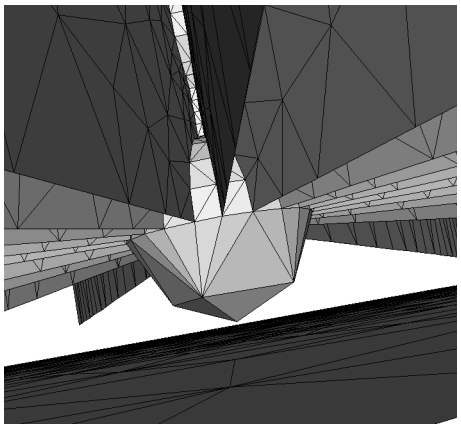
(b) Refinement using collar protection scheme.



(c) Refinement using intestine protection scheme.



(d) Single face in the wheel example using a collar protection scheme



(e) Closer view of the intestine region.

Figure 6.13: Quality refinement of the wheel example.

Chapter 7

Delaunay Refinement and the Finite Element Method

Ruppert's algorithm was designed to produce meshes of bounded aspect ratio triangles. This is a natural geometric criteria used in the traditional convergence analysis of the finite element method. However, for input containing angles smaller than the output angle threshold, it is not possible to to generate a conforming mesh of sufficiently high quality triangles (in the sense of bounded aspect ratio or a minimum angle condition). However, producing a mesh of bounded aspect ratio triangles is not essential for the convergence of the finite element method and thus it is possible to generate appropriate meshes for the finite element method. In this chapter, we discuss the relationship between types of meshes which are produced by Delaunay refinement algorithms and those are needed to produce convergence finite element methods.

In 1976, two similar papers were published describing proofs to replace the traditional minimum angle condition (or bounded aspect ratio requirement) with the maximum angle condition. The paper of Babuška and Aziz demonstrated the an interpolation inequality for linear triangular finite elements with maximum angles bounded away from π [2] (showing that triangulations with very small angles as in Figures 7.1(a) and 7.1(c) are acceptable). They also described a procedure to extend this result to both Lagrange and Hermite finite elements of higher degree, but all analysis was restricted to two dimensions. Jamet provided a more general approach to the maximum angles condition [22]. His analysis gives some general conditions on interpolation operators which ensure the desired interpolation inequalities. In the case of triangular finite elements, this result yields the correct scaling of the error as the maximum angle of the triangle approaches π (i.e. it yields the correct error estimates for triangulations in the form of Figure 7.1(b) where the analysis of Babuška and Aziz fails). However, Jamet's analysis only applies to sufficiently high order finite elements: we must have polynomials of degree $k > \frac{n}{2}$ where n is the dimension of the domain. In two and three dimensions, this eliminates linear elements. Guattery, Miller and Walkington gave an alternative approach which yields the desired estimate for linear elements in 2D [19].

Extension of Delaunay refinement algorithms to accept acute input began first by understand-

ing how to guarantee termination of the algorithm and only then considered interpolation consequences of the results. The first technique for handling acute input angles during 2D Delaunay refinement was the terminator algorithm of Shewchuk [41] and did not provide any guarantee on the maximum angle of the resulting triangulation. Later, the strategy for handling small angles developed by Miller, Pav, and Walkington [30, 35] ensured that any triangle which does not satisfy the desired minimum output angle threshold is acute and thus does not contain large angles. This leads to a bound on the maximum angle in the resulting mesh $(180 - 2\kappa)^\circ$ where κ° is the minimum angle threshold for Delaunay refinement. 2D Delaunay refinement using the intestine protection scheme (Algorithm 5.3) also shares this property.

Inspired by the maximum angle condition, Miller, Phillips and Sheehy consider the problem of output size-competitive meshes in the class of all conforming triangulations which do not contain any large angles [31] (as opposed to the class of "nearly bounded radius-edge" meshes which is considered in the case of Delaunay refinement). In many cases, their algorithm generates smaller meshes than Ruppert's algorithm, but no algorithm has been shown to produce constant factor competitive output in the no large angle setting.

After setting the standard notation, we describe the results of Babuška and Aziz and those of Jamet. We extend the proof of Babuška and Aziz to yield an idential result as in [19]. Further, the interpolation estimate is connected to a geometric quantity: the circumradius. We also discuss the use of average interpolation to derive estimates when the function to be interpolated possesses less regularity than required in the traditional theory and show how estimates of this form can be found independent of geometric restrictions. Next, we describe a very simple Delaunay refinement algorithm which produces suitable finite element meshes given a user sizing function.

7.1 Sobolev Spaces

Throughout this chapter $\Omega \subset \mathbb{R}^2$ is an open, bounded, polygonal set. We will utilize the standard multi-index notation and Sobolev spaces. A multi-index α is an n-tuple of non-negative integers $\alpha = (\alpha_1, ...\alpha_n)$. Let $|a| := \sum_{i=0}^{n} a_n$ and

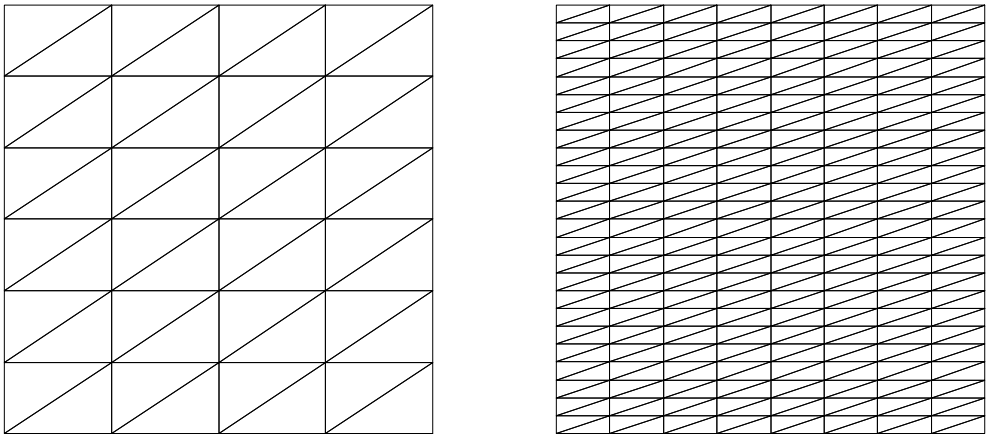
$$\left(\frac{\partial}{\partial x}\right)^\alpha := \frac{\partial^{|\alpha|}}{(\partial x_1)^{\alpha_1} \cdots (\partial x_n)^{\alpha_n}}.$$

The $H^k(\Omega)$-seminorm of a function u is defined by

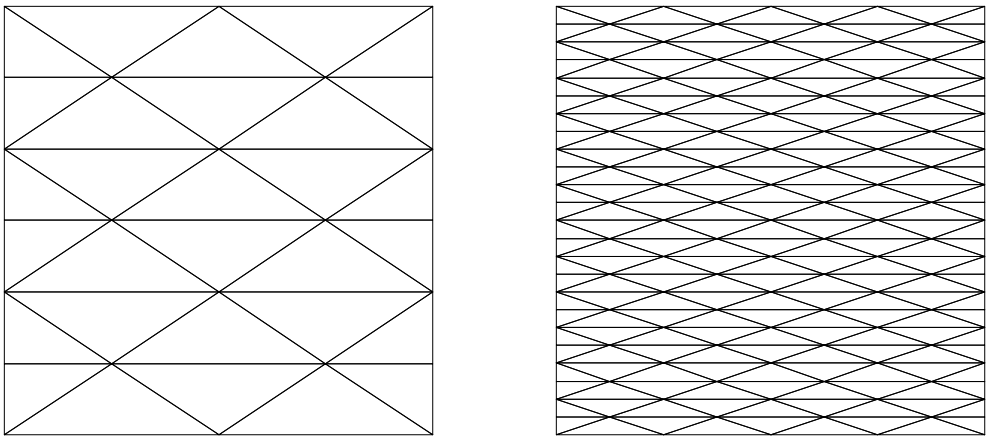
$$|u|^2_{H^k(\Omega)} := \sum_{|\alpha|=k} \int_\Omega \left| \left(\frac{\partial}{\partial x}\right)^\alpha u(x) \right|^2 dx.$$

Then the $H^k(\Omega)$-norm of a function u is defined by

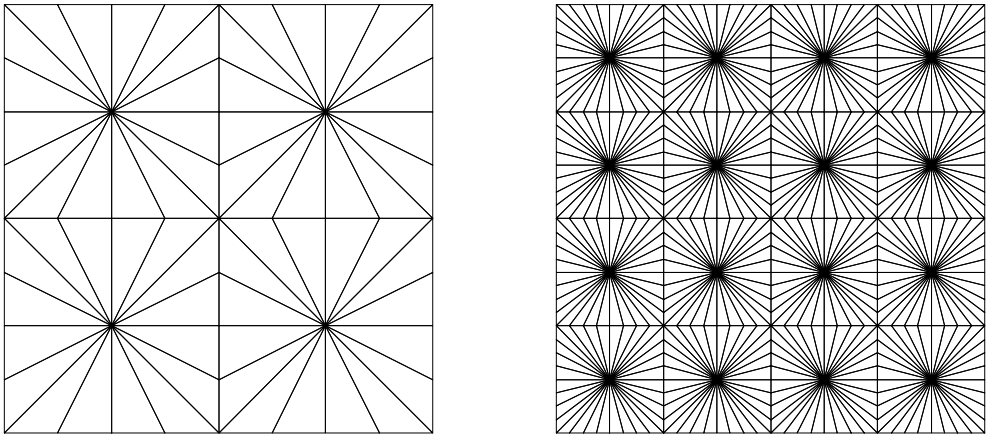
$$||u||^2_{H^k(\Omega)} := \sum_{|\alpha|\leq k} \int_\Omega \left| \left(\frac{\partial}{\partial x}\right)^\alpha u(x) \right|^2 dx.$$



(a) A mesh refinement with narrow triangles but without large angles.



(b) A mesh refinement with narrow triangles which contains large angles.



(c) A mesh refinement with narrow triangles and high degree vertices.

Figure 7.1: Several mesh refinements which do not preserve aspect ratio.

In the $k = 0$ case, this becomes the standard L^2 norm.

$$||u||^p_{L^p(\Omega)} := \int_\Omega |u(x)|^p \, dx.$$

We will also need to consider the mollification of Sobolev functions. Let $\rho \in C^\infty(\mathbb{R}^d)$ such that

- supp$(\rho) \subset B(0, 1)$,
- $\int \rho(x) \, \mathrm{d}x = 1$, and
- $\rho(x) \geq 0$ for all x.

Typically, $\rho(x) = ce^{-1/(1-||x||^2)}$ is selected with some constant c such that the integral is 1. Let $\rho_h(x) = \frac{1}{h^N}(\frac{x}{h})$. Then,

- supp$(\rho_h) \subset B(0, h)$,
- $\int \rho_h(x) \, \mathrm{d}x = 1$,
- $\rho_h(x) \geq 0$ for all x, and
- $||\nabla \rho_h||_{L^1(\mathbb{R}^d)} \leq \frac{1}{h} ||\nabla \rho||_{L^1(\mathbb{R}^d)}$

A function is mollified by convolving it with ρ_h. This is a useful technique for regularizing a non-smooth function. Given $u \in L^1(\mathbb{R}^n)$, define

$$\mathcal{M}_h u(x) := \int_{\mathbb{R}^d} u(x - y)\rho_h(y) \, \mathrm{d}y.$$

For a function $u \in H^1(\Omega)$, we will define $\mathcal{M}_h u$ according to the same formula by first extending u to $H^1(\mathbb{R}^d)$ according to a continuous extension operator (which exists for the domains Ω we are considering).

Two standard properties of mollification are given in Proposition 7.1 and will be used in Section 7.2.4.

Proposition 7.1. *There exists $C > 0$ such that for h sufficiently small, following two inequalities hold for all $u \in H^1(\Omega)$:*

$$||u - \mathcal{M}_h u||_{L^2(\mathbb{R}^d)} \leq Ch \, |u|_{H^1(\Omega)} \, , \tag{7.1}$$

$$|\mathcal{M}_h u|_{H^2(\mathbb{R}^d)} \leq \frac{C}{h} \, |u|_{H^1(\Omega)} \, . \tag{7.2}$$

Proof. Let $w \in L^2(\mathbb{R}^d)$ be an arbitrary function. Let $u \in C_c^\infty(\mathbb{R}^d)$.

$$\begin{aligned}
&\int_{\mathbb{R}^N} (u(x) - \mathcal{M}_h u(x)) w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \left(u(x) - \int_{\mathbb{R}^N} u(x-y)\rho_h(y)\,\mathrm{d}y \right) w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \left(\int_{\mathbb{R}^N} (u(x) - u(x-y))\rho_h(y)\,\mathrm{d}y \right) w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \left(\int_{\mathbb{R}^N} \left(\int_0^1 \frac{\mathrm{d}}{\mathrm{d}s} u(x-sy))\,\mathrm{d}s \right) \rho_h(y)\,\mathrm{d}y \right) w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \left(\int_{\mathbb{R}^N} \left(\int_0^1 -y \cdot \nabla u(x-sy))\,\mathrm{d}s \right) \rho_h(y)\,\mathrm{d}y \right) w(x)\,\mathrm{d}x \\
&\qquad = \int_0^1 \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} -y \cdot \nabla u(x-sy)\rho_h(y) w(x)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}s \\
&\qquad \leq \int_0^1 \int_{\mathbb{R}^N} h\rho_h(y) \left(\int_{\mathbb{R}^N} |\nabla u(x-sy)|^2\,\mathrm{d}x \right)^2 \left(\int_{\mathbb{R}^N} (w(x))^2\,\mathrm{d}x \right)^2 \mathrm{d}y\,\mathrm{d}s \\
&\qquad = h\,||\nabla u||_{L^2(\mathbb{R}^d)}\,||w||_{L^2(\mathbb{R}^d)}\,.
\end{aligned}$$

Now, selecting $w = u - \mathcal{M}_h u$ gives the desired estimate for smooth functions u.

Next, we prove (7.2). Again, let $w \in L^2(\mathbb{R}^d)$ be an arbitrary function.

$$\begin{aligned}
&\int_{\mathbb{R}^N} \frac{\partial^2}{\partial x_i \partial x_j} \mathcal{M}_h u(x) w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \left[\int_{\mathbb{R}^N} \left(\frac{\partial}{\partial x_j} u(x-y) \frac{\partial}{\partial x_i} \rho_h(y) \right) \mathrm{d}y \right] w(x)\,\mathrm{d}x \\
&\qquad = \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} \left(\frac{\partial}{\partial x_j} u(x-y) \frac{\partial}{\partial x_i} \rho_h(y) w(x) \right) \mathrm{d}x\,\mathrm{d}y \\
&\qquad \leq \int_{\mathbb{R}^N} \left[\frac{\partial}{\partial x_i} \rho_h(y) \left(\int_{\mathbb{R}^N} \left(\frac{\partial}{\partial x_j} u(x-y) \right)^2 \mathrm{d}x \right)^{\frac{1}{2}} \left(\int_{\mathbb{R}^N} (w(x))^2\,\mathrm{d}x \right)^{\frac{1}{2}} \right] \mathrm{d}y \\
&\qquad = \frac{1}{h} \left|\left| \frac{\partial}{\partial x_j} u \right|\right|_{L^2(\mathbb{R}^d)} ||w||_{L^2(\mathbb{R}^d)} \left|\left| \frac{\partial}{\partial x_i} \rho \right|\right|_{L^1(\mathbb{R}^d)}.
\end{aligned}$$

Selecting $w = \frac{\partial^2}{\partial x_i \partial x_j} \mathcal{M}_h u$ and then summing over i and j yields the result for smooth function u.

Finally, notice that Ω is an extension domain so $u \in H^1(\Omega)$ can be extended to a compactly supported Sobolev function $u' \in H^1(\mathbb{R}^d)$. Then applying the density of smooth functions in Sobolev spaces completes the result. □

7.2 Interpolation Estimates in 2D

7.2.1 Classical Estimates

The classical estimates of finite element interpolation are duplicated here for completeness. They can be found in many references including [6, 15].

Let K be a simplex. Let h_K denote the length of the longest side of K and ρ be the radius of the largest inscribed sphere of K. Let $\mathcal{I}_k$ denote the standard Lagrange interpolation operator by degree k polynomials on simplex K. Using the Bramble-Hilbert lemma and scaling properties of Sobolev semi-norms, the classical local error interpolation estimate is derived.

Proposition 7.2. *Let $k \geq 2$ be an integer and m be in integer in $[0, k]$. Then there exists $C > 0$ independent of simplex K such that*

$$|u - I_k u|_{H^m(K)} \leq C \frac{h_K^{k+1}}{\rho_K^m} |u|_{H^{k+1}(K)}$$

holds for all $u \in H^{k+1}(K)$.

It is natural to require the triangulation to satisfy a uniform bound on $\frac{h_K}{\rho_K}$ in order to deduce an $O(h^{k+1-m})$ error estimate. This is equivalent to imposing a minimum angle condition on all triangles in the mesh.

However, the term $\frac{h_K}{\rho_K}$ is non-optimal for some triangles. This was observed by Jamet [22] and Babuška and Aziz [2].

7.2.2 Higher order elements, $k \geq 2$

Jamet's analysis yields a general interpolation estimate for by quadratic and higher order elements on arbitrary triangles. The only dependence of the shape of the triangle where the function is defined is through the largest angle of the triangle.

Theorem 7.3. *[22] Let $k \geq 2$ be an integer and m be in integer in $[0, k]$. Then there exists $C > 0$ independent of simplex K such that*

$$||u - I_k u||_{H^m(K)} \leq C \frac{h_K^{k+1-m}}{(\cos(\theta_K/2))^m} ||u||_{H^{k+1}(K)}$$

holds for $u \in H^{k+1}(K)$ where θ_K is the largest angle of K.

This estimate can be restated as

$$||u - I_k u||_{H^m(K)} \leq C h_K^{k+1-2m} R_K^m ||u||_{H^{k+1}(K)}$$

where R_K is the circumradius of K by observing that

$$\frac{h_K}{\cos(\theta_K/2)} \approx R_K. \tag{7.3}$$

This relationship gives a natural geometric interpretation with respect to Delaunay refinement algorithms and will be shown carefully in the next section.

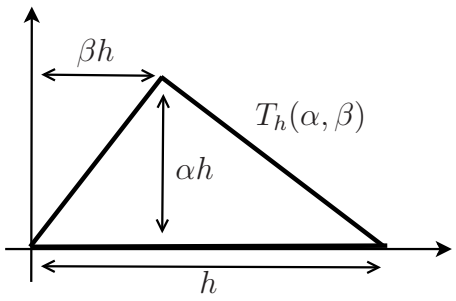


Figure 7.2: Parametrization of a general triangle.

7.2.3 Linear Elements, $k = 1$

The maximum angle condition for linear triangular element is considered by Babuška and Aziz [2].

Theorem 7.4. *[2, Theorem 2.1] Let $\theta_{max} < \pi$. There exists constant $C(\theta_{max}) > 0$ such that*

$$||u - I_1 u||_{H^1(K)} \leq C(\theta_{max}) h_K \, ||u||_{H^2(K)}$$

holds uniformly for all $u \in H^2(K)$ over the class of triangles K with largest angle θ_K such that $\theta_K < \theta_{max}$.

Jamet's work suggests that $C(\theta_{max})$ should have the form $C(\theta) \approx \frac{C}{\cos(\theta_{max}/2)}$, the Jamet's theorem does not apply in the case of linear elements. This result has been shown in [19] using a combinatorial approach. In what follows, we show that this fact is not an immediate corollary of the theorem of Babuška and Aziz, but by revisiting their arguments, the sharp result can be recovered. But first, we formalize the relationship between the largest angle and the circumradius of a general triangle which was mentioned in (7.3).

First, we give a parametrization for any triangle by letting the longest side lie on the x axis. For $0 < \alpha \leq \frac{\sqrt{3}}{2}$ and $0 \leq \beta \leq .5$, let $T_h(\alpha, \beta)$ be the triangle with vertices $(0, 0)$, $(h, 0)$ and $(\beta h, \alpha h)$, as depicted in Figure 7.2. Any triangle can be translated to one of the form $T_h(\alpha, \beta)$ by a sequence of rotation, tranlation and reflection about the y axis, and all of these operations are invariant with respect to the Sobolev semi-norms. Given $u \in H^k(T_h(\alpha, \beta))$, let $I_k u$ be the standard Lagrange interpolant of u by a polynomial of degree at most k.

Consider extending Theorem 7.4 to general triangles by using the linear map $\rho : T_h(\alpha, 0) \to T_h(\alpha, \beta)$ given by $\rho(\xi) = J\xi$ where J is given below. See Figure 7.3.

$$J = \begin{bmatrix} 1 & \frac{\beta}{\alpha} \\ 0 & 1 \end{bmatrix} \quad J^{-1} = \begin{bmatrix} 1 & -\frac{\beta}{\alpha} \\ 0 & 1 \end{bmatrix}$$

Using the theorem on the class of functions $T_h(\alpha, 0)$ and applying the standard techniques gives the estimate below.

$$\begin{aligned} ||u - I_1 u||_{H^1(T_h(\alpha,\beta))} &\leq C|J^{-1}||J|^2 h \, ||u||_{H^2(T_h(\alpha,\beta))} \\ &\leq \bar{C}(1 + \frac{\beta}{\alpha})^3 h \, ||u||_{H^2(T_h(\alpha,\beta))} \end{aligned}$$

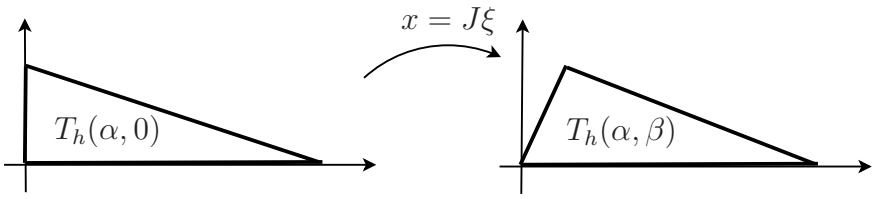


Figure 7.3: Shear operation taking a right triangle into a general triangle.

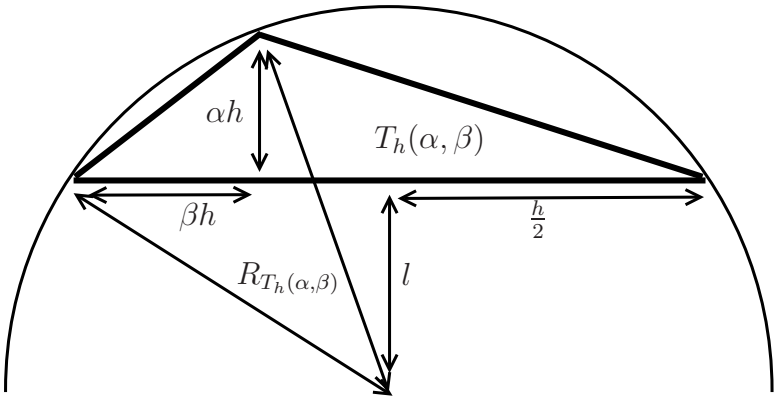


Figure 7.4: Diagram for Proposition 7.5. l is positive in this configuration.

As the maximum angle grows, this estimate is not sharp: it overestimates by two factors of $|J|$. The analysis below follows the arguments of Babuška and Aziz but allows for more general triangles from the beginning of the argument. This will yield a stronger relationship between the maximum angle and the error estimate (through the circumradius of the triangle). First, we estimate the circumradius of $T_h(\alpha, \beta)$. This proposition justifies the relationship (7.3).

Proposition 7.5. $R^2_{T_h(\alpha,\beta)} \geq \frac{\beta^2 h^2}{16\alpha^2} + \frac{h^2}{8}$.

Proof. Define l as the (signed) distance from the circumcenter of $T_h(\alpha, \beta)$ to the midpoint of the side of length h as shown in Figure 7.4. Using the Pythagorean theorem (twice) gives

$$\left(\frac{h}{2}\right)^2 + l^2 = R^2_{T_h(\alpha,\beta)} = \left(\frac{h}{2} - \beta h\right)^2 + (\alpha h + l)^2 . \tag{7.4}$$

Expanding yields

$$\frac{h^2}{4} + l^2 = \frac{h^2}{4} + \beta^2 h^2 - \beta h^2 + l^2 + 2\alpha l h + \alpha^2 h^2.$$

Solving for l gives

$$\begin{aligned} l &= (1 - \beta)\frac{\beta h}{2\alpha} - \frac{\alpha h}{2} \\ &\geq \frac{\beta h}{4\alpha} - \frac{\alpha h}{2}. \end{aligned} \tag{7.5}$$

Now, consider two cases.

Case 1: $\frac{\beta h}{4\alpha} - \frac{\alpha h}{2} < 0$.

This implies that $\beta < 2\alpha^2$. Then, using the fact that $\beta < 1$ gives the following.

$$\begin{aligned}\frac{\beta^2 h^2}{16\alpha^2} + \frac{h^2}{8} &\leq \frac{\beta h^2}{8} + \frac{h^2}{8} \\ &\leq \frac{h^2}{4} \\ &\leq R^2_{T_h(\alpha,\beta)}\end{aligned}$$

The final inequality is a result of (7.4).

Case 2: $\frac{\beta h}{4\alpha} - \frac{\alpha h}{2} \geq 0$.

In this case, substitute the bound for l in (7.5) into (7.4).

$$\begin{aligned}R^2_{T_h(\alpha,\beta)} &\geq \frac{h^2}{4} + \left(\frac{\beta h}{4\alpha} - \frac{\alpha h}{2}\right)^2 \\ &\geq \frac{h^2}{4} + \left(\frac{\beta h}{4\alpha}\right)^2 + \frac{\alpha^2 h^2}{4} - \frac{\beta h^2}{4} \\ &\geq \left(\frac{\beta h}{4\alpha}\right)^2 + \frac{h^2(1-\beta)}{4}\end{aligned}$$

Then, $1 - \beta > \frac{1}{2}$ implies $R^2_{T_h(\alpha,\beta)} \geq \frac{\beta^2 h^2}{16\alpha^2} + \frac{h^2}{8}$. □

Next, consider the following sets of functions.

$$\begin{aligned}\mathscr{T}_h(\alpha,\beta) &:= \left\{u \in H^2(T_h(\alpha,\beta)) \,|\, u(0,0) = 0, u(h,0) = 0, u(\beta h, \alpha h) = 0\right\} \\ \Xi_h(\alpha,\beta) &:= \left\{u \in H^1(T_h(\alpha,\beta)) \,|\, \int_0^h u(\beta s, \alpha s)\,\mathrm{d}s = 0\right\} \\ \Xi'_h(\alpha,\beta) &:= \left\{u \in H^1(T_h(\alpha,\beta)) \,|\, \int_0^h u(s,0)\,\mathrm{d}s = 0\right\}\end{aligned}$$

These sets are motivated as follows. For any $u \in H^2(T_h(\alpha,\beta))$, $(u - I_1 u) \in \mathscr{T}_h(\alpha,\beta)$, $\left(\beta\frac{\partial}{\partial x}(u - I_1 u) + \frac{\partial}{\partial y}(u - I_1 u)\right) \in \Xi_h(\alpha,\beta)$, and $\frac{\partial}{\partial x}(u - I_1 u) \in \Xi'_h(\alpha,\beta)$. When the subscript h equals 1 in any of these sets (and when denoting of $T_h(\alpha,\beta)$), it will be omitted. Also, denote the coordinates by $\mathbf{x} = (x, y)$. The proof begins with a uniform Poincaré inequality over the class of triangles with α and h set to 1 (i.e. only varying β).

Lemma 7.6. *Let*

$$A^2 = \inf_{\substack{u \in \Xi(1,\beta) \cup \Xi'(1,\beta) \\ \beta \in [0, \frac{1}{2}]}} \frac{\displaystyle\int_{T(1,\beta)} \left[u_x^2 + u_y^2\right]\,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} u^2\,\mathrm{d}\mathbf{x}}.$$

Then $A^2 > 0$.

Proof. Note that this inequality is shown for $T(1,0)$ in [2] using the Poincaré inequality. Since there is a bound on the aspect ratio of all triangles in $\{T(1,\beta) \mid \beta \in [0,\frac{1}{2}]\}$ and all these triangles longest edge length of similar length, the usual change of variables yields the lemma. □

The following lemma is the key to extending the proof of Babuška and Aziz to general triangles. The corresponding original lemma lacks the dependence on β.

Lemma 7.7. *Let*

$$B^2(\alpha,\beta) = \inf_{u\in\mathscr{T}(\alpha,\beta)} \frac{\displaystyle\int_{T(\alpha,\beta)} \left[u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2\right] \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(\alpha,\beta)} \left[u_x^2 + u_y^2\right] \,\mathrm{d}\mathbf{x}}.$$

Then $\left(\frac{\beta^2}{\alpha^2}+1\right) B^2(\alpha,\beta) \geq \frac{A^2}{6}$.

Proof. Let $U(x,y) = u(x,\alpha y)$ for $(x,y) \in T(1,\beta)$.

$$\left(\frac{\beta^2}{\alpha^2}+1\right) B^2(\alpha,\beta) = \left(\frac{\beta^2}{\alpha^2}+1\right) \inf_{u\in\mathscr{T}(1,\beta)} \frac{\displaystyle\int_{T(1,\beta)} \left[U_{xx}^2 + \tfrac{2}{\alpha^2}U_{xy}^2 + \tfrac{1}{\alpha^4}U_{yy}^2\right] \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} \left[U_x^2 + \tfrac{1}{\alpha^2}U_y^2\right] \,\mathrm{d}\mathbf{x}}$$

The middle term is split and then using the fact that $\alpha \leq 1$ gives

$$\left(\frac{\beta^2}{\alpha^2}+1\right) B^2(\alpha,\beta) \geq \left(\frac{\beta^2}{\alpha^2}+1\right) \inf_{u\in\mathscr{T}(1,\beta)} \frac{\displaystyle\int_{T(1,\beta)} \left\{U_{xx}^2 + U_{xy}^2 + \tfrac{1}{\alpha^2}\left[U_{xy}^2 + U_{yy}^2\right]\right\} \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} \left[U_x^2 + \tfrac{1}{\alpha^2}U_y^2\right] \,\mathrm{d}\mathbf{x}}$$

Rearranging and ignoring some unimportant terms yields

$$\begin{aligned}\left(\frac{\beta^2}{\alpha^2}+1\right) B^2(\alpha,\beta) \quad \geq \quad \inf_{u\in\mathscr{T}(1,\beta)} \Bigg\{ & \frac{\displaystyle\int_{T(1,\beta)} U_{xx}^2 + U_{xy}^2 \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} \left[U_x^2 + \tfrac{1}{\alpha^2}U_y^2\right] \,\mathrm{d}\mathbf{x}} + \frac{\beta^2}{3\alpha^2} \frac{\displaystyle\int_{T(1,\beta)} U_{xx}^2 + U_{xy}^2 \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} \left[U_x^2 + \tfrac{1}{\alpha^2}U_y^2\right] \,\mathrm{d}\mathbf{x}} \\ & + \frac{2}{3\alpha^2} \left[\frac{\displaystyle\int_{T(1,\beta)} \beta^2\left(U_{xx}^2 + U_{xy}^2\right) + U_{xy}^2 + U_{yy}^2 \,\mathrm{d}\mathbf{x}}{\displaystyle\int_{T(1,\beta)} U_x^2 + \tfrac{1}{\alpha^2}U_y^2 \,\mathrm{d}\mathbf{x}} \right] \Bigg\}\end{aligned}$$

Let $w = \beta U_x + U_y$. By observing $w \in \Xi(1,\beta)$, we can proceed to apply Lemma 7.6 on the numerator of the third term.

$$\begin{aligned}\int_{T(1,\beta)} \beta^2 U_{xx}^2 + U_{xy}^2 + \beta^2 U_{xy}^2 + U_{yy}^2 \,\mathrm{d}\mathbf{x} &\geq \frac{1}{2}\int_{T(1,\beta)} \left(w_x^2 + w_y^2\right) \,\mathrm{d}\mathbf{x} \\ &\geq \frac{A^2}{2}\int_{T(1,\beta)} w^2 \,\mathrm{d}\mathbf{x} \\ &= \frac{A^2}{2}\int_{T(1,\beta)} \left(\beta U_x + U_y\right)^2 \,\mathrm{d}\mathbf{x} \\ &\geq \frac{A^2}{2}\int_{T(1,\beta)} \frac{1}{2}U_y^2 - \beta^2 U_x^2 \,\mathrm{d}\mathbf{x}.\end{aligned}$$

Applying Lemma 7.6 to U_x (since $U_x \in \Xi'(1,\beta)$) gives

$$\int_{T(1,\beta)} U_{xx}^2 + U_{xy}^2 \,\mathrm{d}\mathbf{x} \geq A^2 \int_{T(1,\beta)} U_x^2 \,\mathrm{d}\mathbf{x}.$$

Combining yields

$$\begin{aligned}\left(\frac{\beta^2}{\alpha^2} + 1\right) B^2(\alpha,\beta) &\geq A^2 \inf_{u \in \mathscr{T}(1,\beta)} \frac{\int_{T(1,\beta)} U_x^2 + \frac{\beta^2}{3\alpha^2}U_x^2 + \frac{1}{3\alpha^2}\left(\frac{1}{2}U_y^2 - \beta^2 U_x^2\right) \,\mathrm{d}\mathbf{x}}{\int_{T(1,\beta)} U_x^2 + \frac{1}{\alpha^2}U_y^2 \,\mathrm{d}\mathbf{x}} \\ &\geq \frac{A^2}{6}.\end{aligned}$$

□

The next two lemmas involve very similar proofs to the previous two.

Lemma 7.8. *Let*

$$\bar{A}^2 = \inf_{\substack{u \in \mathscr{T}(1,\beta) \\ \beta \in [0,\frac{1}{2}]}} \frac{\int_{T(1,\beta)} \left[u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2\right] \,\mathrm{d}\mathbf{x}}{\int_{T(1,\beta)} u^2 \,\mathrm{d}\mathbf{x}}.$$

Then $\bar{A}^2 > 0$.

Lemma 7.9. *Let*

$$\bar{B}^2(\alpha,\beta) = \inf_{\substack{u \in \mathscr{T}(\alpha,\beta) \\ \beta \in [0,\frac{1}{2}]}} \frac{\int_{T(\alpha,\beta)} \left[u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2\right] \,\mathrm{d}\mathbf{x}}{\int_{T(\alpha,\beta)} u^2 \,\mathrm{d}\mathbf{x}}.$$

Then $\bar{B}(\alpha,\beta) > \bar{A}$.

Remark. Unlike Lemma 7.7, the inequality in Lemma 7.9 is independent of β. This is because the β dependence results from derivatives in the denominator which are not present in this case.

Now we can prove the desired theorem.

Theorem 7.10. *There exists C, independent of h, α and β such that*

$$|u - I_1 u|_{H^1_{T_h(\alpha,\beta)}} \leq C R_{T_h(\alpha,\beta)} |u|_{H^2_{T_h(\alpha,\beta)}}$$

hold for all $u \in H^2(T_h(\alpha,\beta))$ and any $\alpha, h \in (0,1]$ and $\beta \in [0, \frac{1}{2}]$.

Proof. Observe that

$$|u - I_1 u|_{H^2_{T_h(\alpha,\beta)}} = |u|_{H^2_{T_h(\alpha,\beta)}} \tag{7.6}$$

since $I_1 u$ is a piecewise linear function in H^1. Letting $\bar{u}(x,y) = u(hx, hy)$, Lemma 7.7 can be applied to the function $\bar{u} - I_1 \bar{u}$.

$$\begin{aligned}
|u - I_1 u|^2_{H^1_{T_h(\alpha,\beta)}} &= \frac{1}{h^2} |\bar{u} - I_1 \bar{u}|^2_{H^1_{T(\alpha,\beta)}} \\
&\leq \frac{6}{A^2 h^2} \left(\frac{\beta^2}{\alpha^2} + 1 \right) |\bar{u} - I_1 \bar{u}|^2_{H^2_{T(\alpha,\beta)}} \\
&\leq \frac{6h^2}{A^2} \left(\frac{\beta^2}{\alpha^2} + 1 \right) |u - I_1 u|^2_{H^2_{T_h(\alpha,\beta)}} \\
&= \frac{6}{A^2} \left(\frac{\beta^2}{\alpha^2} + 1 \right) h^2 |u|^2_{H^2_{T_h(\alpha,\beta)}} .
\end{aligned}$$

The inequality then follows by applying Proposition 7.5 □

A standard example shows that this bound scales in th optimal way with respect to β (and thus $R_{T_h(\alpha,\beta)}$). Consider the function $u(x,y) = x^2$ on the triangle $T_h(\alpha,\beta)$. The interpolant in this case is $I_1 u = hx + \frac{\beta}{\alpha} h(\beta - 1)y$. Then excluding higher order terms, we compute the following norms.

$$\begin{aligned}
|u|^2_{H^2_{T_h(\alpha,\beta)}} &\approx 2\alpha h^2 \\
|u - I_1 u|^2_{H^1_{T_h(\alpha,\beta)}} &\approx h^2 \left(1 + \frac{\beta^2}{\alpha^2} \right) \frac{\alpha h^2}{2}
\end{aligned}$$

It is clear that the ratio between these terms matches the bound in Theorem 7.10.

7.2.4 Interpolating Rough Functions

Interpolating functions in $H^1(\Omega)$ leads to some complications. The usual Lagrange interpolant is poorly defined since $H^1(\Omega)$ functions need not be continuous. Techniques of averaging can be used to avoid this problem and produce the optimal interpolants [16, 40]. These approaches are suitable for meshes satisfying minimum angle conditions but estimates involve the maximum

degree of the triangulation. This is a result of the fact that previous methods for average interpolation involve averaging in the reference configuration (or over a patch of elements in the reference configuration) and then ensuring some overlap conditions. For general triangulations, this overlap condition can fail. Figure 7.1(c) demonstrates a possible refinement in which the size of the triangulation is decreasing yet a uniform degree bound will fail.

We will demonstrate a general estimate which holds for any triangulation without any angle conditions. We consider a simplified situation: let $u \in H_0^1(\Omega)$ (i.e. u is 0 on the boundary of Ω) and let h be a uniform sizing parameter for the mesh (i.e. we will require all edges of the triangulation to be no longer than h). The interpolant constructed will give the expected error estimate and satisfy the zero boundary conditions.

Denote

$$\Omega_h := \{x \in \Omega \,|\, \text{dist}(x, \partial\Omega) > h\}$$

and let ψ_h be the following cutoff function.

$$\psi_h(x) = \begin{cases} 1 & \text{if } x \in \Omega_{2h}, \\ \frac{\text{dist}(x,\partial\Omega)-h}{h} & \text{if } x \in \Omega_h \setminus \Omega_{2h}, \\ 0 & \text{if } x \in \mathbb{R}^2 \setminus \Omega_h. \end{cases}$$

The distance to the boundary function is Lipschitz and ψ_h belongs to $H^1(\mathbb{R}^2)$.

Proposition 7.11. *There exists a constant $C > 0$ such that for all $u \in H_0^1(\Omega)$,*

$$|\psi_h u|_{H^1(\Omega)} \leq C\,|u|_{H^1(\Omega)}\,.$$

Proof. Proof follows by applying the product rule and the Poincare inequality.

$$\begin{aligned} ||\nabla(\psi_h u)||_{L^2(\Omega)} &\leq ||\nabla(\psi_h)u||_{L^2(\Omega_h\setminus\Omega_{2h})} + ||\psi_h \nabla u||_{L^2(\Omega)} \\ &\leq ||\nabla\psi_h||_{L^\infty(\Omega_h\setminus\Omega_{2h})}\,||u||_{L^2(\Omega_h\setminus\Omega_{2h})} + ||\nabla u||_{L^2(\Omega)} \\ &\leq \frac{1}{h}\,||u||_{L^2(\Omega\setminus\Omega_{2h})} + ||\nabla u||_{L^2(\Omega)} \\ &\leq \frac{1}{h} Ch\,|u|_{H^1(\Omega)} + |u|_{H^1(\Omega)}\,. \end{aligned}$$

□

Since the Lagrange interpolation operator is poorly defined, we instead consider the interpolation operator $\hat{I}_{h,\mathcal{T}} : L^2(\Omega) \to H^1(\Omega)$ defined by

$$\hat{I}_{h,\mathcal{T}} u = I_1\left(M_h\left(\psi_h u\right)\right)$$

where I_1 is the linear Lagrange interpolant, M_h is the mollifier discussed in Section 7.1, and ψ_h is the cuttoff function above. The cutoff function is used to ensure that the interpolant has the same boundary values as the the function u. Then that the interpolant is in the typical finite element space.

Theorem 7.12. *There exists C depending only upon Ω such that for all $u \in H_0^1(\Omega)$ and all triangulations $\mathcal{T}$ of Ω with no edges longer than h,*

$$\left|\left|u - \hat{I}_{h,\mathcal{T}} u\right|\right|_{L^2(\Omega)} \leq Ch \left|u\right|_{H^1(\Omega)}.$$

Remark. This is consistent with the classical interpolation theory in Proposition 7.2 which gives

$$\left|\left|u - \hat{I}_{h,\mathcal{T}} u\right|\right|_{L^2(\Omega)} \leq Ch^2 \left|u\right|_{H^2(\Omega)}$$

for functions u in $H^2(\Omega)$ *without restriction* on the triangulation.

Proof. First, the estimate is divided into three terms:

$$\begin{aligned}\left|\left|u - \hat{I}_{h,\mathcal{T}} u\right|\right|_{L^2(\Omega)} \leq & \left|\left|u - \psi_h u\right|\right|_{L^2(\Omega)} + \left|\left|M_h(\psi_h u) - \psi_h u\right|\right|_{L^2(\Omega)} \\ & + \left|\left|I_1(M_h(\psi_h u)) - M_h(\psi_h u)\right|\right|_{L^2(\Omega)}.\end{aligned}$$

The first term can be estimated using the Poincare inequality where $|\partial\Omega|$ denotes the perimeter of Ω. Throughout, this argument, C denotes the running constant and is not labeled after each estimate.

$$\begin{aligned}\left|\left|u - \psi_h u\right|\right|_{L^2(\Omega)} &= \left|\left|(1 - \psi_h)u\right|\right|_{L^2(\Omega \setminus \Omega_{2h})} \\ &\leq \left|\left|u\right|\right|_{L^2(\Omega \setminus \Omega_{2h})} \\ &\leq C|\partial\Omega_{2h}| h \left|u\right|_{H^1(\Omega \setminus \Omega_{2h})} \\ &\leq C|\partial\Omega| h \left|u\right|_{H^1(\Omega)}.\end{aligned}$$

The second term is estimated using (7.1) and Proposition 7.11:

$$\begin{aligned}\left|\left|M_h(\psi_h u) - \psi_h u\right|\right|_{L^2(\Omega)} &\leq Ch \left|\psi_h u\right|_{H^1(\Omega)} \\ &\leq Ch \left|u\right|_{H^1(\Omega)}.\end{aligned}$$

Finally, analysis of the third term involves Proposition 7.2, (7.2), and Proposition 7.11.

$$\begin{aligned}\left|\left|I_1(M_h(\psi_h u)) - M_h(\psi_h u)\right|\right|_{L^2(\Omega)} &\leq Ch^2 \left|M_h(\psi_h u)\right|_{H^2(\Omega)} \\ &\leq Ch \left|\psi_h u\right|_{H^1(\Omega)} \\ &\leq Ch \left|u\right|_{H^1(\Omega)}.\end{aligned}$$

Combining the three terms yields the result. □

7.3 2D Delaunay Refinement for the FEM

As noted earlier, the 2D Delaunay refinement algorithm described by Miller, Pav and Walkington [30] and Algorithm 5.3 (the intestine protection scheme) produce meshes with no large angles and thus by noting the theorem of Jamet and that of Babuška and Aziz, these meshing algorithms are suitable for the finite element method. In light of Theorem 7.10, it follows that triangulations produced by Shewchuk's terminator algorithm [41] and Algorithm 5.2 (the collar protection scheme) also produce suitable meshes for the finite element method: although large angles may exist, the circumradius of any remaining triangle is proportional to the local feature size.

However, the interpolation estimate in Theorem 7.10 suggests a different Delaunay refinement algorithm which focuses on producing a mesh with no large circumradii rather than ensure triangle quality via angle bounds. Given a user specified sizing function, $h : \Omega \to (0, \infty)$ which is bounded below by some positive constant, Algorithm 7.1 will produce a triangulation such that the circumradius of each triangle is bounded by the minimum value of h over the triangle. The following algorithm will terminate as long as the input satisfies one condition: all acutely adjacent input segments must have equal length (modulo powers of 2).

Algorithm 7.1 Simple Delaunay Refinement for FEM

Action	Insert the circumcenter of a simplex unless the simplex is a triangle and the circumcenter encroaches upon a segment. In this case, queue the encroached segment for splitting.
Priority	Segments are processed before triangles.
Unacceptability	A segment s is unacceptable if its diametral disk is non-empty. A triangle t is unacceptable if $\inf_{x \in t} h(x) < R_t$.
Safety	It is safe to split any simplex.

This algorithm can be thought of as a graded version of Chew's first Delaunay refinement algorithm [13] (given in Algorithm 7.2), which introduced the idea of circumcenter insertion and produce uniform meshes. Chew observed that in the case of a constant sizing function and appropriate restrictions on input segment lengths, refinement of triangles with large circumradii leads to triangulation with minimum angle at least $\frac{\pi}{6}$.

Algorithm 7.2 Chew's first Delaunay refinement algorithm [13]

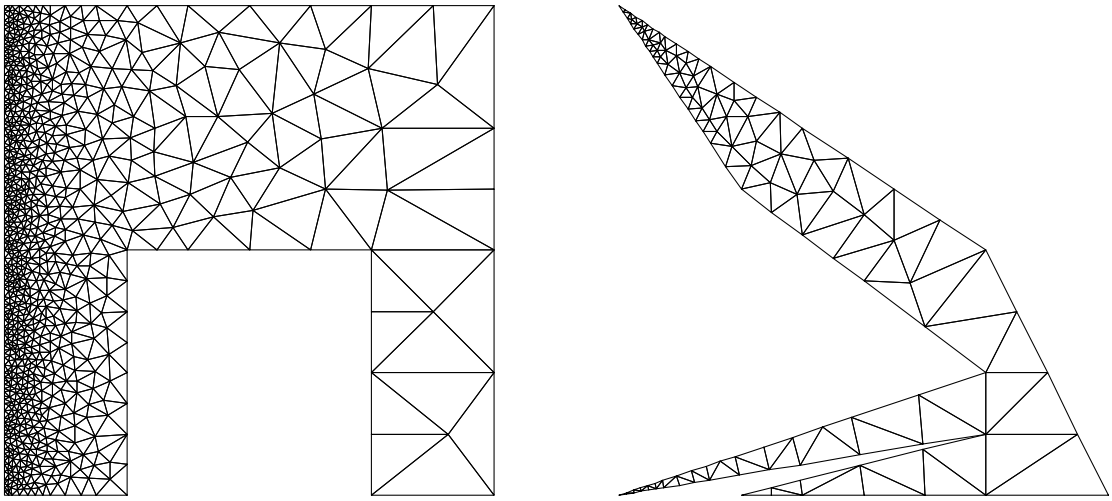
Action	Insert the circumcenter of a triangle.
Priority	Triangles are prioritized by circumradius, larger circumradii are processed first.
Unacceptability	A triangle is unacceptable if it has a circumradius larger than h.
Safety	It is safe to split any simplex.

Theorem 7.13. *Given a sizing function $h : \Omega \to \mathbb{R}$ which is bounded away from 0 and an input PLC in which adjacent segments are split at equal lengths, Algorithm 7.1 terminates and for each triangle t in the resulting triangulation,*

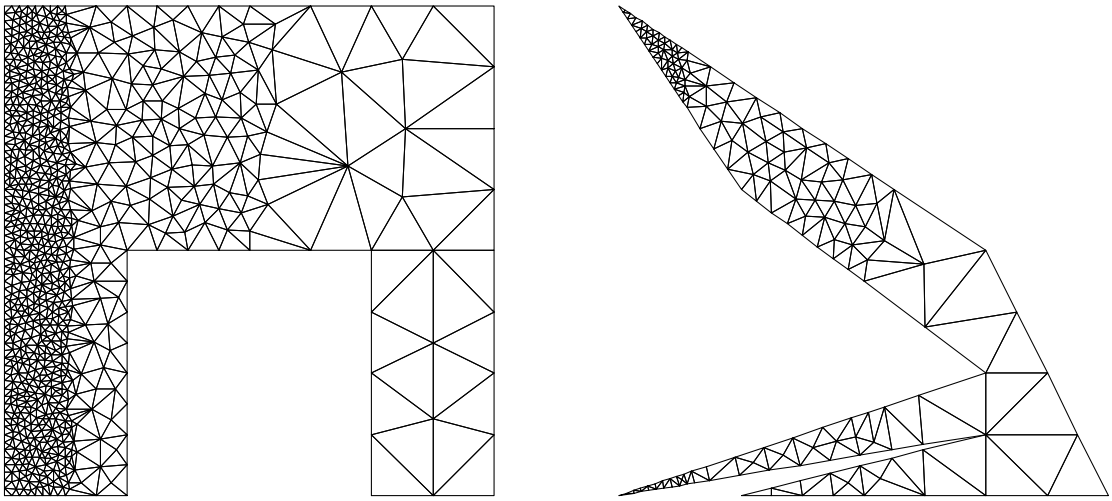
$$\frac{1}{2}\min\left(h(t),\, \text{lfs}(t)\right) \leq R_t \leq \inf_{x \in t} h(x)$$

where $h(t) := \inf_{x \in B(q,(1+\sqrt{2})r_q)} h(x)$ and q is the last vertex of t inserted into the mesh.

Two triangulations produced by this algorithm is shown in Figure 7.5. Despite the lack of any guarantee on the minimum angle in the resulting triangulation, the vast majority of the triangles do not contain small angles. This is consistent with the result of Chew's algorithm which shows that in the case of a constant sizing function, a minimum angle bound can be achieved. Triangles with large angles typically only occur near discontinuities in the sizing function.



(a) Two meshes generated using Algorithm 7.1 and sizing function which is quadratic in the horizontal direction.



(b) Two meshes generated using Algorithm 7.1 and a discontinuous sizing function.

Figure 7.5: Algorithm 7.1 examples

Bibliography

[1] Umut A. Acar, Benoît Hudson, Gary L. Miller, and Todd Phillips. SVR: practical engineering for a fast 3D meshing algorithm. In *Proceedings of the 16th International Meshing Roundtable*, pages 45–62, 2007.

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

[3] Marshall Bern and John Gilbert. Provably good mesh generation. *Journal of Computer and System Sciences*, 48:231–241, 1994.

[4] Charles Boivin and Carl Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185–1213, 2002.

[5] Adrian Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[6] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.

[7] David E. Cardoze, Gary L. Miller, Mark Olah, and Todd Phillips. A bezier-based moving mesh framework for simulation with elastic membranes. In *Proceedings of the 13th International Meshing Roundtable*, pages 71–80, 2004.

[8] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver exudation. *Journal of the ACM*, 47(5):883–904, 2000.

[9] Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *Proceedings of the 16th International Meshing Roundtable*, pages 477–494, 2007.

[10] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional Delaunay mesh generation. In *SODA '03: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 295–304, 2003.

[11] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional Delaunay mesh generation. *Discrete and Computational Geometry*, 36(3):419–456, 2006.

[12] Andrey Chernikov and Nikos Chrisochoides. Three-dimensional semi-generalized point placement method for delaunay mesh refinement. In *16th International Meshing Roundtable*, pages 25–44, October 2007.

[13] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Computer Science Department, Cornell University, 1989.

[14] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *SCG '93: Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 274–280, 1993.

[15] Philippe G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

[16] Ph. Clément. Approximation by finite element functions using local regularization. *RAIRO: Analyse Numérique*, 9(R-2):77–84, 1975.

[17] David Cohen-Steiner, Éric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3D. *Computational Geometry: Theory and Applications*, 28(2-3):217–233, 2004.

[18] Herbert Edelsbrunner and Damrong Guoy. An experimental study of sliver exudation. *Engineering With Computers*, 18(3):229–240, 2002.

[19] Stephen Guattery, Gary L Miller, and Noel Walkington. Estimating interpolation error: a combinatorial approach. In *SODA '99: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 406–413, 1999.

[20] Benoît Hudson. Safe Steiner points for delaunay refinement. Research Notes of the 17th International Meshing Roundtable, 2008.

[21] Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, 2006.

[22] Pierre Jamet. Estimations d'erreur pour des elements finis droits presque degeneres. *RAIRO: Analyse Numérique*, 10(3):43–61, 1976.

[23] Ravi Jampani and Alper Üngör. Construction of sparse well-spaced point sets for quality tetrahedralizations. In *Proceedings of the 16th International Meshing Roundtable*, pages 63–80, 2007.

[24] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, 2007.

[25] Patrick M. Knupp. Algebraic mesh quality metrics. *SIAM Journal on Scientific Computing*, 23(1):193–218, 2001.

[26] Xiang-Yang Li and Shang-Hua Teng. Generating well-shaped Delaunay meshes in 3D. In *SODA '01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, 2001.

[27] Gary L. Miller. A time efficient delaunay refinement algorithm. In *SODA '04: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 400–409, 2004.

[28] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. Fully incremental 3D Delaunay

refinement mesh generation. In *Proceedings of the 11th International Meshing Roundtable*, pages 75–86, 2002.

[29] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. An incremental Delaunay meshing algorithm. Technical Report 02-CNA-023, Center for Nonlinear Analysis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2002.

[30] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. When and why Ruppert's algorithm works. In *Proceedings of the 12th International Meshing Roundtable*, pages 91–102, 2003.

[31] Gary L. Miller, Todd Phillips, and Donald Sheehy. Size competitive meshing without large angles. In *34th International Colloquium on Automata, Languages and Programming*, 2007.

[32] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. On the radius–edge condition in the control volume method. *SIAM Journal on Numerical Analysis*, 36(6):1690–1708, 1999.

[33] Scott A. Mitchell and Stephen A. Vavasis. Quality mesh generation in three dimensions. In *Symposium on Computational Geometry*, pages 212–221, 1992.

[34] Michael Murphy, David M. Mount, and Carl W. Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *International Journal of Computational Geometry and Applications*, 11(6):669–682, 2001.

[35] Steven E. Pav. *Delaunay Refinement Algorithms*. PhD thesis, Carnegie Mellon University, May 2003.

[36] Steven E. Pav and Noel J. Walkington. Robust three dimensional Delaunay refinement. In *Proceedings of the 13th International Meshing Roundtable*, pages 145–156, 2004.

[37] Steven E. Pav and Noel J. Walkington. Delaunay refinement by corner lopping. In *Proceedings of the 14th International Meshing Roundtable*, pages 165–181, 2005.

[38] Alexander Rand. Reordering Ruppert's algorithm. In *18th Fall Workshop on Computational Geometry*, 2008.

[39] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

[40] L. Ridgway Scott and Shangyou Zhang. Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Mathematics of Computation*, 54(190):483–493, 1990.

[41] Jonathan Richard Shewchuk. Mesh generation for domains with small angles. In *SCG '00: Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pages 1–10, 2000.

[42] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):86–95, 2002.

[43] Hang Si. On refinement of constrained Delaunay tetrahedralizations. In *Proceedings of the 15th International Meshing Roundtable*, pages 510–528, 2006.

[44] Hang Si and Klaus Gartner. Meshing piecewise linear complexes by constrained Delaunay

tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, 2005.

[45] Alper Üngör. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed delaunay triangulations. In *Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN'04)*, pages 152–161, Buenos Aires, Argentina, 2004.

[46] David F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24:167–171, 1981.