1.2 Mesh Generation

1.2.1 Simplicial Complex

**Definition 1.2 (Simplex).** An $n$-simplex with vertices \{v_0, v_1, \ldots, v_n\} is defined as the point set
\[ [v_0, v_1, \ldots, v_n] := \left\{ \lambda_0 v_0 + \lambda_1 v_1 + \cdots + \lambda_n v_n : \sum_{i=0}^{n} \lambda_i = 1, \lambda_i \geq 0 \right\}. \]

**Definition 1.3 (Simplex Orientation).** An $n$-simplex with vertices \{v_0, v_1, \ldots, v_n\} has two orientations. Let $\sigma \in S_n$ be a permutation, the simplex $[v_0, v_1, \ldots, v_n]$ and $[v_{\sigma(0)}, v_{\sigma(1)}, \ldots, v_{\sigma(n)}]$ has the same orientation if and only if $\sigma$ is an even permutation. Otherwise, if $\sigma$ is an odd permutation, then
\[ [v_{\sigma(0)}, v_{\sigma(1)}, \ldots, v_{\sigma(n)}] = -[v_0, v_1, \ldots, v_n]. \]

**Definition 1.4 (Boundary).** Suppose $[v_0, v_1, \ldots, v_n]$ is an $n$-simplex, its boundary is defined as
\[ \partial [v_0, v_1, \ldots, v_n] = \sum_{i=0}^{n} (-1)^i [v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n]. \]

**Definition 1.5 (Simplicial Complex).** Suppose $C$ is a set of simplexes, such that if $\sigma_i$ and $\sigma_j$ are in $C$, then their intersection $\sigma_i \cap \sigma_j$ is either empty, or a simplex in $C$.

**Definition 1.6 (Poincaré Duality).** Given a cell decomposition of a manifold $\Omega$, denoted as $\mathcal{V}$, its Poincaré dual is a simplicial complex $\mathcal{T}$, each cell $W_i$ corresponds to a 0-simplex $v_i$, furthermore
\[ W_i \cap W_j \cap \ldots W_k \neq \emptyset \iff [v_1, v_2, \ldots, v_k] \in \mathcal{T}, \]

1.2.2 Convex Hull

**Definition 1.7 (Convex Hull).** The convex hull of $P$ is the intersection of all convex sets, which contains $P$.

Given a discrete point set $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^3$, $n > 3$, the convex hull of $P$ is represented by its boundary, which is a 2 dimensional complex, in particular, a
triangulated polyhedral surface. The convex hull \(\text{conv}(P)\) can be computed using the incremental algorithm.

**First Step.** We construct a 3-simplex, namely a tetrahedron using \(p_1, \ldots, p_4\). The signed volume of a tetrahedron \([p_1, p_2, p_3, p_4]\) is given by

\[
V([p_1, p_2, p_3, p_4]) = \frac{1}{6} \det \begin{bmatrix}
x_1 & x_2 & x_3 & x_4 \\
y_1 & y_2 & y_3 & y_4 \\
z_1 & z_2 & z_3 & z_4 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

If the signed volume is negative, then we exchange \(p_1\) and \(p_2\). The convex hull \(\text{conv}(p_1, p_2, p_3, p_4)\) is assigned by this tetrahedron.

**The k-th Step.** We process one more vertex \(v_k\) and enlarge \(\text{conv}(p_1, \ldots, p_{k-1}, p_k)\) to \(\text{conv}(p_1, \ldots, p_k)\). We go through each of the triangular faces of \(\text{conv}(p_1, \ldots, p_{k-1})\), \([p_a, p_b, p_c]\). If the signed volume of the tetrahedron \([p_k, p_a, p_b, p_c]\) is negative, then we say the face is **visible** from \(p_k\), otherwise the face is **invisible**. If all the faces are invisible, then the point \(p_k\) is inside \(\text{conv}(p_1, \ldots, p_{k-1})\). In this case, \(\text{conv}(p_1, \ldots, p_k)\) is identical to \(\text{conv}(p_1, \ldots, p_k)\). Otherwise, there are some visible faces. We remove all the visible faces, the left invisible faces form a simply connected polyhedral surface with a single boundary. Each edge on the boundary is adjacent to one visible face and one invisible face, therefore the boundary is called the **silhouette**. We connect each silhouette edge with \(p_k\) to form a new triangular face, with consistent orientation. This gives \(\text{conv}(p_1, \ldots, p_k)\).

**Iteration.** We repeat the procedure to enlarge the convex hull, until we exhaust all the points in \(P\).

The incremental convex hull algorithm can be generalized to arbitrary dimension directly. We can improve its efficiency by carefully design the order of \(p_i\)'s, such that \(p_k\) is the furthest point from \(\text{conv}(p_1, \ldots, p_{k-1})\). Namely, at each step, we choose the new point to maximize the volume of the next convex hull. This will increase the chance for a new point to be already inside the convex hull of the preceding points in the sequence.
1.2.3 Delaunay Triangulation and Voronoi Diagram

Figure 1.1 shows Delaunay triangulation on a planar domain (left frame) and on the surface (right frame). For a planar Delaunay triangulation, we can draw a circum-circle for each triangle, then the interior of the circum-circle doesn’t contain any vertex of the triangulation. Similarly, for a surface Delaunay triangulation, all the edges are geodesic segments. If the triangulation is dense enough, for each geodesic triangle, we can draw a unique geodesic circum-circle, the interior of the circum-circle doesn’t contain any vertex of the whole triangulation. Delaunay triangulation can be defined using this *empty circle* property.

**Definition 1.8 (Delaunay Triangulation).** Suppose \( V = \{v_i\}_{i=1}^{k} \) is a finite set of points in \( \mathbb{R}^d \), a triangulation \( \mathcal{T} \) of \( V \) is called Delaunay, if for any \( d \)-simplex \( \sigma \) in \( \mathcal{T} \), the circum-sphere of \( \sigma \) is denoted as \( s(\sigma) \). The interior of \( s(\sigma) \) intersects \( V \) is empty.

Delaunay triangulation is dual to voronoi diagram as shown in Figure 1.1.

**Definition 1.9 (Voronoi Diagram).** Given a discrete point set \( V = \{v_i\}_{i=1}^{k} \) contained in \( \mathbb{R}^d \), the Voronoi diagram of \( V \) is a cell decomposition of \( \mathbb{R}^d \),
\[
\mathbb{R}^d = \bigcup_{i=1}^{k} W_i,
\]
where \( W_i \) is a voroni cell, defined as
\[
W_i := \left\{ x \in \mathbb{R}^d | d(x, v_i) \leq d(x, v_j), \forall j \right\}.
\]
12 1 Computational Algorithms

where \(d(x, y)\) is the Euclidean distance between \(x\) and \(y\).

![Fig. 1.2 Dualit between the Delaunay triangulation and the Voronoi diagram.](image)

The Poincaré dual of a Voronoi diagram is the Delaunay triangulation.

The algorithm for computing Delaunay triangulation can be converted to that for convex hull. Given a set of distinct points \(P = \{p_1, p_2, \ldots, p_n\}\) in \(\mathbb{R}^d\), we lift them to the graph of the parabola

\[
x_{d+1} = |x|^2 = x_1^2 + x_2^2 + \cdots + x_d^2,
\]

namely, \(q_i := (p_i, |p_i|^2) \in \mathbb{R}^{d+1}\). Then we compute the convex hull of \(Q = \{q_1, q_2, \ldots, q_n\}\). The projection of the lower hull (consisting of faces with normals pointing to the negative \(e_{d+1}\) direction) produces the Delaunay triangulation \(\mathcal{F}\) of \(P\).

This algorithm can be easily proved as follows. Suppose \([q_0, q_1, \ldots, q_d]\) is a face (a \(d\)-simplex) of the convex hull, the supporting hyper-plane through the face is \(\pi\). Then \(\pi\) intersects the parabola at an ellipsoid, whose projection is the circum-sphere of \(p_0, p_1, \ldots, p_d\). For any \(k > d\), \(q_k\) is above \(\pi\) since \(\pi\) supporting the convex hull, therefore its projection \(p_k\) is outside the circum-sphere. This shows the triangulation \(\mathcal{F}\) satisfies the empty-sphere condition, hence is Delaunay.

### 1.2.4 Delaunay Refinement

Mesh generation plays an fundamental role for solving geometric PDEs and in engineering applications. In Finite Element Analysis, the mesh quality is measured by two criteria:
1. The sampling densities, the resolutions of the samples, or equivalently the sizes of the elements need to be uniformly converge to zero;
2. The smallest corner angles in the triangulation are always bounded from below, equivalently the ratio between the circum-circle radius and the inner-circle radius is bounded.

If the above two conditions are satisfied, then the discrete solution converge to the smooth solution, when the meshes are refined. In the following we explain how to generate high quality meshes based on Delaunay triangulation.

Given a compact convex planar domain \( \Omega \), we uniformly sample the boundary \( \partial \Omega \). The boundary sample points are \( P = \{p_0, p_1, \ldots, p_n\} \) such that the distance between \( p_k \) and \( p_{k+1} \) is

\[
\varepsilon < d(p_k, p_{k+1}) < (1 + \lambda) \varepsilon,
\]

where \( \lambda \) is a small positive number.

We compute the Delaunay triangulation \( T \) of \( P \). Suppose \( f_\alpha \) is a triangular face of \( T \), whose circum-radius is greater than \( \varepsilon \), then we compute its circum-center \( q_\alpha \). We insert the Steiner point \( q_\alpha \) into \( P \), and update the Delaunay triangulation \( T \) of augmented \( P \). We repeat this procedure, until the circum-radias of each triangle is no greater than \( \varepsilon \), then we stop.

**Proposition 1.1.** The algorithm terminates after finite steps.

*Proof.* Each time a new Steiner point \( q_\alpha \) is added to \( P \), it must be the center of a circum-circle with radius greater than \( \varepsilon \). This means the distance from \( q_\alpha \) to any point in the preceding Steiner points in \( P \) is greater than \( \varepsilon \). This implies the mutual distances among all the Steiner points in \( P \) are greater than \( \varepsilon \).

Then for each Steiner point \( q_\alpha \), we can draw a disk \( B(q_\alpha, \varepsilon/2) \). By induction, all such disks are disjoint and contained in \( \Omega \). Therefore the maximum number of possible Steiner points equals to
\[
\frac{4|\Omega|}{\pi \varepsilon^2} < \infty.
\]

Hence, the algorithm terminates after finite times of adding Steiner points. \( \square \)

**Proposition 1.2.** In the final Delaunay triangulation, the minimal corner angle is greater than \( \pi/6 \).

**Proof.** Suppose at one step, a Steiner point \( q_\alpha \) is added to \( P \), and the Delaunay triangulation \( \mathcal{T}_\alpha \) is updated to \( \mathcal{T}_{\alpha+1} \). Then it can be shown that all the new edges in \( \mathcal{T}_{\alpha+1} \) are connected to \( q_\alpha \) (as shown in Figure 1.3). Because \( \mathcal{T}_\alpha \) is Delaunay, the circum-circle of \( f_\alpha \) is empty, hence all the new edges are greater than the circum-radius \( \varepsilon \). This shows all the edges lengths in all the intermediate and final triangulations are greater than \( \varepsilon \).

In the final triangulation, for each triangle, the edge lengths are greater than \( \varepsilon \), the circum-radius is no greater than \( \varepsilon \), hence by sine law, the triangle inner angles are greater than \( \sin^{-1} \frac{1}{2} = \pi/6 \). \( \square \)

When the domain is not convex, and more constraints are added, such as preserving feature points, feature lines, the algorithm can be modified to be more sophisticated to meet the requirements, as shown in Figure 1.4 left frame.

![Fig. 1.4 Mesh generation for planar and surface domains.](image)

Surface domain mesh generation can be converted to planar mesh generation using conformal mapping. As shown in Figure 1.4 right frame, a human facial surface is conformally mapped onto a planar domain. Then a high quality mesh is generated on the planar domain, and mapped back to the surface. Because the mapping is angle-preserving, the shape of the triangles are well-preserved. This guarantees the quality of the generated mesh. From the figure, we can see all the small triangles on the surfaces are close to be equilateral.