Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

Adaptive Tetrahedral Mesh Generation by Constrained Delaunay Refinement

Hang Si
Weierstrass Institute for Applied Analysis and Stochastics
`si@wias-berlin.de`

submitted: August 22, 2006

No. 1176
Berlin 2006

2000 *Mathematics Subject Classification.* 65M50, 65N50.

Key words and phrases. tetrahedral mesh generation, adaptive mesh generation, constrained Delaunay, mesh refinement, mesh quality.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Adaptive Tetrahedral Mesh Generation by Constrained Delaunay Refinement

Hang Si*

Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstrasse 39, 10117, Berlin, Germany

SUMMARY

This paper discusses the problem of refining a constrained Delaunay tetrahedralization (CDT) for adaptive numerical simulation. A simple and efficient algorithm which makes use of the classical Delaunay refinement scheme is proposed. It generates an isotropic tetrahedral mesh corresponding to a sizing function which can be either user-specified or automatically derived from the input CDT. The quality of the produced meshes is guaranteed, i.e., most output tetrahedra have their circumradius-to-shortest-edge ratios bounded except those in the neighborhood of small input angles. Good mesh conformity can be obtained for smoothly changing sizing information. The algorithm has been implemented. Various examples are provided to illustrate its theoretical aspects as well as practical performance.

KEY WORDS: tetrahedral mesh generation, adaptive mesh refinement, constrained Delaunay property, mesh quality

1. INTRODUCTION

Let Ω be a closed and bounded domain in $\mathbb{R}^3$ and let $\partial\Omega$ be a piecewise linear discretization of its boundary, i.e., $\partial\Omega$ is a set of vertices in $\mathbb{R}^3$ together with a set of non-crossing segments and facets. The *constrained tetrahedralization* $\mathcal{T}$ of $\partial\Omega$ is a tetrahedralization of its vertices which respects the segments and facets as well. In particular, $\mathcal{T}$ is a *constrained Delaunay tetrahedralization* (CDT) [18] if the Delaunay criterion [2] is satisfied everywhere in $\mathcal{T}$ except in the neighborhood of the facets. CDTs retain many nice properties of Delaunay tetrahedralizations which make them useful in many contexts [18]. It is well-known that a constrained tetrahedralization may not exist for an arbitrary polyhedron [1, 26] and the problem of deciding whether or not a polyhedron has a constrained tetrahedralization is NP-hard [8]. However, the existence of a CDT is guaranteed [14] if $\partial\Omega$ is slightly refined with few additional points. Various algorithms for efficiently constructing CDTs from an

*Correspondence to: Hang Si, Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstrasse 39, 10117, Berlin, Germany
†E-mail: si@wias-berlin.de

arbitrary domain have been proposed [17, 33]. A robust software implementation is publicly available [37].

CDTs are generally not well-suited in numerical simulation of physical models based on partial differential equations. Numerical methods like finite element and finite volume methods have special demands on the meshes [4, 16]. The element shape must satisfy certain quality measures, e.g. with respect to the aspect ratio (the longest edge divided by the inradius of an element). In order to capture the details of the solution field as well as to reduce the CPU time, the size of the mesh should be dense enough in places where the solution or its gradient changes rapidly, and as sparse as possible in the rest of the region. Usually, a CDT may contain many badly-shaped tetrahedra, and the number of nodes (the degrees of freedom) may be insufficiently small. Hence both the quality and mesh size of a CDT may need improving in order to meet the basic requirements.

Adaptive numerical methods which combine mesh generation, numerical approximation, linear system solving, and error estimation are effective to solve complex problems. In an adaptive simulation in which an approximated solution is computed repeatedly, the desired mesh size in a single loop is usually obtained from the solution of the previous loop through an error estimator. It is convenient to introduce a *sizing function* (or control space [7, 23]) to specify the desired size feature of the problem. For example, the function specifies the isotropic or anisotropic mesh size at any point in the domain.

The three-dimensional *adaptive mesh refinement* problem can be described as follows: given a mesh domain Ω in $\mathbb{R}^3$, the boundary constrained tetrahedralization $\mathcal{T}$ of $\partial\Omega$, and a sizing function H defined on Ω, find a set of additional points that refines $\mathcal{T}$, such that all tetrahedra of the refined mesh have a quality measure within certain bounds, and the mesh size conforms to H. In this paper, we study a special case of the general problem by assuming that $\mathcal{T}$ is a CDT. We refer to our problem as *CDT refinement*. This problem is motivated by two issues: (1) to extend previous work on CDTs [17, 33, 37]; and (2) to have a robust and efficient method to refine any CDT into a state acceptable for adaptive numerical simulations (an example is shown in Fig. 1). Like most other refinement methods, CDT refinement does not guarantee the removal of all badly-shaped elements. Thus a combination with mesh smoothing or optimization step is necessary.

1.1. Previous Work

The Delaunay refinement methods pioneered by Chew [5, 6], Ruppert [9], and Shewchuk [13] are well known for having theoretical guarantees on the quality of mesh elements. Moreover, the dual graph of the resulting mesh is a Voronoï diagram. This property is extremely useful for finite volume meshes [12, 24]. In three dimensions, only one class of badly-shaped tetrahedra, so called *slivers* (a sliver has no short edges but nearly zero volume), can survive. The mesh optimization is essentially used to remove slivers [19, 21, 22]. The main limitation against the elegance of the basic scheme is that no input angle should be smaller than 90°. This condition is not likely to be satisfied in most of the realistic problems. Much work [15, 27, 20] has gone into removing the restrictions. However, all of these methods are not adaptive, i.e., they do not allow to account for an arbitrary sizing function.

The algorithm of Miller et al [12] finds a well-spaced point set conforming to the domain boundary by sphere-packing, then triangulates the point set using the Delaunay criterion. Recently, Oudot et al [32] designed a volume meshing algorithm which greedily samples the

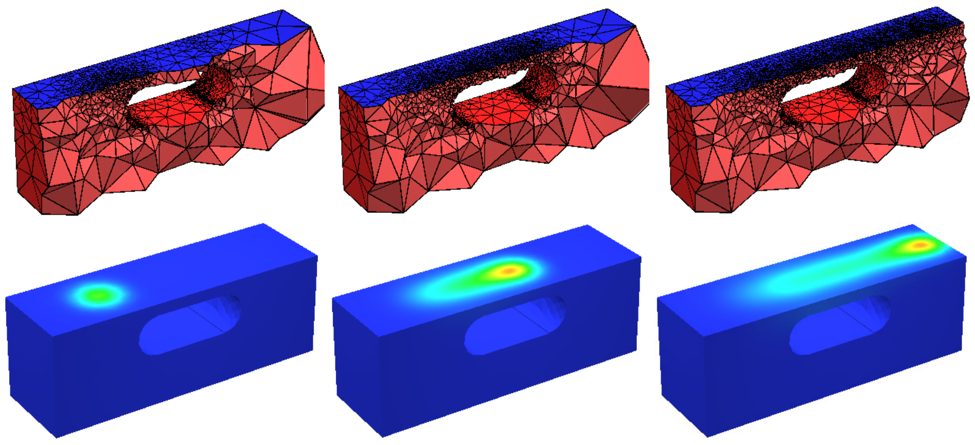


Figure 1. Adaptive mesh refinement in a transient heat conduction simulation (an example of `WIAS-SHarP` [35]). **Top**: three adaptive meshes at different time steps. Each mesh was refined based on the error estimated from the solution of the previous time step. **Bottom**: the corresponding finite volume solutions.

interior and the boundary of the domain using a similar Delaunay refinement scheme. Both algorithms support user-defined sizing functions. However, these methods are not designed for refining CDTs. The initial triangulation does not contain boundary data, and thus the boundaries have to be enforced by the refinement.

Another class of refinement methods [10, 11], popularly used in engineering, works in two phases: (1) point generation using the given sizing information, and (2) point insertion according to the Delaunay criterion. In part (2), some generated points are not inserted due to the closeness to the existing points. This approach is able to quickly generate a number of additional points and can be parallelized easily. From the theoretical point of view, this approach does not guarantee mesh quality. It heuristically relies on mesh optimization.

There are tetrahedral mesh generation methods directly based on mesh smoothing and optimization [25, 30, 31]. Assume a set of additional nodes has been generated inside the mesh domain, the basic idea of these methods is to introduce a mesh-dependent energy function, and then minimize the function by changing both the node positions and mesh connectivity. The generated mesh contains well-shaped tetrahedra. However, the CPU time of these methods depends dramatically on the number and the initial locations of the nodes. They are suitable as a post step of mesh refinement to improve the final mesh quality.

1.2. Our Contribution

In this paper, a practical mesh refinement algorithm which builds on several previous work [9, 10, 13, 32] is presented. This algorithm generates an isotropic mesh corresponding to a sizing function H which can be either user-specified or automatically derived. In a nutshell, the CDT is refined incrementally by appropriately inserting points into it. At each step, a new point v is generated by the basic Delaunay refinement rules, v is inserted only if the local mesh is sparse according to H. The process terminates when no new point can be inserted. Our

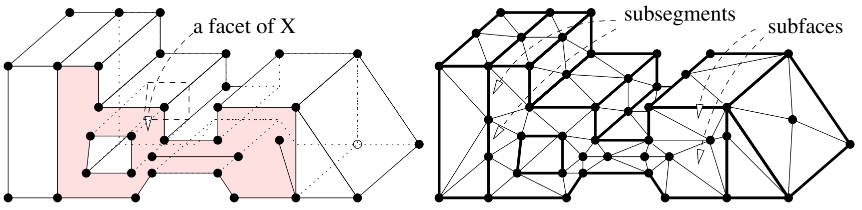


Figure 2. **Left**: a piecewise linear complex (PLC). The shaded area highlights one of its facets. **Right**: a constrained Delaunay tetrahedralization (CDT) of the left PLC. The surface mesh of the CDT consists of subsegments and subfaces.

algorithm makes two improvements over the basic Delaunay refinement algorithms [9, 13]: (1) it has no restriction on the input angle but certain mesh quality can still be guaranteed; and (2) it takes a user-defined sizing function into account, which enables the mesh adaption under specific features.

Our analysis establishes several advantages of the algorithm. Good mesh quality is guaranteed in the bulk of the domain. The remaining low-quality tetrahedra are well-located in the neighborhood of small input angles. By assuming some smoothness of the sizing function, good mesh conformity can be obtained as well. The constrained Delaunay property of the generated meshes can be enforced by applying the segment splitting rules introduced in [33].

The remainder of this paper is organized as follows. In Section 2, after a brief review of some basic definitions in CDTs, the proposed algorithm is described in detail. In Section 3, we provide theoretical analysis regarding the conditions on mesh quality and mesh conformity of our algorithm. Section 4 discusses ways of specifying sizing functions, which include automatically deriving a sizing function, and the use of a background mesh. In Section 5, we demonstrate various examples corresponding to our analysis. We conclude with some remarks and discuss some possible future improvements.

2. CONSTRAINED DELAUNAY REFINEMENT

In this section, the algorithm for refining CDTs is presented. It behaves like the Delaunay refinement algorithm of Shewchuk [13], i.e. it finds the badly-shaped tetrahedra and eliminates them by inserting their circumcenters. However, the insertion of circumcenters is restricted by the local mesh sizing information specified on input. We refer to this algorithm as *constrained Delaunay refinement.*

2.1. Definitions

The boundary of the mesh domain is represented by a *piecewise linear complex* X [12], i.e., X is a set of vertices in $\mathbb{R}^3$, together with a set of segments and *facets* (see Fig. 2 left). Our algorithm takes a CDT $\mathcal{T}$ of X as input. The segments and facets of X are represented as a union of *subsegments* and *subfaces* in $\mathcal{T}$ (see Fig. 2 right). Any tetrahedron τ in $\mathcal{T}$ is *constrained Delaunay*, i.e., the circumsphere of τ encloses no vertex of X that is visible from the inside of τ, the *visibility* is blocked by the subsegments and subfaces of X [18].

Let $H : X \to \mathbb{R}^+$ be a *sizing function* such that for each point $p \in X$, $H(p)$ specifies the desired length of edges connecting at a vertex inserted at the location of p. H is *isotropic* if the edge length does not vary with respect to the directions at p, otherwise, it is *anisotropic*. In the scope of this paper, we assume H is isotropic. An ideal sizing function is defined analytically at any point of X. In practice, it is more convenient to approximate H by a discrete function specified at some points in X, the size of other points is obtained by means of interpolation.

Let t be a tetrahedron, r the radius of the circumsphere of t, l the shortest edge length of t. The *radius-edge ratio* of t is defined by r/l. A regular tetrahedron (whose all edges have equal length) minimizes the ratio, which is $\sqrt{6}/4 \approx 0.612$. Badly shaped tetrahedra (e.g., needles, wedges, etc) usually have a large radius-edge ratio, except slivers which may have ratios as small as $\sqrt{2}/2 \approx 0.707$. In this sense, the radius-edge ratio is not a proper quality measure for tetrahedra. Nevertheless, for a theoretical point of view, it can be useful. In practice, the radius-edge ratio needs to be replaced by better quality measures such as aspect ratio, the largest dihedral angle, etc.

A subsegment or subface can have infinitely many circumspheres in $\mathbb{R}^3$. However, its smallest circumsphere (i.e. the diametric circumsphere) is unique. In the scope of this paper, we tacitly use the term circumsphere to mean the unique one. A subsegment or a subface is said to be *encroached* if a vertex lies inside or on its circumsphere.

2.2. The Algorithm

Given a CDT $\mathcal{T}$ to be refined, a sizing function H, a radius-edge ratio bound B, and two parameters α_1, α_2, the algorithm incrementally adds points into $\mathcal{T}$ and updates $\mathcal{T}$ into a refined mesh.

At each step, a new point v is generated by the basic Delaunay refinement scheme, i.e. v is found by the following three *point-generating rules*.

- $R1$ If a subsegment s is encroached, then v is the midpoint of s.
- $R2$ If a subface f is encroached, then v is the circumcenter of f. However, if the choice of v encroaches upon some subsegments, then reject v. Instead, use $R1$ to find a v on one of the encroached subsegments.
- $R3$ If a tetrahedron t satisfies one of the following two cases, $R3.1$ or $R3.1$, where:

  - $R3.1$ t has a radius-edge ratio greater than B.
  - $R3.2$ there is a corner p of t, such that $\alpha_1 H(p) < r$, where r is the radius of the circumsphere of t,

  then v is the circumcenter of t. However, if this choice of v encroaches upon any subsegment or subface, then reject v. Instead, use $R1$ or $R2$ to find a v on one of the encroached subsegments or subfaces.

Once the point v is found, the *point-accepting rule* decides whether or not v can be inserted into the mesh. Let P be a set of vertices collected as follows:

- If v is found by $R1$, then P contains two endpoints of s.
- If v is found by $R2$, then P contains the endpoints of subfaces which v is intended to split.
- If v is found by $R3$, then P contains the endpoints of tetrahedra which v is intended to split.

Then v is inserted if $\alpha_2 H(p) < |v - p|$ for all $p \in P$, where $| \cdot |$ is the Euclidean distance. Otherwise, v is not inserted.

If v passes the point-accepting rule, then it is inserted into the current mesh, and the local mesh of v is rearranged according to the Delaunay criterion.

Remark. $R3.1$ tests if t has bad quality, and $R3.2$ checks the H-conformity of the corners of t. $R3.1$ has priority higher than $R3.2$, that is, $R3.2$ is triggered only if all tetrahedra have radius-edge ratio larger than B.

In the point-accepting rule, if v is found by $R1$ or $R2$, only the endpoints of the subsegment or subfaces of the same facet on which v lies have the right to accept or reject v. It appears that v can be very close to some existing vertices in terms of H, i.e., there exist a point $p \notin P$, such that $\alpha_2 H(p) > |v - p|$. However, we will show in the next section that the distance $|v - p|$ is always bounded by a constant times the radius of a protecting ball.

2.3. Repair Non-Delaunay Segments

Generally, the mesh generated by the above algorithm may not be a CDT. A non-Delaunay subsegment may not get split due to the point-accepting rule. Such subsegments may result some internal faces which are both locally non-Delaunay and unflippable.

We can force the splitting all encroached subsegments by ignoring the point-accepting rule if the point is generated by $R1$, then all non-Delaunay subsegments will be split. However, this simple approach will not terminate if there are acute input angles. For example, if two segments form an angle smaller than $90°$, then inserting a point on one segment by $R1$ may cause another segment to become encroached. The situation may be repeated forever. We need a rule to replace $R1$ which can guarantee both the removal all non-Delaunay subsegments and the termination.

In [33], a simple segment recovery algorithm, which uses a set of segment splitting rules, is introduced for constructing CDTs. The algorithm takes a piecewise linear complex X, incrementally splits non-Delaunay segments of X until all subsegments of X are Delaunay. The termination of the algorithm is guaranteed. In the following, we propose a new rule $R1^*$ for repairing encroached segments which uses these segment splitting rules.

$R1^*$ If a segment s is encroached by at least one existing vertex, then v is found by using one of the segment splitting rules of [33]; otherwise (s is encroached by a rejected circumcenter of an encroached subface or a bad quality tet), v is the midpoint of s.

It will be shown in the next section that if $R1^*$ is used instead of $R1$ in the above algorithm and the point-accepting rule accepts the points generated by $R1^*$ whenever they are used to repair non-Delaunay subsegments, then the subsegments in the output mesh are Delaunay and the mesh is again a CDT.

3. ANALYSIS

The central idea of the algorithm is the following: Only inserts a point when the local mesh of the point is sparse. The sparseness is indicated by the values of the sizing function at its adjacent vertices. In the isotropic case, one can assume that each vertex p of the mesh is surrounded by two virtual balls, one *sparse ball* with radius $\alpha_1 H(p)$, and one *protecting ball* with radius $\alpha_2 H(p)$. The space outside the sparse ball of p is *sparse* from the viewpoint of p, while the space inside the protecting ball of p is free of additional points. Notice that if $\alpha_1 \to +\infty$ (i.e. no sparse space) and $\alpha_2 \to 0$ (i.e. no protecting ball), it becomes the basic Delaunay refinement algorithm [13].

In the following, we provide conditions on the sizing function H, and on the parameters α_1, α_2, and B to ensure the theoretical guarantees of our algorithm. Specifically, we will show:

- The termination of the algorithm only depends on α_2.
- The mesh quality is governed by both B and α_2.
- The properties of H will influence the mesh conformity.
- The mesh size depends on both α_1, α_2, and H.

3.1. Termination

For each output vertex v, its *parent* p_v is defined as follows: if v is an input vertex, p_v is the closest output vertex to v ($v \neq p_v$); if v is an inserted vertex, and if $P(v)$ denotes the set of vertices collected by the point-accepting rule, then $p_v \in P(v)$ is the closest vertex to v immediately after v is inserted. If there are several such vertices, then choose the one which is the most recently inserted. Notice that p_v may not be the closest output vertex to v.

Given a point p in a PLC X, the *local feature size* [9] $lfs(p)$ is the radius of the smallest ball centered at p that intersects two non-incident features of X (where each of two features might be a vertex, segment or facet). $lfs()$ is defined for all points in X, it satisfies a 1-Lipschitz condition, i.e., for any two points p and q in X, $lfs(p) \leq lfs(q) + |p - q|$.

The definition of *input angle* is from Cheng et al. [20]. Simply speaking, an input angle of the PLC X is any angle formed by two incident segments, or a segment and a facet, or the dihedral angle formed by two incident facets. Let θ_m be the smallest acute input angle of X. In case there is no acute angle, set $\theta_m = 90°$.

Lemma 1 shows that in the output mesh, the length of the shortest edge of each vertex is bounded by $\alpha_2 H()$ and $lfs()$.

Lemma 1. *Let v be a vertex of the output mesh, and let p be the vertex closest to v in the output mesh. Then:*

$$|v - p| \geq \min\{\alpha_2 H(v), C\alpha_2 H(p_v), lfs(v)\}, \tag{1}$$

where $C = \sin\theta_m/\sqrt{2}$.

Proof. We prove this lemma by enumerating all cases which will result in the presence of v and p, respectively deriving the length bound for $|v - p|$ in each case, and taking the minimum in the end.

Assume v is an input vertex. If p is also an input vertex, then $|v - p| \geq lfs(v)$. Now assume p is an inserted vertex. If $v \in P(p)$, then $|v - p| > \alpha_2 H(v)$, otherwise, $|v - p| \geq lfs(v)$ (since v is disjoint with the segment or facet on which p lies).

Assume p is an input vertex and v is an inserted vertex. If $p \in P(v)$, then $|v - p| = |v - p_v| > \alpha_2 H(p_v)$; otherwise $|v - p| \geq lfs(v)$ (since p is disjoint with the segment or facet on which v lies).

In the following, we examine the cases where both v and p are inserted vertices. The notation $p \prec v$ means p is inserted before v.

Assume v is found by R1. Let s be the segment on which v lies.

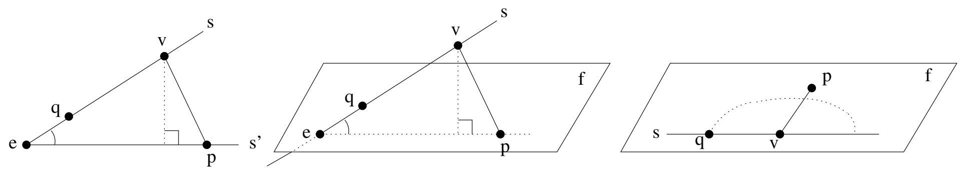


Figure 3. Suppose v is found by R1, vp is the shortest edge connected at v, q illustrates the possible location of the parent of v.

(1) Assume p is found by R1. Let s' be the segment on which p lies.

(1a) s and s' are coincident. If $p \prec v$, then $|v-p| = |v-p_v| > \alpha_2 H(p_v)$, otherwise, $|v-p| > \alpha_2 H(v)$.

(1b) s and s' are disjoint. Then $|v-p| \geq lfs(v)$.

(1c) s and s' share a common input vertex e (see Fig. 3 left). Let θ be the angle formed by s and s'. Note $\theta < 90°$ (since p is the closest vertex to v). Then $|v-p| \geq |v-e|\sin\theta \geq |v-p_v|\sin\theta > \alpha_2 H(p_v)\sin\theta_m$.

(2) Assume p is found by R2 and let f be the facet on which p lies.

(2a) s and f are disjoint. Then $|v-p| \geq lfs(v)$.

(2b) s and f share one common input vertex e (see Fig. 3 middle). Let θ be the angle formed by s and line segment eq. Note $\theta < 90°$. Then $|v-p| \geq |v-e|\sin\theta \geq |v-p_v|\sin\theta > \alpha_2 H(p_v)\sin\theta_m$.

(2c) s belongs to f (see Fig. 3 right). Suppose $p \prec v$. Then p does not encroach upon the subsegment which v splits, hence $|v-p| > |v-p_v|$. However this is not possible since p_v is closer to v than p is. We thus conclude that $v \prec p$. Then $|v-p| > \alpha_2 H(v)$.

(3) Assume p is found by R3. Similar to the case of (2c), only $v \prec p$ is possible. Then $|v-p| > \alpha_2 H(v)$.

For the next three cases, (4) - (6), we assume v is found by R2, and let f be the facet on which v lies:

(4) Assume p is found by R1. Let s be the segment on which p lies.

(4a) s and f are disjoint. Then $|v-p| \geq lfs(v)$.

(4b) s and f intersect at exactly one input vertex e (refer to Fig. 3 middle, switch the positions of v and p), then let θ be the input angle formed by s and f. Note $\theta < 90°$, and $|v-p| \geq |v-e|\sin\theta \geq |v-p_v|\sin\theta > \alpha_2 H(p_v)\sin\theta_m$.

(4c) s is part of f (refer to Fig. 3 right, switch the positions of v and p). If $p \prec v$, then $|v-p| \geq |v-p_v| > \alpha_2 H(p_v)$; otherwise, their exists a $q \in s$, such that q is either an input vertex or $q \prec v$ (see below), then $|v-p| \geq |v-q|/\sqrt{2} \geq |v-p_v|/\sqrt{2} > \alpha_2 H(p_v)/\sqrt{2}$.

Now we show that the vertex q exists. It can be found by the following iterative process: initialize $i := 0$, $q_0 := p$; (i) let q_{i+1} be the endpoint of the subsegment split by q_i which is closer to v; if q_{i+1} is an input vertex or $q_{i+1} \prec v$, then let $q := q_{i+1}$ and return; otherwise, let $i := i+1$ and goto (i). The iterative process will terminate since R1 has a priority higher than R2.

(5) Assume p is found by R2, and let f' be the facet on which p lies.

(5a) f and f' are coincident. If $p \prec v$, then $|v-p| = |v-p_v| > \alpha_2 H(p_v)$, otherwise, $|v-p| > \alpha_2 H(v)$.

(5b) f and f' are disjoint. Then $|v-p| \geq lfs(v)$.

(5c) f and f' intersect at an input vertex e (refer to Fig. 3 middle). Let θ be the input angle formed by line segments ev and ep. Note $\theta < 90°$, and $|v-p| \geq |v-e|\sin\theta \geq |v-p_v|\sin\theta > \alpha_2 H(p_v)\sin\theta_m$.

(5d) f' and f intersect at a common segment s, let θ be the input dihedral angle formed by f and f'. Note $\theta < 90°$. Using the same arguments in case (4c), there is a q on s which is either an input vertex or $q \prec v$, and $|v-q| > \alpha_2 H(p_v)/\sqrt{2}$. Then $|v-p| \geq |v-q|\sin\theta > \alpha_2 H(p_v)\sin\theta_m/\sqrt{2}$.

(6) Assume p is found by R3. If $p \prec v$, then let $q \in f$ be either an input vertex or $p \prec v$ (such q can be found by using the similar iterative process in case (4) and the fact that R2 has a higher priority than R3). Then $|v-p| \geq |v-q|/\sqrt{2} \geq |v-p_v|/\sqrt{2} > \alpha_2 H(p_v)/\sqrt{2}$. Otherwise, $v \prec p$, $|v-p| > \alpha_2 H(v)$.

Assume v is found by R3. If $p \prec v$, then $|v-p| = |v-p_v| > \alpha_2 H(p_v)$, otherwise, we have the following cases:

(7) Assume p is found by R1. Let s be the segment on which p lies. Similar to case (4), there is $q \in s$ such that q is either an input vertex or $q \prec v$. Then $|v-p| \geq |v-q|/\sqrt{2} \geq |v-p_v|/\sqrt{2} > \alpha_2 H(p_v)/\sqrt{2}$.

(8) Assume p is found by R2. Let f be the facet on which p lies. Similar to case (6), there is $q \in f$ such that q is either an input vertex or $q \prec v$. Then $|v-p| \geq |v-q|/\sqrt{2} \geq |v-p_v|/\sqrt{2} > \alpha_2 H(p_v)/\sqrt{2}$.

(9) Assume p is found by R3. Then $|v-p| > \alpha_2 H(v)$.

We have now checked all cases which can occur for v and p. In each case, $|v-p|$ is larger or equal to one of the bounds, namely $\alpha_2 H(v)$, $C\alpha_2 H(p_v)$, or $lfs(v)$. The smallest value of the constant C appears in case (5d), namely, $C = \sin\theta_m/\sqrt{2}$.

Finally, we can replace $R1$ by $R1^*$ in each of the above cases, and the bounds do not change. Thus Inequality (1) holds for $R1^*$ as well. ■

Since $H()$ and $lfs()$ are always positive, $|v-p|$ will not be zero as long as $\alpha_2 > 0$. Theorem 1, that guarantees the termination of the algorithm, is a direct consequence of Lemma 1.

Theorem 1. *The algorithm terminates if* $\alpha_2 > 0$. ■

3.2. Constrained Delaunay Property

By using $R1^*$ instead of $R1$ to split encroached segments, we show that the resulting mesh can be guaranteed to have constrained Delaunay property.

Theorem 2. *There exists a positive constant* D_S*, such that for* $\alpha_2 \leq D_S$*, by repeatedly using* $R1^*$ *to split encroached segments, one can guarantee that all subsegments are not encroached, thus they are Delaunay segments.*

Proof. Notice that if the point-accepting rule is ignored, then the repeated applying of $R1^*$ will terminate and all encroached segments will be split [33]. This is the essential oberservation to prove the theorem. Let the point-accepting rule always accept the points generated by $R1^*$, provided they lie on encroached segments (or subsegments).

For an encroached segment S_0, whose endpoints are a and b, $R1^*$ will generate a point $v \in S_0$. One can choose a constant D_{S_0}, such that

$$D_{S_0} < \min\{\frac{|a-v|}{H(a)}, \frac{|v-b|}{H(b)}\}.$$

Then, if $0 < \alpha_2 \leq D_{S_0}$, v will be accepted and S_0 is split into two subsegments. Similarly, for any encroached segment (or subsegment) S_i, one can choose a constant D_{S_i}, such that S_i will be split if $0 < \alpha_2 \leq D_{S_i}$. We thus can form a finite set of constants $\{D_{S_0}, D_{S_1}, \cdots, D_{S_m}\}$. The theorem is proved by choosing $D_S = \min\{D_{S_0}, D_{S_1}, \cdots, D_{S_m}\}$. ■

Once all segments are Delaunay (they are also strongly Delaunay since they are not encroached upon), the existence of a CDT is guaranteed [14]. A CDT can be constructed by using the facet recovery algorithm of [33] or the flip algorithm of [17].

3.3. Mesh Quality

In this section, we consider the output mesh quality. Our goal is to show that the algorithm is able to create a mesh with most of the tetrahedra having their radius-edge ratio bounded, where only a few poor-quality tetrahedra remain in well defined locations.

We say a tetrahedron is *skinny* if its radius-edge ratio is smaller than B. A vertex v is called *sharp* if there are two segments or a segment and a facet intersecting at v forming an acute angle. A segment s is called *sharp* if it contains a sharp vertex or if there are two facets sharing s forming an acute dihedral angle; a facet f is called *sharp* if it contains a sharp segment or there is another segment or facet adjacent to f forming an acute angle or acute dihedral angle.

Theorem 3. *Suppose the quality bound B is larger than 2 and the constrained Delaunay property is maintained. Then there exists a constant D_Q, such that, for $\alpha_2 = D_Q$, the internal tetrahedra have a radius-edge ratio smaller than B. The circumcenter of any skinny tetrahedron is within a distance of $\sqrt{2}\alpha_2 H(p)$ from p, where p is a sharp vertex or a vertex inserted on a sharp segment or a sharp facet.*

Proof. If there is no acute input angle and $B > 2$, then the basic Delaunay refinement algorithm guarantees that the distance between any output vertex v and its nearest neighbor is at least $\frac{lfs(v)}{D+1}$, where $D > 1$ is a fixed constant (Theorem 6 in [13]). Our theorem can be proved if D_Q is chosen sufficiently small such that the inequality

$$D_Q < \frac{lfs(v)}{H(v)(D+1)}$$

holds for each output vertex v, i.e., the protecting-ball of v is always empty and no later generated vertex will be rejected.

Now consider the case where there are acute input angles. The theorem can be proved by a "bad tetrahedra elimination" procedure. Since the constrained Delaunay property is required, rule $R1^*$ is used instead of $R1$ in the following runs of the algorithm. At the initialization,

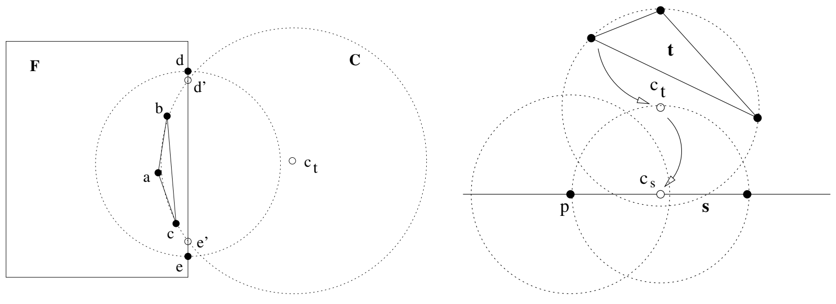


Figure 4. **Left**: The case $t \in \Phi_1$ (defined in Section **??**), one case of the existence of an encroached segment. The triangle abc is a face of t and lies inside a facet F. C and c_t are the circumcircle and circumcenter of abc, respectively. de is the segment of F which is both encroached upon and visible by a. **Right**: The case $t \in \Phi_4$. The circumcenter c_s of s lies inside the protecting ball of p, one of the endpoints of s.

choose $0 < D_{Q_0} \leq D_S$ and run the algorithm with $\alpha_2 = D_{Q_0}$. The output mesh contains skinny tetrahedra (otherwise we are done). Let t be such a skinny tetrahedron, c_t be its circumcenter. t can be categorized into one of the four sets $\Phi_1, ..., \Phi_4$, which are:

- Φ_1 contains all t such that there is a corner of t encroaches upon at least a segment or a subface.
- Φ_2 contains all t such that c_t lies inside the mesh and does not encroach upon any segment or subface.
- Φ_3 contains all t such that c_t encroaches upon a segment (or a subface) which is non-sharp.
- Φ_4 contains all t such that c_t encroaches upon a sharp segment (or a sharp subface).

Consider a tetrahedron $t \in \Phi_1$. Let v be the corner of t, and let s be the segment (or subface) which is encroached upon by v and s is visible by v (see Fig. 4 left). s is non-conforming Delaunay, and s is not split because the point c_s generated by $R1^*$ (or $R2$) is rejected by the point-accepting rule, i.e., c_s lies inside at least one of the protecting balls of its corners.

t will be eliminated if s is split. This can be achieved by shrinking the protecting balls of the endpoints of s such that c_s lies outside all of them. It is possible to choose a sufficiently small $D_{Q_1} > 0$, such that all tetrahedra of Φ_1 can be removed by running the algorithm with $\alpha_2 = D_{Q_1}$. The newly inserted vertices may create new poor quality tetrahedra which can be classified into Φ_2, Φ_3, and Φ_4.

Consider a tetrahedron $t \in \Phi_2$, c_t is rejected by some protecting balls of existing vertices at the neighborhood of t. t can be eliminated by shrinking these protecting balls such that c_t lies outside all of them. It is possible to choose a sufficiently small D_{Q_2}, $0 < D_{Q_2} \leq D_{Q_1}$, such that no $t \in \Phi_2$ can survive after running the algorithm with $\alpha_2 = D_{Q_2}$. There are possibly remaining poor quality tetrahedra of Φ_3 and Φ_4.

Consider a tetrahedron $t \in \Phi_3$, the circumcenter of the segment (or subface) s is rejected by lying inside some protecting balls of its endpoints. s will be split by shrinking these protecting balls. Consequently, either t gets eliminated during the split of s, or c_t does not encroach upon any segment or subface and is accepted for insertion, or t becomes a tetrahedron of Φ_2. It is possible to choose a sufficiently small D_{Q_3}, $0 < D_{Q_3} \leq D_{Q_2}$, such that no $t \in \Phi_2 \cup \Phi_3$ can survive after running the algorithm with $\alpha_2 = D_{Q_3}$. Now the possibly remaining poor quality tetrahedra can only belong to Φ_4.

If $t \in \Phi_4$, then the point c_s generated by $R1^*$ (or $R2$) of an encroached segment (or subface) s is rejected by lying inside some protect balls of the endpoints of s (see Fig. 4 right). Let p be such a vertex. Then $|c_t - c_s| < |c_s - p| < \alpha_2 H(p)$. Hence $|c_t - p| < \sqrt{2}\alpha_2 H(p)$. ■

3.4. Mesh Conformity

Next, we consider the mesh conformity with respect to the sizing function H. For each vertex v, let $S(v)$ and $L(v)$ denote the lengths of the shortest edge and the longest edge among all edges containing v, respectively. We are interested in the values $\frac{S(v)}{H(v)}$ and $\frac{L(v)}{H(v)}$. Theorem 4 gives bounds for these quantities at those output vertices, where the local mesh quality is satisfied.

Theorem 4. *Suppose all tetrahedra containing v have their radius-edge ratio bounded by B (given on input), and the rule R3.2 is not applicable on any of them, then:*
(i) $\frac{S(v)}{H(v)} \geq \min\{\alpha_2, C\alpha_2\frac{H(p_v)}{H(v)}, \frac{lfs(v)}{H(v)}\}$.
(ii) $\frac{L(v)}{H(v)} \leq 2\alpha_1$;

Proof. The first claim follows directly from the inequality (1). Let t be a tetrahedron which contains v and has a longest edge of length $L(v)$. Moreover, let r be the circumradius of t. Then: $\frac{L(v)}{2} \leq r \leq \alpha_1 H(v) \Longrightarrow \frac{L(v)}{H(v)} \leq 2\alpha_1$. ■

When the local mesh quality is satisfied, and the mesh is saturated. then Theorem 4 shows that the mesh conformity at each vertex v is related to H, α_1, α_2, and lfs. Specifically:

- H, α_2, and lfs together decide the lower bound of the mesh conformity.
- The term $\frac{H(p_v)}{H(v)}$ indicates that H should not vary too much in v's neighborhood, e.g., H is 1-Lipschitz. The term $\frac{lfs(v)}{H(v)}$ indicates that H is constrained by lfs which is dependent upon the boundary of the CDT.
- α_2 plays a contradictory role between mesh quality and mesh conformity. It needs to be small in order to guarantee the mesh quality. While it is desired to be as large as possible for good mesh conformity.
- α_1 limits the length of the longest output edge connected at v. It directly controls the resulting mesh size, i.e., the smaller it is, the bigger the mesh size will be.

4. SPECIFYING SIZING FUNCTIONS

Our algorithm needs a sizing function H which is defined over the mesh domain and specifies the local mesh size, e.g., the desired edge length or element density. The data of a sizing function can be based on either a priori known information or on a posteriori error estimation. If no sizing function is given, a simple approach is proposed to automatically derive a sizing function from the boundary data of the CDT. Alternatively, a background mesh, whose vertices contain the size information, can be supplied along with the CDT.

4.1. A Sizing Function Derived from the CDT

It is possible that an appropriate sizing function may not be available in advance, for example, when initially generating a mesh for adaptive simulation. In such a case, we propose a simple method to automatically derive a sizing function from the input CDT:

- If p is a point of the input CDT, then $H(p) = lfs(p)$.
- Otherwise $H(p)$ is interpolated from its adjacent vertices by the Shepard interpolation [3], where the weights are set to be the second inverse power of the distances, i.e.,

$$H(p) = \frac{\sum_{i=1}^{n} |p - v_i|^{-2} H(v_i)}{\sum_{i=1}^{n} |p - v_i|^{-2}}. \tag{2}$$

  here v_i is a vertex connecting to p in the current mesh.

The above method guarantees that every point of the mesh is assigned a size. If p is a vertex of the initial CDT, then $lfs(p)$ can be efficiently computed by searching locally the smallest distance of the nearest vertex, subsegment, and subface. For each inserted vertex p, Equation (2) has the effect that the closest node has the biggest influence on the size of p. Notice that H can be computed on the fly, i.e., each new point can be assigned its size after it is inserted. There are similar approaches [10, 11, 31], but they all require additional data structures, e.g., a background mesh or a KD-tree.

Our method attempts to approximate the local feature size on the mesh nodes it is easier and much more efficient than to comput $lfs()$ (see Fig. 5). Due to the simplicity of the method, the obtained H may not be smooth and thus may fail to result in a good quality mesh. Nevertheless, the mesh quality can be improved by choosing an appropriate value for α_2.

4.2. Use of a Background Mesh

If H is known in advance, then the most popular and flexible way for specifying H is through a background mesh whose vertices or elements encode the information about the desired mesh size. The background mesh can be any grid structure (such as a uniform grid or Octree) or an unstructured mesh (such as a CDT).

We use an unstructured mesh as the background mesh. Hence it can be the initial CDT or a tetrahedral mesh obtained at the previous iteration in an adaptive process. At any point p of the current mesh, $H(p)$ can be obtained by means of interpolation in the background mesh:

- locate p in a tetrahedron t which contains p;
- compute $H(p)$ as the P^1 interpolation of the sizes $H(p_i)$ at the vertices p_i of t.

5. EXAMPLES

The algorithm has been integrated into TetGen – a quality Delaunay tetrahedral mesh generator [37]. The input can be either a PLC or a CDT. A sizing function H can be optionally specified through a background mesh. Parameters B, α_1, and α_2 are all adjustable at runtime. Each of them has a default value ($B = 2.0, \alpha_1 = \sqrt{2}, \alpha_2 = 0.5$).

The next two examples (Fig. 6 and 7) were built for analysis purposes. We study the effects of using different combinations of the parameters (B, α_1, and α_2) and compare the results according to the analysis in Section 3.

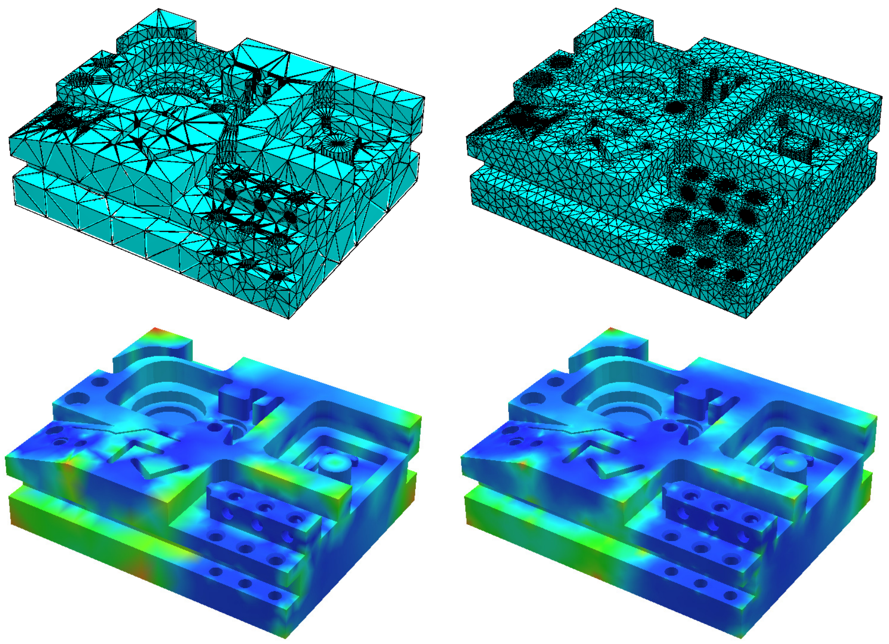
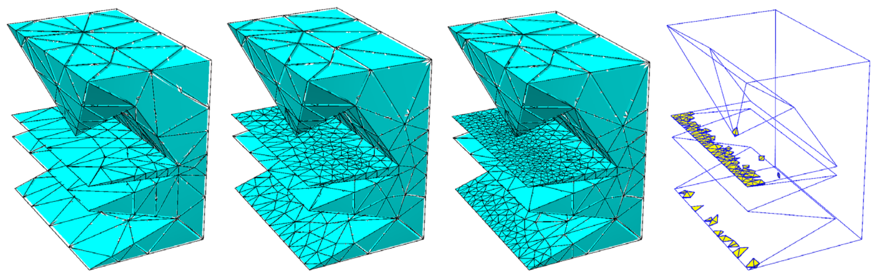


Figure 5. Sizing function derived from local feature sizes and Shepard interpolation. The top two pictures show a CDT (left, $9k$ nodes, $27k$ tetrahedra) and the refined mesh (right, $124k$ nodes, $533k$ tetrahedra) of a mechanical part. At the bottom, the density of sizing functions are shown. Left is the initial H function obtained by the local feature sizes of the input points; on the right is the final H function derived by our method. The CPU time of mesh refinement, including the H-generation, is less than 15 sec. (on a Intel machine clocked at 3.60GHz).

The geometry of the first example (Fig. 6) contains many acute input angles. The basic Delaunay refinement algorithm [13] generally will not terminate on such input. Fig. 6 shows sequences of meshes created by our algorithm with various combinations of (B, α_2). The sizing functions are automatically derived. Remaining poor-quality tetrahedra are plotted for selected meshes. The size statistics of each mesh is given in the form $n_v/n_t/n_b$, where n_v denotes the number of nodes, n_t denotes the number of tetrahedra, and n_b denotes the number of remaining poor-quality tetrahedra. The results validate our claim (Theorem 3) on the mesh quality, i.e., for an appropriate α_2, most of tetrahedra have a bounded radius-edge ratio, poor-quality tetrahedra are all close to small input angles. Notice that few slivers may remain in the volume, they can be removed by a mesh smoothing step.

Fig. 7 shows three refined meshes resulting from a unit cube. Each mesh is obtained by specifying a piecewise smooth sizing function through a background mesh. The parameters ($B = 2.0$, $\alpha_1 = \sqrt{2}$, and $\alpha_2 = 0.05$) were chosen in such a way that the hypothesizes of Theorem 4 are satisfied. Table I lists the statistics of the ratios $\frac{S(v)}{H(v)}$ and $\frac{L(v)}{H(v)}$ (defined in Theorem 4) at mesh vertices. From all three meshes, L_v is strictly bounded by $2\alpha_1$, which verifies claim (ii) of Theorem 4. The smallest S_v is much larger than α_2. The meshes well conform to the specified sizing functions.



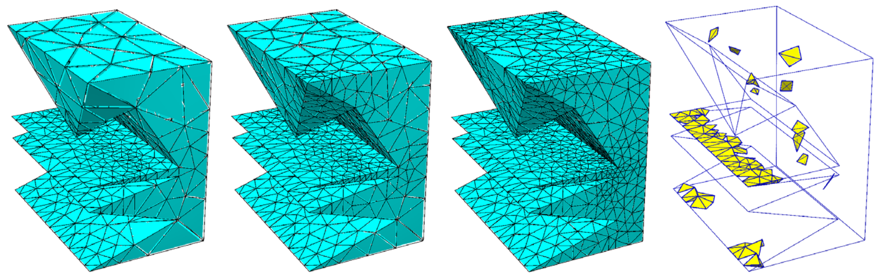


Figure 6. Test case for various combinations of parameters (B, α_2).

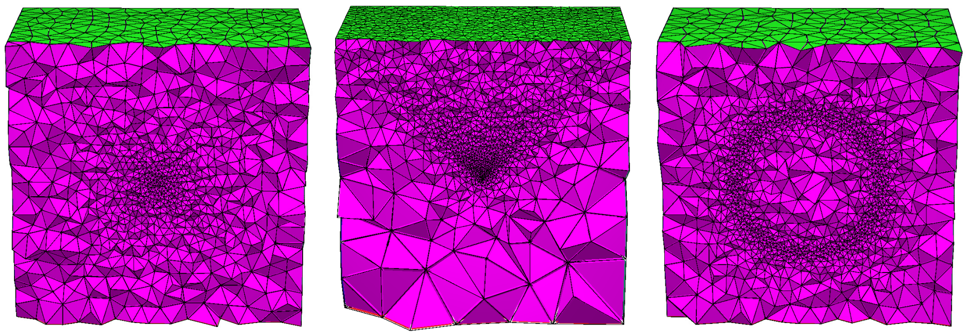


Figure 7. Meshes created by specifying different sizing functions.

Table I. Statistics of the ratios $S_v = \frac{S(v)}{H(v)}$ and $L_v = \frac{L(v)}{H(v)}$ (defined in Theorem 4) at mesh vertices of the meshes shown in Fig. 7.

	Mesh 1		Mesh 2		Mesh 3	
	S_v	L_v	S_v	L_v	S_v	L_v
< 0.5	0	0	0	0	0	0
$0.5 - 1/\sqrt{2}$	58	0	0	0	0	0
$1/\sqrt{2} - 1$	3221	1	283	0	0	0
$1 - \sqrt{2}$	15062	113	10778	14	1927	49
$\sqrt{2} - 2$	4246	3867	1187	1044	94186	12594
$2 - 2\sqrt{2}$	0	18606	0	11190	12276	95746
$> 2\sqrt{2}$	0	0	0	0	0	0

To further demonstrate both the robustness and efficiency of our algorithm, we tested it on two complicated and challenging geometries taken from Inria's large repository `http://www-rocq1.inria.fr/gamma`.

Fig. 8 shows a meshed result of a Boeing 747. The input is a surface mesh of the plane skin (Fig. 8 (a), $2,874$ nodes, $5,738$ triangles) enclosed by a bounding box. A smooth sizing function was used (Fig. 8 (b)) to govern the desired mesh size, which is small near the plane skin and gradually increasing toward the bounding box. The resulting mesh (Fig. 8 (c) - (e) $490,692$ nodes, $2,709,770$ tetrahedra) was generated with parameters: $B = 2.0, \alpha_1 = 0.5, \alpha_2 = 0.25$.

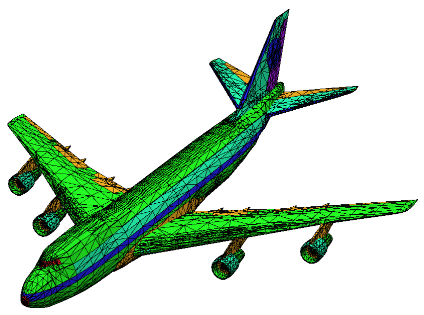
Fig. 9 shows the tetrahedral mesh of a car and the Navier-Stokes solution of a channel flow simulation based on the mesh. The mesh domain (Fig. 9 (a)) represents a car body (surface mesh, $2,813$ nodes, $5,594$ triangles) inside a channel, the internal of the car is not meshed. The initial CDT was re-used as the background mesh for specifying the sizing function H, which was simply chosen such that the size is small on the car surface and big on the wall of the channel. The tetrahedral mesh (Fig. 9 (b) and (c), $15,783$ nodes, $866,474$ tetrahedra) was created by default parameters. A velocity field of the solution (Fig. 9 (d)) is viewed using pdelib [34].

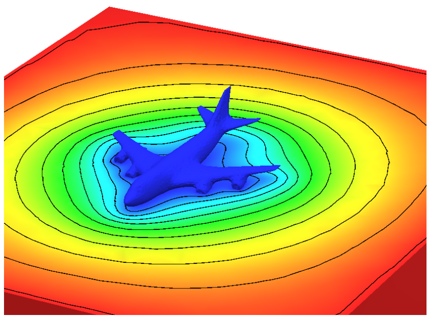
Fig. 10 shows the histograms of radius-edge ratios of the above two meshes. Both meshes contain high-quality tetrahedra in the bulk of the respective mesh domain. For example, over 94% of the tetrahedra of the Boeing 747 mesh have radius-edge ratios between 0.612 and 1.1. Only about 0.4% of the tetrahedra are bad-quality and they are all close to the input acute angles of the surface mesh (see Fig. 8 (f)).

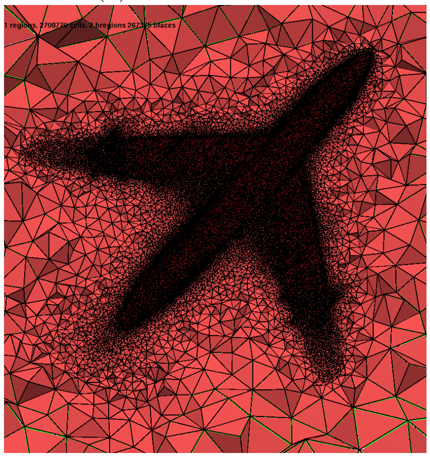
Finally, the efficiency of the algorithm is partially reflected in the CPU times elapsed during the mesh refinement. It took 59 seconds for the refinement of Boeing 747 and 21 seconds for the Car. The meshing speed is about $44.5k$ tetrahedra/sec. for the Boeing 747 and about $39.5k$ tetrahedra/sec. for the car. The times were obtained on an Intel machine with a CPU clocked at 3.60GHz.

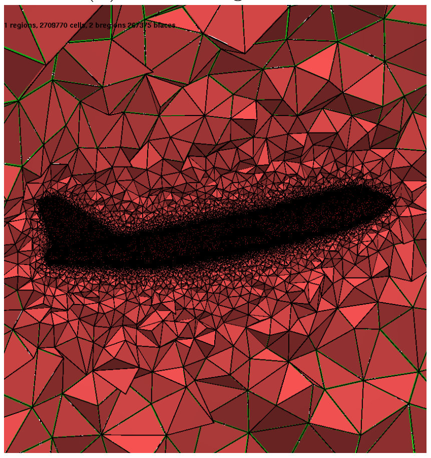
6. CONCLUSIONS AND DISCUSSION

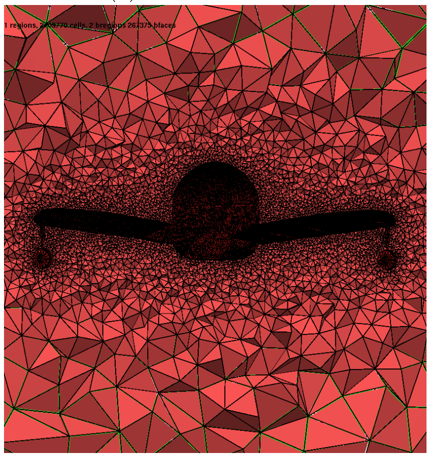
In this paper, the problem of refining constrained Delaunay tetrahedralizations is raised in the context of adaptive numerical simulations. A practical algorithm based on a Delaunay


(a) The surface mesh


(b) The sizing function


(c) Tet mesh detail


(d) Tet mesh detail


(e) Tet mesh detail

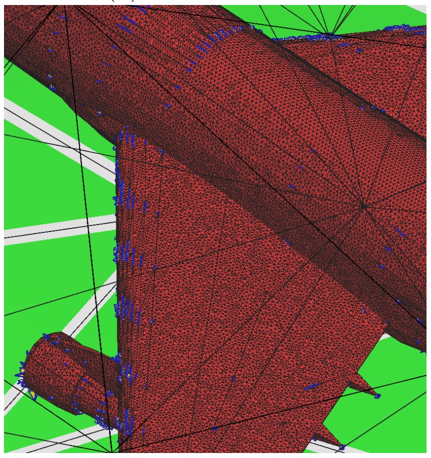
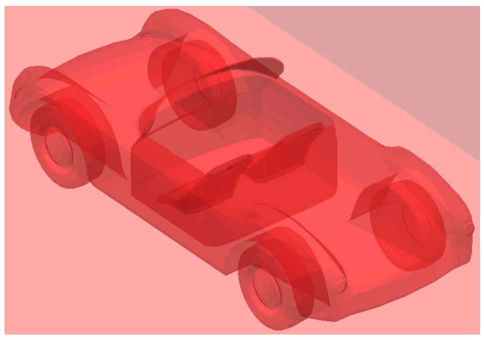
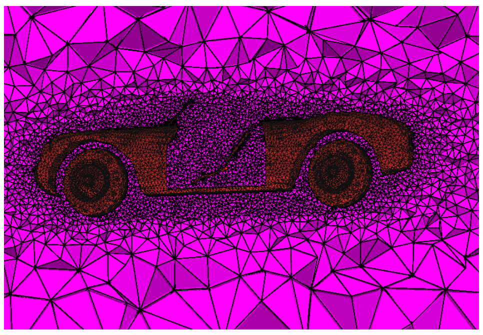
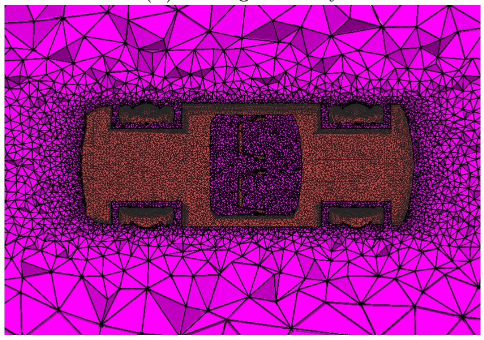

(f) Highlights of remaining bad quality tetrahedra

Figure 8. Tet Mesh of Boeing 747 (490, 692 nodes, 2, 709, 770 tetrahedra).


(a) The geometry


(b) Tet mesh detail


(c) Tet mesh detail

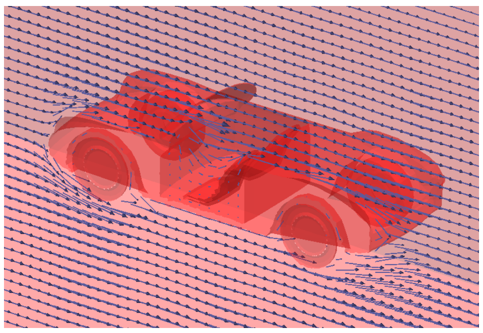

(d) The velocity field

Figure 9. Tet Mesh of Car (157, 083 nodes, 866, 474 tetrahedra) and the velocity field of a channel flow solution (by AcuSolve [36]).

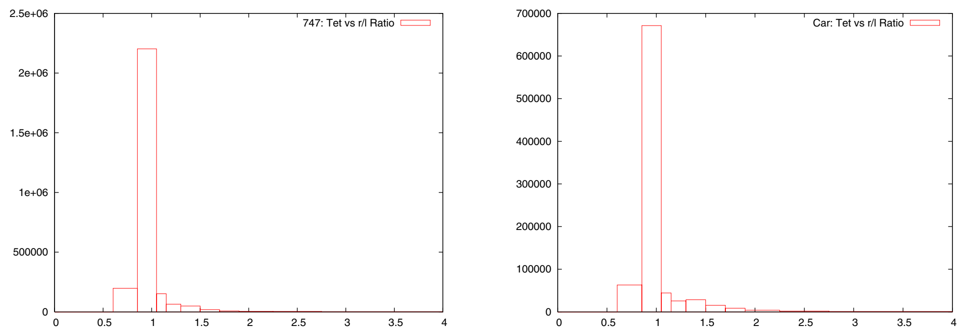


Figure 10. The radius-edge ratio histograms of meshes in Fig. 8 and Fig. 9.

refinement scheme is proposed. The algorithm eliminates the angle restrictions in the basic Delaunay refinement algorithms [9, 13] and supports adaptive refinement through a user-defined sizing function. Theoretical analysis shows that the mesh quality can be guaranteed, while good mesh conformity can be obtained for smooth sizing functions.

Let us point out some possibilities to improve the mesh quality and to reduce the mesh size. Notice that the circumcenter used in the basic Delaunay refinement scheme might not be the best position for a Steiner point. Alternatively, the off-center [29] may be considered.

Although our algorithm is proposed for refining CDTs, it can be used to refine any boundary constrained tetrahedralizations (non-CDTs) as well. Notice that the point-generating rules and the point-accepting rule do not rely on the type of input. A CDT is eligible for an efficient implementation of these rules. But is not the case for arbitrary non-CDTs, where required features might not be found locally. Whatsoever, a background grid (such as Octree) may then be used.

It would be interesting to adapt the algorithm for anisotropic H. For this purpose, the usual ways of measuring edge lengths, updating locally Delaunay property around Steiner points, and the interpolation of H must be modified by anisotropic means [23].

ACKNOWLEDGEMENTS

This work was supported by the WIAS research project `Numerical Mthods` in the topic of `numerical solution of partial differential equations`. The author would like to thank Jürgen Fuhrmann and Klaus Gärtner for the initial stimulating of this work and many helpful discussions. Speical thank to Peter Philip for the improvement of the manuscript.

REFERENCES

1. Schönhardt E, *Über die Zerlegung von Dreieckspolyedern in Tetraeder.* Math. Ann. 1928, **98**: 309–312.
2. Delaunay B, *Sur la sphére vide.* Bul. Acad. Sci. URSS, Class. Sci. Nat., 1934, 793–800.
3. Shepard D *A Two-Dimensional Interpolation Function for Irregularly Space Data.* In Proc. 23th Nat. Conf. ACM, 1968, 517–524.
4. Babuška I, Aziz, A K, *On the Angle Condition in the Finite Element Method.* SIAM Journal on Numerical Analysis, 1976, **13**(2): 214–226.
5. Chew P L, *Guaranteed-Quality Triangular Meshes.* Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.
6. Chew P L, *Guaranteed-Quality Delaunay Meshing in 3D.* In Proc. 19th Annu. Sympos. Comput. Geom., 1997, 274–280.
7. George P L, *Automatic Mesh Generation. Application to Finite Element Methods.* Wiley, 1991.
8. Ruppert J, Seidel R, *On the Difficulty of Triangulating Three-dimensional Non-convex Polyhedra.* Discrete Comput. Geom., 1992, **7**: 227–253.
9. Ruppert J, *A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation.* J. Algorithms, 1995, **18**(3): 548–585.
10. Weatherill N P, Hassan O, *Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints.* Int. J. Numer. Meth. Engng, 1994, **37**: 2005–2039.
11. Frey P J, Borouchaki H, George P L, *Delaunay Tetrahedralization Using an Advancing Front Approach.* In Proc. of 5th Intl. Meshing Roundtable, 1996.
12. Miller G L, Talmor D, Teng S H, Walkington N, Wang H, *Control Volume Meshes using Sphere Packing: Generation, Refinement and Coarsening.* In Proc. of 5th Intl. Meshing Roundtable, 1996.
13. Shewchuk J R, *Tetrahedral Mesh Generation by Delaunay Refinement.* In Proc. 14th Annu. Sympos. Comput. Geom., 1998.
14. Shewchuk J R, *A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations.* In Proc. 14th Annu. Sympos. Comput. Geom., 1998.

15. Shewchuk J R, *Mesh Generation for Domains with Small Angles.* In Proc 16th Annu. ACM Sympos. Comput. Geom., 2000.
16. Shewchuk J R, *What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures.* In Proc. 11th Intl. Meshing Roundtable, 2002.
17. Shewchuk J R, *Updating and Constructing Constrained Delaunay and Constrained Regular Triangulations by Flips.* In Proc. 19th Annu. Sympos. Comput. Geom., 2003.
18. Shewchuk J R, *General-Dimensional Constrained Delaunay and Constrained Regular Triangulations.* To appear in Discrete & Computational Geometry.
19. Cheng S W, Dey, T K, Edelsbrunner H, Facello M A, Teng S H, *Sliver Exudation.* J. ACM, 1999, **47**: 883–904.
20. Cheng S W, Dey T K, Ramos E A, Ray T, *Quality Meshing for Polyhedra with Small Angels.* In Proc. 20th Annu. ACM Sympos. Comput. Geom., 2004.
21. Edelsbrunner H, Li X Y, Miller G, Stathopoulos A, Talmor D, Teng S H, Üngör A, Walkington N, *Smoothing Cleans up Slivers.* In. Proc. 32th Annu. ACM Sympos. Theory of Computing, 2000, 273–277.
22. Edelsbrunner H, Guoy D, *An Experimental Study of Sliver Exudation.* Engineering with Computers, 2002, **18**: 299–240.
23. Frey P J, George P L, *Mesh Generation. Application to Finite Elements.* Hermès, Paris, 2000.
24. Fuhrmann J, Langmach H, *Stability and Existence of Solutions of Time-Implicit Finite Volume Schemes for Viscous Nonlinear Conservation Laws.* Appl. Numer. Math., 2001, **37**(1-2): 201–230.
25. Du Q, Wang D, *Tetrahedral Mesh Generation and Optimization Based on Centroidal Voronoï Tessellations.* Int. J. Nummer. Meth. Engng, 2003, **56**: 1355–1373.
26. Rambau J, *On the Generalization of Schönhardt's Polyhedron.* MSRI Preprint, 2003.
27. Pav S, Walkington N, *A Robust 3D Delaunay Refinement Algorithm.* In Proc. 13th Intl. Meshing Roundtable, 2004.
28. Pav S, Walkington N, *Delaunay Refinement by Corner Lopping.* In Proc. 14th Intl. Meshing Roundtable, 2005.
29. Üngör A, *Off-centers: A New Type of Steiner Points for Computing Size-Optimal Guaranteed-Quality Delaunay Triangulations.* In Proc. LATIN, 2004, 152–161.
30. Persson P O, Strang G, *A Simple Mesh Generator in MATLAB.* SIAM Review, 2004, **46**(2): 329–345.
31. Alliez P, Cohen-Steiner D, Yvinec M, Desbrun M, *Variational Tetrahedral Meshing.* ACM Transactions on Graphics, 2005, **24**(3): 617–625.
32. Oudot S, Rineau L, Yvinec M, *Meshing Volumes Bounded by Smooth Surfaces.* In Proc. 14th Intl. Meshing Roundtable, 2005.
33. Si H, Gärtner K, *Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations.* In Proc. 14th Intl. Meshing Roundtable, 2005, 147–163.
34. pdelib2, *Tool Box for Solving Partial Differential Equations.* `http://www.wias-berlin.de/software/pdelib`.
35. WIAS-SHARP, *Surface Hardening Program.* `http://www.wias-berlin.de/software/sharp`.
36. AcuSolve, *a Finite Element Incompressible Flow Solver.* `http://www.acusim.com`.
37. TetGen, *A Quality Tetrahedral Mesh Generator.* `http://tetgen.berlios.de`.