Chapter 7

Delaunay-based mesh generation methods

Introduction

Delaunay triangulation and the related construction methods resulting in this triangulation have been extensive fields of research for a very long time. In particular, these topics are one of the major concerns in computational geometry. It is therefore not really surprising to find a great deal of literature about Delaunay triangulation, starting with the pioneering paper by Delaunay himself, [Delaunay-1934]. Relevant references include [Shamos,Preparata-1985], [Joe-1991], [Fortune-1992], [Rajan-1994], [Boissonnat,Yvinec-1995], [Ruppert-1995] among many others. Delaunay triangulation problems are of interest for a number of reasons. Firstly, numerous theoretical issues can be investigated. Then, a wide range of applications in various disciplines exists, including geometry, computational geometry, etc., as well as computer science and many engineering problems.

In the scope of this book, Delaunay triangulation problems are of great interest as they can serve to support efficient and flexible mesh generation methods. In this respect, people concerned with engineering applications have investigated Delaunay based mesh generation methods. The main references for this topic include [Lawson-1977], [Hermeline-1980], [Watson-1981], [Bowyer-1981] in the early 80s and many others in the next decade such as [Weatherill-1985], [Mavriplis-1990], etc. An entire book, [George,Borouchaki-1997], has been written to provide a thorough discussion of Delaunay triangulation and meshing for engineering purposes. Therefore, the aim of this chapter is simply to outline the main ideas related to Delaunay type methods without claiming exhaustivity, and we consider this approach to the same extent as the two previous approaches. The purpose is to provide an understanding of the method and, moreover, to discuss it from the same standpoint as the other possible approaches (using a quadtree/octree method or an advancing-front strategy).

★
★ ★

This chapter is divided into six parts. The first part introduces some theoretical issues regarding Delaunay triangulation. The second part discusses the notion of a constrained triangulation. Then, we show how to take advantage of the previous material so as to develop a Delaunay-type mesh generation method. The fourth part briefly introduces several variants of this technique suitable for mesh generation. Finally, extensions are proposed to cope with more complex mesh generation problems. In particular, we explain how to complete a mesh conforming to a pre-specified size (or density) and how to generate anisotropic meshes. This result will be used when designing a parametric surface mesh generation method (Chapter 15).

7.1 The Delaunay triangulation

The Delaunay triangulation can be introduced in various manners (depending on the context of application). Among them, a convenient way is to use the dual of this triangulation, the Voronoï diagram.

7.1.1 The Voronoï diagram

Let $\mathcal{S}$ be a finite set of points $(P_i)_{i=1,\dots,n}$ in d dimensions. The Voronoï diagram for $\mathcal{S}$ is the set of cells, V_i, defined as :

$$V_i = \{P \quad \text{such that} \quad d(P,P_i) \leq d(P,P_j), \quad \forall j \neq i\} \tag{7.1}$$

where $d(.,.)$ denotes the usual Euclidean distance between two points. A cell V_i is then the set of the points closer to P_i than any other point in $\mathcal{S}$. The V_i's are closed (bounded or not) convex polygons (polyhedra in three dimensions, d-polytopes in d dimensions); these non-overlapping cells tile the space, and constitute the so-called Voronoï diagram associated with the set of points $\mathcal{S}$ in $\mathbb{R}^d$.

7.1.2 Delaunay triangulation and Voronoï diagram

A triangulation problem typically concerns the construction of a triangulation of the convex hull of the P_i's such that the P_i's are element vertices. The construction of the Delaunay triangulation of this convex hull can be achieved by considering that this triangulation is the dual of the Voronoï diagram associated with $\mathcal{S}$.

Based on Definition (7.1), each cell V_i of the Voronoï diagram is a non-empty set and is associated with one point in $\mathcal{S}$. From these V_i's, the dual can be constructed, which is the desired Delaunay triangulation. For instance, in two dimensions, the cell sides are midway between the two points they separate, thus, they are segments that lie on the perpendicular bisectors of the edges of the triangulation. In other words, joining the vertices in $\mathcal{S}$ belonging to two adjacent cells results in the desired triangulation. The latter is unique and consists of simplices (triangles or tetrahedra

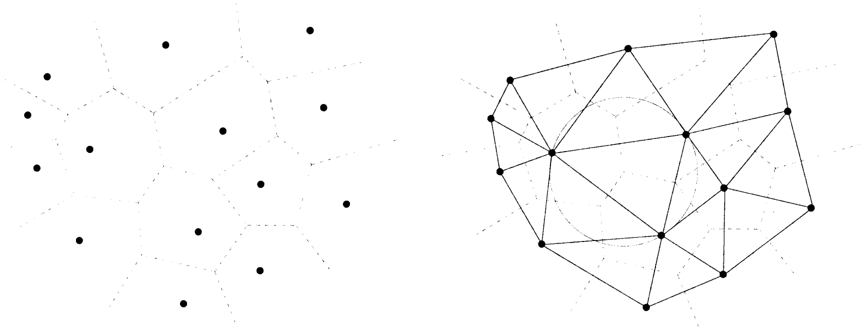

Figure 7.1: *Left-hand side : Voronoï diagram (in two dimensions). Right-hand side : corresponding Delaunay triangulation.*

according to d) provided the points in $\mathcal{S}$ are in general position (cf. Chapter 1). Otherwise, elements other than simplices can be constructed which can be easily split into simplices (thus resulting in several solutions for a unique set of points).

7.1.3 Theoretical issues

This section recalls some classical theoretical issues about the Delaunay triangulation. In this respect, a fundamental theorem, the so-called *lemme général de Delaunay*, will be given. But first, we need to provide the definition of the well-know *empty sphere criterion*.

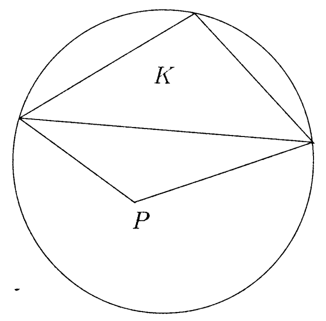

Figure 7.2: *In this two-dimensional example, the empty sphere criterion is violated as the disc of triangle K encloses a point P (the closed ball of K contains a point other than the vertices of K). Note that this example is special as the point P is the vertex of a triangle adjacent to K which is opposite the common edge.*

The empty sphere criterion. In two dimensions, this definition refers to the open disc circumscribing a triangle, in three dimensions it concerns the open sphere (ball) circumscribing a tetrahedron. This criterion is also referred to as the *Delaunay criterion.*

The *lemme général de Delaunay* can be enounced as follows[1] :

General lemma. *Let $\mathcal{T}$ be a given arbitrary triangulation of the convex hull of a set of points $\mathcal{S}$. If for each and every pair of adjacent simplices in $\mathcal{T}$, the empty sphere criterion holds, then this criterion holds globally and $\mathcal{T}$ is a Delaunay triangulation.*

Remark 7.1 *This lemma is a rather simple characterization of the Delaunay triangulation. Note that a local property about the Delaunay criterion for two adjacent elements results in a global property about the entire triangulation.*

The proof of this lemma can be achieved in several ways and can be found in numerous references. For the sake of simplicity, we assume, in the following sections, that the given points are in general position. With this background, we discuss one method (among many others) that allows the construction of the Delaunay triangulation of the convex hull of a given set of points.

Incremental method. Given $\mathcal{T}_i$, the Delaunay triangulation of the convex hull of the first i points in $\mathcal{S}$, we consider P the $(i+1)^{th}$ point of this set.

The purpose of the incremental method is to construct $\mathcal{T}_{i+1}$, the Delaunay triangulation including P as an element vertex, from $\mathcal{T}_i$. To this end, we introduce a procedure, the so-called *Delaunay kernel* which can be simply written as :

$$\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{C}_P + \mathcal{B}_P\,, \tag{7.2}$$

where $\mathcal{C}_P$ is the *cavity* and $\mathcal{B}_P$ is the *ball* associated with point P. Without loss of generality, we assume that P is included[2] in $\mathcal{T}_i$, then :

- the cavity $\mathcal{C}_P$ is the set (the union) of elements in $\mathcal{T}_i$ whose circumballs contain point P and
- the ball $\mathcal{B}_P$ is the set of elements formed by joining P with the external faces of the above cavity.

From a practical point of view, the directly usable decisive result is that the cavity is a star-shaped set with respect to point P.

[1]Published in French in 1934, the original lemma is, *in extenso*, as follows :
" Soient T des tétraèdres tout à fait arbitraires qui partagent uniformément l'espace à n dimensions étant contigus par des faces entières à n-1 dimensions et tels qu'un domaine quelconque limité (c'est-à-dire à diamètre limité) ait des points communs seulement avec un nombre limité de ces tétraèdres, alors la condition nécessaire et suffisante pour qu'aucune sphère circonscrite à un tel tétraèdre ne contienne dans son intérieur aucun sommet d'aucun de ces tétraèdres est que cela ait lieu pour chaque paire de deux de ces tétraèdres contigus par une face à n-1 dimensions, c'est-à-dire que dans chaque telle paire le sommet d'un de ces tétraèdres ne soit intérieur à la sphère circonscrite à l'autre, et réciproquement."
This text can be found in [Delaunay-1934], page 793.

[2]In fact, three situations are possible including this case. The other cases are when P is outside all elements although it belongs to a circumscribing ball and the case where P is outside all elements and balls. In such cases, the definition of the cavity is slightly different but the same construction scheme holds.

Theorem 7.1 *Let $\mathcal{T}_i$ be a Delaunay triangulation and let P be a point enclosed in $\mathcal{T}_i$. The above construction scheme completes $\mathcal{T}_{i+1}$, a Delaunay triangulation including P as an element vertex.*

There are several ways to prove this theorem. Here, we give two different proofs based on what is assumed and what must be proved (another proof, based on the Voronoï duality, can be found in [George,Borouchaki-1997]).

Proof 7.1.a. shows that $\mathcal{T}_{i+1}$ is a Delaunay triangulation and establishes that $\mathcal{B}_P$ conforms to the previous definition. Proof 7.1.b. shows that Relationship (7.2) where $\mathcal{B}_P$ is defined as above results in a Delaunay triangulation since $\mathcal{T}_i$ is Delaunay.

Proof (7.1.a.) This proof is completed in two parts. First, $\mathcal{T}_{i+1}$, a Delaunay triangulation, exists as the dual of the corresponding Voronoï diagram. Moreover, for the same reason, $\mathcal{T}_{i+1}$ is unique. Thus, the only thing we have to show is that $\mathcal{T}_{i+1}$ is the triangulation as defined by Relationship (7.2), meaning that $\mathcal{C}_P$ and $\mathcal{B}_P$ are exactly the previously introduced sets.

As the elements that violate the Delaunay criterion are those of $\mathcal{C}_P$ (and only those), the remaining part of $\mathcal{T}_i$ remains unchanged and this part becomes a part of triangulation $\mathcal{T}_{i+1}$. Then, we just have to establish that $\mathcal{B}_P$ is the appropriate construction to replace $\mathcal{C}_P$.

Let $\mathcal{R}_P$ be the re-triangulated cavity in $\mathcal{T}_{i+1}$. We will show that $\mathcal{R}_P$ is the above $\mathcal{B}_P$. The set $\mathcal{R}_P$ is a set of elements having P as a vertex. This is proved by contradiction. To this end, let assume that there exists one element in $\mathcal{R}_P$ without P as one of its vertex, then this element is necessarily a member of $\mathcal{C}_P$ and thus violates the Delaunay criterion. Therefore, all elements in $\mathcal{R}_P$ have P as a vertex. As a consequence, they can be written as (P,f) where f is a face. Assume that f is not an external face of $\mathcal{C}_P$, then there exists an element in $\mathcal{R}_P$ that shares the face f with the element (P,f). In other words, this element can be written as (Q,f), where Q is different from P. This leads to a contradiction, hence f is necessarily an external face of $\mathcal{C}_P$. This enables us to conclude.

The solution exists, is unique and $\mathcal{R}_P = \mathcal{B}_P$ is a valid way to replace the cavity. This particular solution is then the desired solution. □

We now turn to a different proof. Before giving it however, we recall a fundamental lemma.

Lemma 7.1 *The Delaunay criterion for a pair of adjacent elements is symmetric.*

We consider a pair of adjacent simplices sharing a face and P (respectively Q) the vertex in these simplices opposite that face. Then,

$$Q \notin B_P \Leftrightarrow P \notin B_Q \tag{7.3}$$

where B_P (resp. B_Q) denotes the ball associated with the simplex having P (resp. Q) as a vertex.

Proof (7.1.b). In this discussion, we do not infer the duality between the Voronoï diagram and the Delaunay triangulation (so as to prove the existence of a solution). Consider $\mathcal{T}_i$ a Delaunay triangulation and a point P contained in some elements but not a vertex element. We would like to show that the above construction computes $\mathcal{T}_{i+1}$ a Delaunay triangulation with P as an element vertex.

At first, P is an element vertex of $\mathcal{T}_{i+1}$ according to the definition of $\mathcal{B}_P$. Then, we just have to establish that $\mathcal{T}_{i+1}$ is a valid Delaunay triangulation.

We first establish that the triangulation is valid (regarding the topology) as $\mathcal{C}_P$ is a connected set of elements. Assume that $\mathcal{C}_P$ consists of two connected components, one of these including an element, denoted by K_0, which separates this connected component from that enclosing P. Then define the segment joining the centroid of this element to P. This segment intersects one face of K_0, thus consider the element, say K_1, sharing this face with K_0. By definition, before introducing point P, the pair K_0 and K_1 complies with the Delaunay criterion (as members of $\mathcal{T}_i$). Hence the vertex of K_1 opposite the common face is outside the circumball of K_0. As a consequence, the circumball of K_1 necessarily encloses P. As a result, K_1 is a member of $\mathcal{C}_P$. Repeating the same discussion, it is shown that all elements between the two connected components of the cavity are in fact members of this set. Hence the cavity is a connected set. The triangulation of $\mathcal{B}_P$ is then valid, in terms of its connections.

Moreover, since the external faces of $\mathcal{C}_P$ are visible by P, this triangulation is valid (regarding the geometry). The reason is obvious in two dimensions. We proceed by adjacency from triangle K_0, the triangle within which point P falls. Then, the three edges of K_0 are visible from P. Let K_1 a triangle sharing an edge f_1 with K_0. Then if K_1 violates the Delaunay criterion, it is in the cavity and the faces of K_1 other than f_1 are visible by P. This is due to the fact that P is inside the circumball of K_1 and that f_1 separates P and the vertex of K_1 opposite f_1. Thus, applying the same discussion allows the result for all the triangles in the cavity. However, the same argument does not extend in three dimensions as a face does not have the required separation property. Indeed, following the same construction from K_0, it is possible to meet as tetrahedron K_1 an element whose faces other than f_1, the face common with K_0, include one face, g, not visible by P. In this case, K_2, the element sharing the face g with K_1 is necessarily in the cavity, thus leading to the desired property.

Exercise 7.1 *Given the above situation, prove that K_2 is a member of the cavity of point P. Hint : examine the region within which P falls.*

To complete the proof, we have to show that $\mathcal{T}_{i+1}$ is a Delaunay triangulation. To this end we use the above general lemma. Then, the only thing which must be established is that the empty sphere criterion holds for all and every pair of adjacent elements. To account for all possible configurations of such pairs, the elements in $\mathcal{T}_{i+1}$ are classified into three categories :

i) those in $\mathcal{B}_P$,

ii) those having one element outside $\mathcal{B}_P$ and sharing an external face of $\mathcal{C}_P$,

iii) the remaining pairs.

Obviously, the elements falling in the third category conform to the Delaunay criterion. Those of category *ii*) are Delaunay too. Actually, while their circumballs do not enclose P, all the vertices opposite the face shared with the cavity are not inside the circumballs associated with the elements in $\mathcal{B}_P$. This is due to the symmetric property of the Delaunay criterion. Those of category *i*) are also Delaunay. The proof is again obtained by contradiction. We consider an external face of the cavity, say f, and we consider the element of this cavity having this face (a former element of $\mathcal{T}_i$). Let K_{old} be this element, together with the new element constructed with this face, K_{new}. Then, assume that the circumball of K_{new} includes a vertex, this vertex is necessarily outside the circumball of K_{old} and is therefore outside the cavity. While the elements outside the cavity are Delaunay, this results in a contradiction. Thus, the entire proof is completed. □

7.1.4 Practical issues

In this section, we briefly consider some practical issues that can be derived from the above theoretical background.

First, the incremental method can be used to define a constructive triangulation method even in the case where the given points are not in general position (*i.e.*, when four or more co-circular, five or more co-spherical points are in the initial set, which is usually a plausible situation for realistic engineering applications).

Then, replacing the problem of constructing a triangulation of the convex hull of the given set of points by that of triangulating a convex "box" enclosing all the initial points, we can compute a solution using the same incremental method. Indeed, the box defines a convex hull problem for set $\mathcal{S}$ enriched with the corners of this box. As a consequence, all vertices fall within this box.

In the previous discussion, we did not account for numerical problems that can arise such as those related to round-off errors. As the key to the method is the proper definition of the cavity, any wrong decision when determining whether an element is in this set may lead to an invalid cavity. In other words, due to round-off errors, the above construction may fail, thus resulting in an unsuitable triangulation. Hence, at the cavity construction step, a correction is applied to ensure the star-shapedness of the cavity, see [George,Hermeline-1992]. Basically, this means that we explicitly check the visibility property (which is equivalent to the starshapedness property) and, in case of a failure, we modify the cavity accordingly.

Remark 7.2 *In the case where such a correction is applied, the resulting triangulation could be non Delaunay.*

7.2 Constrained triangulation

As pointed out in Chapter 1, a constrained triangulation problem concerns a triangulation problem of a set of points in the case where some constrained entities (edges or faces) are specified that must be present in the resulting triangulation.

In this section, we discuss three aspects related to a constrained triangulation. We show how to maintain such an entity (edge or face) when it exists at some step. Then we turn to a method suitable for entity enforcement in two and three dimensions.

7.2.1 Maintaining a constrained entity

A triangulation procedure such as the incremental method (Relationship (7.2)) can be constrained so as to preserve a specified edge (a face in three dimensions) which is introduced at some step of the incremental scheme and must be retained. To this end, the construction of the cavity is modified.

When (see above) visiting the elements by adjacency, we do not pass through the specified edges (faces in three dimensions) which have been created at some previous step. This means that two elements are not regarded as adjacent if they share a specified entity (edge or face) created in the triangulation at a previous step. Hence the cavity construction is modified in this way. This results in a triangulation where some edges (faces) are specified. Due to this constraint, this triangulation is no longer a Delaunay triangulation (it is referred to as a constrained Delaunay triangulation for which the Delaunay criterion is not required between a pair of elements separated by a constrained item).

Theorem 7.2 *Let $\mathcal{T}_i$ be an arbitrary triangulation and let P be a point enclosed in $\mathcal{T}_i$, then Relationship (7.2) determines $\mathcal{T}_{i+1}$, a valid triangulation having P as element vertex.*

This theorem just means that Relationship (7.2) (considered together with a correction algorithm in some cases, see the above discussion) still results in a valid triangulation even when the initial triangulation is an arbitrary triangulation and some constraints are present.

Proof. This proof is obvious since the cavity involved in the construction is a star-shaped region due to the way in which it is constructed (starting from the base and completed by adjacency while, at the same time, the required visibility property is explicitly achieved by a correction stage). Then, the definition of the ball is valid and the resulting triangulation is valid as well. □

The previous construction is not, in general, a solution to ensure the existence of a pre-specified set of edges (edges and faces in three dimensions) in a triangulation at the time the endpoints of these items have been inserted in the triangulation. Thus, other methods must be developed which work well in this case.

Remark 7.3 *In three dimensions, the above discussion holds for a constrained face but is not a solution for a constrained edge. We can remove an edge by turning around it by means of face adjacencies.*

7.2.2 Constraints in two dimensions

We consider a series of edges whose endpoints are in set $\mathcal{S}$ and we want to make sure that these items are edges of the triangulation at the time all the points in $\mathcal{S}$ have been inserted. In general, this property does not hold as can be seen in Figure 7.3 where a simple example is depicted.

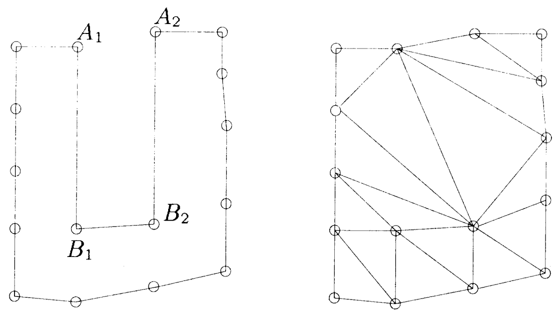

Figure 7.3: *In this simple two-dimensional example, we have displayed the triangles which are intersected or close to two missing edges (in particular, some triangles, part of the triangulation of the convex hull, are not shown). Actually, edges A_1B_1 and A_2B_2 are missing in the triangulation although their endpoints are element vertices.*

The problem we face is then to enforce[3] the missing edges. In two dimensions, a rather simple procedure can be used to get this result, the so-called *diagonal swapping* (see also Chapter 18).

Given a pair of adjacent triangles sharing an edge, we consider the quadrilateral formed by these two triangles. If this polygon is convex then it is possible to swap its diagonal (the former common edge) so as to create the alternate diagonal. In this way, we have removed an edge (while a new one is created). A repeated use of this procedure enables us to suppress all the edges that intersect a specified segment (actually, an edge that must be constructed) and results in the desired solution.

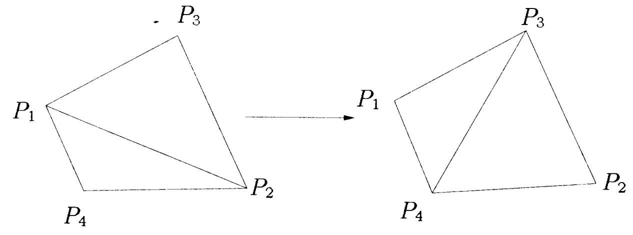

Figure 7.4: *Diagonal swapping. The quadrilateral formed by this pair of adjacent triangles is a convex region, so its diagonal can be swapped.*

[3]This is an *a posteriori* approach to the constrained triangulation problem. Note that an *a priori* solution can be also envisaged, as will be discussed hereafter and in Chapter 9.

Applied to each missing entity, this procedure computes the desired triangulation which, however, is obviously not a Delaunay triangulation.

Two theoretical issues can be invoqued to justify the above method.

Theorem 7.3 *Given a set of segments, there exists a triangulation incorporating these segments as element edges.*

Theorem 7.4 *Given an arbitrary triangulation and a set of segments (whose endpoints are vertices of this triangulation), a triangulation where these entities are edges can be computed using only the diagonal swapping operator.*

In fact, given an arbitrary triangulation, it is always possible to obtain a specified triangulation having the same vertices, by means of diagonal swapping only. Thus, a non cyclic process of diagonal swapping is a solution to the above problem (note that the diagonal swapping is a reversible process). Note that the previous theorems hold in two dimensions only.

7.2.3 Constraints in three dimensions

The same problem is much more difficult in three dimensions. Actually, the constrained entities could be a series of edges and a series of faces (assumed to be triangular) and, at the time all the endpoints of these entities have been inserted, some of these edges and faces might not be present in the triangulation.

The problem is split into two parts. At first we enforce[4] the missing edges and, once this has been completed, we enforce the missing faces (indeed, there exist geometrical configurations where the three edges of a face exist while the face itself is not formed meaning that one or several edges of the current triangulation pass through the triangle whose edges are the three above edges).

Theoretical results can be put forward to prove that an edge can be enforced in a triangulation by means of generalized swapping and, if necessary, by creating some points, the *Steiner points*, to overcome the situations where no more swaps can be successfully done. This result can be seen as an extension of Theorem (7.4) to three dimensions. Regarding the constraint by a face, the situation is not so clear. In practice, heuristics must be used.

Generalized swapping procedure. It is appealing to extend the two-dimensional diagonal swapping by considering the pattern formed by a pair of adjacent tetrahedra. This leads to a face swapping procedure whose converse application results in removing an edge. In fact, the latter operator is only a simple occurrence of a more general operator dealing with a general pattern, the so-called *shell* (see Chapter 2). A shell is the set of tetrahedra sharing a given edge. Then, the three-dimensional version of the swap operator can be seen as remeshing this polyhedron by suppressing the common edge (see Chapter 18).

[4]See the above footnote about *a posteriori* or *a priori* solutions for the problem.

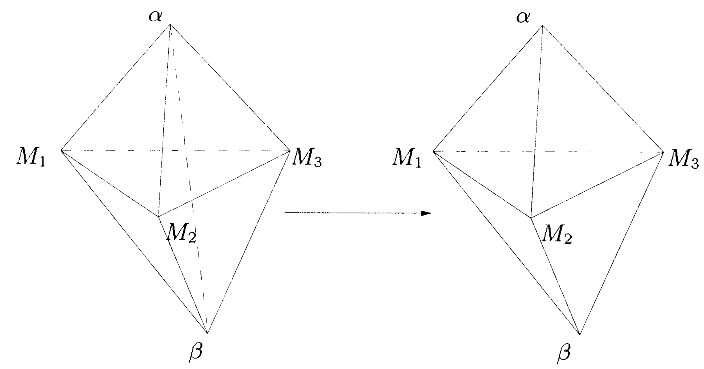

Figure 7.5: *The polyhedron consisting of the two tetrahedra $M_1M_2M_3\alpha$ and $\beta M_1M_2M_3$ (right side of the figure) is convex and can be re-meshed using the three tetrahedra $M_1\alpha\beta M_2$, $M_2\alpha\beta M_3$ and $M_3\alpha\beta M_1$. Conversely, this three element configuration can be re-meshed by means of two elements. In the first transformation, a face has been removed while in the second an edge has been removed.*

About Steiner points. A rather obvious example (Figure 7.7) shows that it is not always possible to triangulate a region. Nevertheless, adding a single point in the region depicted at the bottom of the figure leads to a solution. As a result, one could expect to meet such situations when considering a constraint in a triangulation and would like to use a similar method to obtain a valid solution. The question is then how to detect such a pathology and how many points are strictly needed to overcome the difficulty and where this (these) point(s) must be located. Actually, this leads to finding the *visibility kernel* of a given polyhedron.

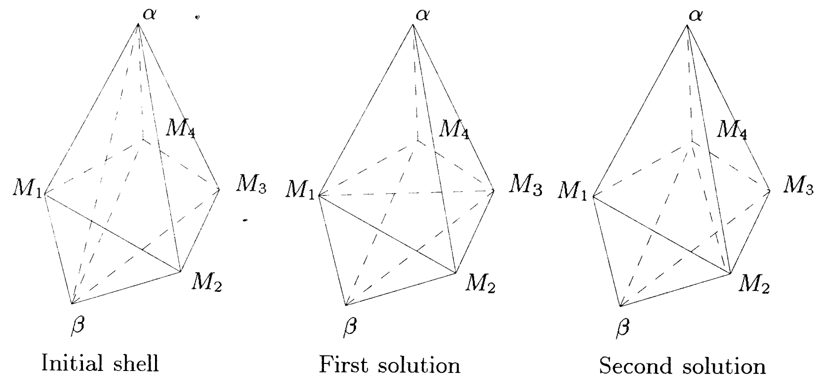

Figure 7.6: *The initial pattern is the shell associated with the edge $\alpha\beta$. Two alternate remeshings of this polyhedra are possible if it is convex.*

Following the previous discussion, we propose a heuristic method to enforce a set of constraints. First, we deal with the problem of edge enforcement, then we turn to the face enforcement problem. The key idea is to locally modify the current triangulation by means of the generalized swapping operator, creating some Steiner points when the previous operator fails.

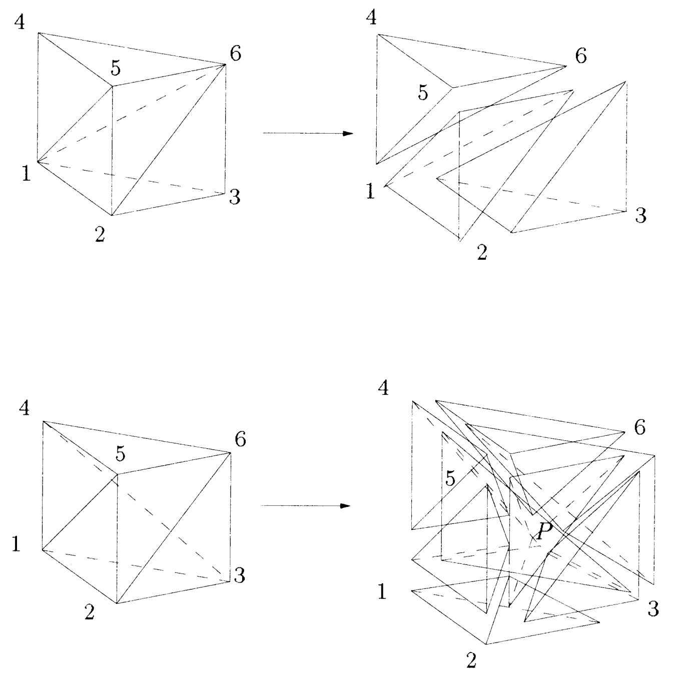

Figure 7.7: *The prism at top can be partitioned by simply using three tetrahedra. The prism at bottom, the so-called Schönhardt polyhedron, cannot be split with three tetrahedra. This prism differs from the previous one in the way in which its quadrilateral faces are decomposed into triangular faces. To find a valid mesh, a point must be created in the visibility kernel of this polyhedron. This point is then joined with all the external faces, thus resulting in a suitable mesh.*

Edge enforcement. The elements in the current triangulation that are intersected by a missing edge are identified (such a set is called a *pipe*). Then we meet two situations. Either only faces of these elements are intersected by the missing edge or the latter intersects, at least, one edge of the current triangulation. The first case leads to applying the generalized swapping to every pair of adjacent tetrahedra (in the case where the so-formed region is convex) while the second situation leads to modifying the shell(s) of interest. Steiner points are created when no more swaps can be successfully completed.

Except for the numerical problems, the above idea makes it is possible to regenerate all the missing edges. It is now possible to consider the missing faces (if any, since most of them exist at the time their edges are present in the mesh).

Face enforcement. A similar procedure is used. The set of elements corresponding to a missing face are exhibited. In this set, a series of edges exists which intersect the missing face. These edges are then swapped, using Steiner points in some cases, until a missing face is intersected by only one edge. Then an ultimate generalized swap results in the desired solution (assuming that the corresponding pattern is convex).

This heuristic, while not theoretically proved (actually, the problem is probably a NP-problem), has proved to work well in most situations.

7.3 Classical Delaunay meshing

Delaunay triangulation algorithms serve as a basis for designing a meshing method, a so-called Delaunay type mesh generation method.

In practice, the problem we face is now rather different. Up to now, we have discussed a triangulation problem. This means a problem of triangulating the convex hull of a given set of points. But, for a typical meshing problem, the input is a closed polygonal curve (polyhedral surface) defining a region (or several such curves (surfaces) defining a multiply connected region). The problem is then to generate a set of vertices in this not necessarily convex domain and to make sure that the above (curve or surface) discretization is present in the resulting triangulation. This means that we meet a problem of constrained triangulation as a series of edges and faces must be present in the mesh.

Despite these differences, some of the previous material on Delaunay triangulation, possibly with some extensions, can be applied to meshing problems when the domain is supplied via a boundary discretization.

The intention that the mesh be suitable for applications takes several forms. For classical Delaunay meshing, as discussed in this section, it means that the mesh elements should be well-shaped while their sizes should be adequate with regard to the sizing information available in this case (basically, the sizes of the boundary items serving as input data). However, applications typically require meshes in which the element sizes, and even their shapes, vary across the mesh according to a given specification. This point will be discussed in further sections.

A Delaunay type meshing method is generally one step of a mesh generation procedure including three successive steps :

Step 1 : the mesh parameterization (boundary description, specification or construction of a function defining the element size distribution, ...),

Step 2 : the boundary discretization,

Step 3 : the creation of the field vertices and elements, in other words, the Delaunay type method itself.

This general scheme is close to that found in other methods such as the advancing-front type method (Chapter 6) and is slightly different to that of a method based on an *quadtree* (Chapter 5) where the boundary mesh can be constructed during the domain mesh construction.

In a Delaunay type method (Step 3 of the above scheme) the resulting mesh is a mesh of the box enclosing the domain. Field points are created and inserted in the current mesh so as to form the elements by means of the Delaunay kernel.

7.3.1 General scheme

The previous material can be now used to develop a Delaunay type mesh generation method. Here is a scheme for such a method (suitable in both two and three dimensions) :

- Preparation step.
  - Data input: point coordinates, boundary entities and internal entities (if any),
  - Construction of a bounding box and meshing of this box by means of a few elements.
- Construction of the box mesh.
  - Insertion of the given points in the box mesh using the Delaunay kernel.
- Construction of the empty mesh (which is boundary conforming).
  - Search for the missing specified entities,
  - Enforcement of these items,
  - Definition of the connected components of the domain.
- Internal point creation and point insertion.
  - (1) Internal edges analysis, point creation along these edges.
  - Point insertion via the Delaunay kernel and return to (1) until edge saturation.
- Domain definition.
  - Removal of the elements exterior to the domain,
  - Classification of the elements with respect to the connected components.
- Optimization.

Note that the domain definition is only achieved at the end of the process. In this way, the convex mesh of the box is present throughout the process which facilitates the necessary searching operations (for instance, when finding a base so as to initialize the cavity).

In the following sections, we describe the different stages of this general scheme and we focus on the main difficulties expected.

Preliminary requirements. In contrast to advancing-front methods (Chapter 6), no specific assumption is made on the nature of the input data related to the boundary discretization.

In particular, the orientation of the boundary items is not required and does not offer any specific interest (whereas it may increase the processing speed in quadtree-octree methods).

7.3.2 Simplified Delaunay type triangulation method

Without loss of generality, we define a convex "box", large enough to enclose the domain. In this way we again encounter a situation where the previously described incremental method can be used.

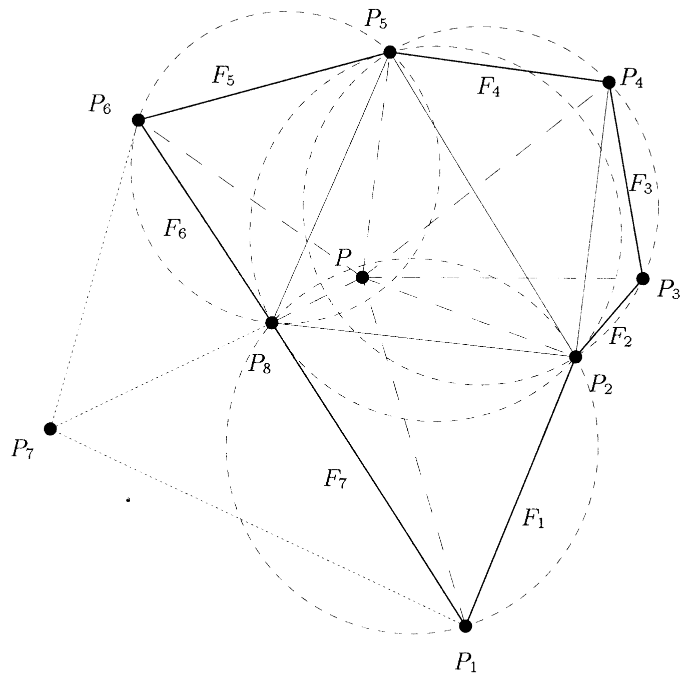

Figure 7.8: *Inserting point P (P included in the current mesh). In this two-dimensional example, only the triangles close to point P are displayed. The base is reduced to the triangle $(P_2P_5P_8)$. The cavity is formed by the triangles in solid lines. The external faces of this cavity are denoted by $F_1, F_2, ..., F_7$. The ball consists of the elements in dotted lines. It is formed by joining P with the F_i's.*

In fact, introducing a box, enables us to return to a convex hull problem where the set $\mathcal{S}$ consists of the vertices of the given boundary discretization and four (eight) additional points (the corners of the introduced box, a square in two dimensions, a cube in three dimensions).

Once the box has been triangulated by means of two triangles (five or six tetrahedra), we find a situation where all the points in $\mathcal{S}$ (apart from the box corners) are strictly included in this initial triangulation. Due to this simple property, which will be maintained throughout the meshing process, the construction method reduces to the case where the points that must be inserted are always inside the current triangulation.

Thus the construction method relies on properly defining the cavity associated with the point to be inserted knowing that this point necessarily falls within a mesh element. This construction, for a given point P, is sequentially achieved as follows :

1. we search in the current mesh for the element within which point P falls. As a result we obtain a set of elements, the so-called *base* associated with P. This base can be reduced to one element, two elements when P is located on an edge (a face in three dimensions) or more when, in three dimensions, P falls on one edge.
2. starting with the elements in the base, we visit by adjacency the current mesh so as to determine the elements whose circumballs contain point P. This results (possibly after a correction stage) in the desired cavity.

Then, this cavity is replaced by the corresponding ball and the mesh with P as vertex is completed.

This simple procedure is applied to all the points known at this stage (typically, the boundary points). At completion, we have created a mesh of the box enclosing the domain and **not** a mesh of this domain.

7.3.3 Boundary integrity

The mesh resulting from the above method is a mesh of the box enclosing the domain. As previously seen, the boundary entities defining the domain, are not necessarily present in this box mesh. In other words, we face a constrained meshing problem. Typically, two approaches can be envisaged to solve this problem, one being an *a priori* approach and the other an *a posteriori* approach. In the first case, the boundary discretization is such that it naturally appears in the mesh, in the second case, some boundary entities are missing in the current mesh which need to be enforced.

Delaunay-conforming boundary mesh. Before constructing the box mesh, we analyze the boundary discretization to see whether it is Delaunay or not. At this time, this notion simply means that the boundary entities are automatically present in the mesh based on their endpoints. If the given discretization is not Delaunay, then we modify it (see Chapter 9) so as to meet this property. Hence, boundary integrity is no longer a problem.

Boundary enforcement. We are given a mesh where some edges (faces) are missing. In this approach, we return to the method discussed for the constrained triangulation, and, by means of local mesh modifications, we modify the current mesh in such a way as to ensure the existence of all boundary entities and to obtain the desired boundary integrity.

7.3.4 Identifying the elements in the domain

When the boundary entities of a given domain are present in the mesh, it is possible to identify the elements of the mesh which lie within this domain. Bear in mind that we have triangulated a "box" enclosing the domain and that we now need to discover this domain.

A rather simple algorithm, based on coloring (see Chapter 2), can be used to discover the connected component(s) of the domain. In this way the internal elements can be determined where, furthermore, it will be possible to create the field points. The scheme of this algorithm is as follows :

1. Assign the value $v = -1$ to all elements of the box mesh (where the boundary entities now exist) and set $c = 0$, c being seen as a color :
2. Find an element having as a vertex one of the box corners and set it to the value $v = c$, put this element in a list;
3. Visit the three (four) elements adjacent by an edge (a face in three dimensions) to the elements in the list :
   - if the color of the visited element is not -1, the element has been already colored, thus return to 3;
   - else if the face (edge) common with the visited element and the current element in the list is not a boundary entity, assign the value $v = c$ to this element, put it in the list and return to 3;
   - else if the common face (edge) is a boundary member, return to 3;
4. Set $c = c + 1$, empty the list and if an element with $v = -1$ exists, put it in the list and return to 3. □

Variants of this algorithm can be used to complete the same task. Nevertheless, at completion of such a procedure the elements are classified according to the different connected components of the domain.

7.3.5 Field point creation

We now have a mesh for the domain where the element vertices are typically the boundary points, meaning that no (or few[5], in three dimensions) internal vertices exist in the mesh. Thus, to fulfill the numerical requirements (well-shaped elements and adequate sized elements), we have to create some field points in the domain.

Several methods can be envisaged to achieve this; among these, we focus here on one method using the edges of the current mesh as a spatial support for the field points.

Preliminary requirements. At first, a stepsize h is associated with all the boundary vertices (by means of the average of the lengths (surface areas) of the edges (faces) sharing a boundary vertex).

[5]The necessary Steiner points.

Edge analysis. The key idea is to consider the current mesh edges and to construct a set of points on them. The process is then repeated as long as the creation of a point on an edge is needed. In other words, as long as the edges are not *saturated*. This iterative process starts from the mesh obtained after the domain definition (see above), constructs a first series of points, inserts them into the current mesh and repeats the processing on the resulting mesh.

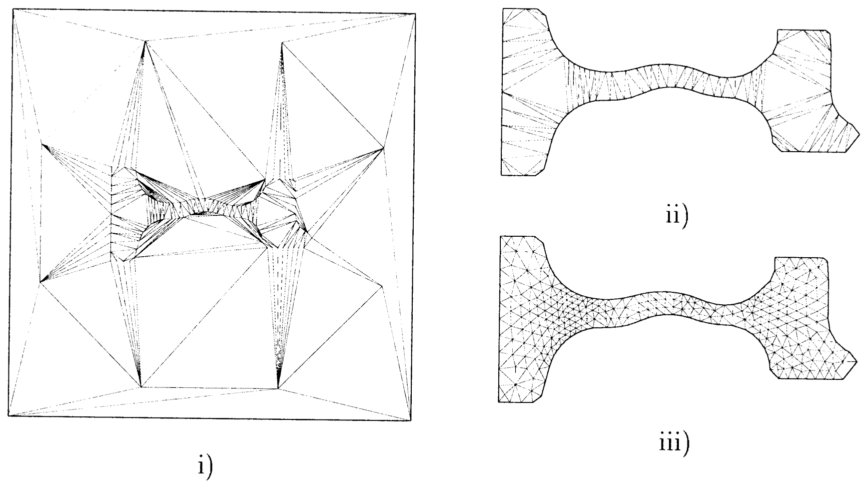

Figure 7.9: *i) Mesh of the box enclosing a domain for a mesh generation problem in two dimensions. One can see the four corners used to define the box enclosing the domain together with some extra points defined between the box and the domain for efficiency reasons. ii) Corresponding empty mesh. This mesh is nothing other than a coarse discretization of the domain resulting from the previous coloring algorithm. Actually, this mesh displays the edges where a first wave of field points will be created. iii) Final mesh after internal point insertion and optimization.*

Then, the current mesh edges are examined and their lengths are compared with the stepsizes related to their endpoints. The goal of the method is to decide if one or several points must be created along the visited edge. If so, both the number of required points, n, and their location must be determined.

The objective is twofold. We want to introduce suitably spaced points along the edges in order to saturate them and to obtain a smooth point distribution.

We demonstrate an arithmetic type of point distribution for an edge AB. If

- $h(0) = h_A$ denotes the stepsize associated with $P_0 = A$, one of the endpoints,
- $h(n+1) = h_B$ is that related to $P_{n+1} = B$, the other endpoint,

one can define a sequence α_i (and thus the corresponding P_i's) as :

$$\left\{ \begin{array}{lcl} \alpha_0 & = & h(0) + r \\ \alpha_n & = & h(n+1) - r \\ \alpha_i & = & d(P_i, P_{i+1}) \end{array} \right. \tag{7.4}$$

where $d(P_i, P_{i+1})$ is the (Euclidean) distance between P_i and P_{i+1}, while r is the ratio of the distribution. This requires us to solve the system :

$$\left\{ \begin{array}{lcl} \sum\limits_{i=0}^{n} \alpha_i & = & d \\ \alpha_{i+1} & = & \alpha_i + r \end{array} \right. \tag{7.5}$$

to find both r and n. The solutions are :

$$n = \frac{2d}{h(0) + h(n+1)} - 1 \quad \text{and} \quad r = \frac{h(n+1) - h(0)}{n+2} .$$

As n must be an integer value, the solution is rescaled so as to obtain an exact discretization of the edge AB in terms of n and r. The α_i's and thus the sequence of points is determined at the time n and r are established. Then, with each thus-defined point is associated a value, h, derived from the h's of the supporting edge. This means that the control space[6] is completed on the fly.

The process is repeated for all the current mesh edges and the series of points created in this way is then filtered, using simply a (structured) grid (cf. Chapter 1). This treatment is related to the fact that the vertices are well-positioned along one edge but this property does not hold globally. For instance, one may observe the case of all the edges emanating from one point. The retained points are then inserted using the Delaunay kernel (Relationship (7.2)) and the entire process is iterated as long as some mesh edges need to be subdivided, *i.e.*, are still not saturated.

It could be noted that this rather simple method is independent of the spatial dimension.

7.3.6 Optimization

Once the field points have been inserted, we have constructed a mesh for the domain that needs to be optimized to some extent. The goal is to optimize the mesh with respect to a quality criterion which is suitable for our purpose (a finite element style computation).

Up to now, we have considered a classical mesh generation problem. The aim is then to produce well-shaped elements, in other words isotropic elements that are as regular as possible (equilateral triangles in two dimensions and regular tetrahedra in three dimensions[7]). In terms of sizes, we have very little information about what is expected, thus we try to conform as best we can to the sizes defined at the boundaries and, elsewhere, to have a reasonably smooth variation.

While various quality measures have been proposed (see Chapter 18), a "natural" measure for the quality of a simplex is :

$$Q_K = \alpha \frac{h_{max}}{\rho_K} \tag{7.6}$$

[6]The control space, see Chapter 1, is the current mesh considered together with the h's of its vertices.

[7]A regular tetrahedron is an element with equilateral triangular faces.

where α is a normalization factor such that the quality of a regular element is one, h_{max} is the longest edge of the element, *i.e.*, its *diameter* and ρ_K is its inradius. This quality adequately measures the shape or aspect ratio of a given element. It ranges from 1, for an equilateral triangle (regular tetrahedron), to ∞, for a totally flat element; to return to a range of variation from 0 to 1, the inverse of $\mathcal{Q}_K$ could be used. Based on the above element quality, the quality of a mesh, $\mathcal{T}$, is given by :

$$\mathcal{Q}_M = \max_{K \in \mathcal{T}} \mathcal{Q}_K . \tag{7.7}$$

The aim is then to minimize this value. It could be observed that the point placement method results, in principle, in a good location for the field points. If so, then good quality elements may be expected. While effective in two dimensions, this result is not so easily attained in three dimensions. Thus some degree of optimization could be of interest.

Optimization procedures. The aim is to optimize the current mesh by means of local modifications. In this respect, two categories of optimization techniques can be identified (see Chapter 18), the topological techniques that preserve the point coordinates and modify their connections and the metric techniques that move the points while preserving vertex connectivity.

The local optimization operators associated with these techniques allow to move the nodes (for example, using a weighted barycentrage); to remove points; to remove edges (for example, by merging their endpoints) and to flip edges (in two dimensions) or faces (generalized edge swapping, in three dimensions).

A judicious use of these ingredients can efficiently improve a given mesh both in terms of CPU cost and of quality improvement.

7.3.7 Practical issues

In this short section, we would like to give some indications regarding computer implementation of the above scheme.

In terms of basic algorithms. Four steps of the previous scheme require careful computer implementation. The major effort concerns an efficient implementation of the Delaunay kernel, then the boundary enforcement problem as well as the optimization process must be considered together with the point creation step.

Regarding the Delaunay kernel, the operations that are involved are :

- a fast searching procedure so as to define the *base*,
- a convenient way of passing from one element to its neighbors to complete the *cavity* by adjacency,
- an inexpensive evaluation of the circumcenters and the circumradii of the mesh elements to evaluate the Delaunay criterion,
- a low cost update of a mesh when inserting a point.

Regarding the point creation step, the creation itself proves to be inexpensive while the filter which is needed can be relatively time consuming. A grid is then constructed to minimize the cost of this task (for instance using a *bucket sorting* algorithm, Chapter 2). Moreover, a cloud of points is inserted randomly (especially when the clusters contain a large number of points) in order to make the process more efficient.

Regarding the boundary problem, local modification operators must be carefully implemented. Note that this is also of interest for the optimization step as the required operators are basically the same.

Remark 7.4 *In the previous computational issues, we have not really mentioned accuracy problems. Indeed, the only thing we need is to be sure that a surface area (volume) is positive (to ensure the visibility criterion for a cavity).*

In terms of memory resources and data structures. First, it could be noted that a simple data structure probably provides a good chance of reaching a desirable level of efficiency. Moreover, the memory resources must be minimized as much as possible, which is the case with a simple structure.

Thus the internal data structure used must store (and maintain) the following (according to the facilities of the programming language)

- the point coordinates,
- the element vertices,
- the element neighbors (in terms of edge (face) adjacency),
- the element circumcenters,
- the element circumradii,
- and some extra resources (for instance for the grid used for the above filter).

7.3.8 Application examples

In this section, we give some application examples in both two dimensions and three dimensions. Some statistics are also presented.

In two dimensions, a mesh quality, *i.e.*, $\mathcal{Q}_M$ of Relationship (7.7), close to one can be expected regardless of the polygonal discretization of the domain boundary (meaning that equilateral triangles can be constructed[8] whatever the size of the given edges serving at their basis). In three dimensions, the expected value for $\mathcal{Q}_M$ depends on how good a tetrahedron one can construct for each given surface triangle. Hence, the three-dimensional quality is related to the quality of the surface mesh serving as data.

Table 7.1 gives some statistics about a selected series of examples of different geometries. In this table *np* is the number of vertices, *ne* is the number of tetrahedra, *target* is the targeted value for the tetrahedron with the worst quality while

[8]Obviously, if the point leading to this triangle falls within the domain.

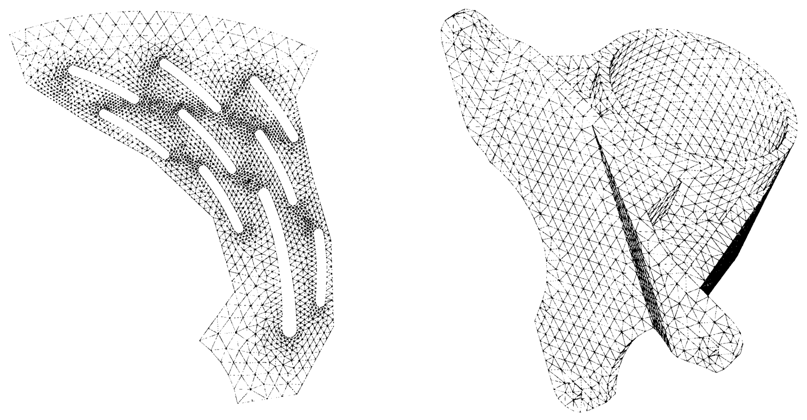

Figure 7.10: *Left-hand side : a two-dimensional geometry. The mesh contains* 2,445 *vertices and* 4,340 *triangles. The worst quality is* 1.12. *Right-hand side : a three-dimensional geometry (Data courtesy of MSC). The resulting mesh includes* 13,001 *tetrahedra and* 3,703 *vertices. The quality is* 5.83 *while the CPU cost is* 1.73 *seconds (HP9000/C180).*

$\mathcal{Q}_M$ is the value obtained. The row $1-2$ indicates the percentage of elements for which $1 < \mathcal{Q}_K < 2$, thus with a nice aspect ratio (*i.e.,* close to a regular tetrahedron) while t is the CPU time (HP 9000/735) required to achieve the mesh (including i/o).

-	np	ne	$\mathcal{Q}_{th}$	$\mathcal{Q}_M$	$1-2$	t
Case 1	105	286	7.38	9.61	66	0.36
Case 2	880	2,658	6.01	7.05	78	1.19
Case 3	1,917	7,230	10.24	11.65	76	2.75
Case 4	62,304	369,304	38.06	42.06	91	53.30

Table 7.1: Statistics related to the four selected examples (classical isotropic meshes constructed solely from the data of a discretization of the corresponding domain boundaries).

One should observe that $\mathcal{Q}_M$ is indeed in the range of *target* and that the number of nicely shaped elements is greater, in proportion to the total number, if the domain volume is large.

To appreciate more precisely the efficiency of the mesh generation algorithm, we give Table 7.2, where v is the number of elements constructed within a minute, for six selected examples with various sizes. Moreover, we denote by *Del* the part, in percentages, of the mesh generation algorithm devoted solely to the triangulation process (the point insertion process).

It can be seen that v is directly related to *Del* and that v is less for the small meshes than for the large ones.

-	np	ne	t (sec., HP 9000/735)	v	*Del*
Example 1	1,014	3,601	1.54	140,000	17
Example 2	36,252	191,279	28.26	406,000	49
Example 3	62,495	369,304	53.30	415,000	44
Example 4	214,184	1,243,871	179.	417,000	46
-	np	ne	t (sec., HP PA 8000)	v	*Del*
Example 5	518,759	3,067,937	373.62	492,000	54
Example 6	1,452,192	8,700,574	1125.	464,000	49

Table 7.2: Mesh generation algorithm efficiency.

To conclude, notice that the theoretical complexity of Delaunay type methods is not easy to calculate. We have mentioned that the geometry of the domain influences only a moderate amount of the global computational cost of the algorithm. However, the boundary enforcement procedure can be extremely costly in some cases. Nevertheless, the theoretical complexity is mostly in $O(n \log(n))$, n being the number of mesh elements,[George,Borouchaki-1997], as in the triangulation algorithms.

7.4 Other methods

A particular Delaunay type mesh generation method has been discussed based on the scheme given in the previous sections. But most of the steps included in this scheme can be modified giving rise to various different approaches. In the following paragraphs, we give some indications about the main variants that have been suggested.

7.4.1 Point insertion methods

Using an enclosing box is not strictly required. For instance, it is possible, given a set of points, to sort them so as to ensure that a given point, P_{i+1}, is always *exterior* to the current mesh (based on the i previous points). Applying such a trick to the boundary points-results in the mesh of the convex hull of this set of points. Then, internal points can be created and inserted returning to the classical point insertion method or an equivalent method. For instance, in two dimensions, a point is inserted in its *base* by splitting it and some diagonal swapping completes a Delaunay type mesh.

7.4.2 Boundary enforcement

In practice, two main variations can be developed to carry out the boundary enforcement step included in a Delaunay type meshing method.

- The boundary (and more generally the specified entities) are enforced after the field points have been inserted.

- The specified entity not present at the time their endpoints are inserted are split so as to ensure than the resulting partition will be automatically formed (we meet here the notion of a *Delaunay admissible entity* as already seen and as will be discussed in detail in Chapter 9).

7.4.3 Field point creation

Some popular methods, different to that based on the edge analysis given above, also prove useful for point creation. Among them are :

- the creation, under some conditions, of the centroid of the elements in the current mesh, [Hermeline-1980],
- the use of the circumcenters as internal points. In two dimensions, this leads to an upper bound on the angles of the triangles, [Holmes,Snyder-1988], [Chew-1989b],
- an advancing-front point placement strategy as discussed in Chapter 6,
- the use of the "variogram" [Tacher,Parriaux-1996],
- and many other methods, for instance, using a pre-determined set of points, [Weatherill-1985], [Baker-1986], [Mavriplis-1990],
- etc.

We briefly return to an interesting approach that combines an advancing-front point insertion technique with the insertion of the set of points using the Delaunay kernel.

Combined advancing-front-Delaunay approach. In this approach, the internal points correspond to optimal points created from a front, the elements being created during the insertion of these points into the current mesh using a Delaunay approach. This technique has been suggested in two and three dimensions (see [Merriam-1991], [Mavriplis-1992], [Muller *et al.* 1992], [Rebay-1993] and [Marcum,Weatherill-1995], [Frey *et al.* 1998], for example).

Here, the *front* represents the interface between two elements, one being classified as acceptable and the other not. As with the classical advancing-front approach, the initial front is the given boundary discretization. A first mesh of the bounding box of the domain is created that contains no (or very few) internal points and the boundary integrity is enforced (cf. Sections 7.3.2, 7.3.3). The tetrahedral elements in this mesh are classified into two categories, *accepted* (that of quality compatible with the target quality) and those to be *processed.* By assumption, all external tetrahedra are accepted. The *active* elements, adjacent to accepted elements, serve to define the current front. The front is then composed of the set of faces that separate an accepted element from an unaccepted one. The algorithm stops when all elements are accepted.

For each front item, an optimal point is computed so that the element formed by this point and the front item would be optimal. If an optimal point is located

outside the circumscribed disk (resp. ball) associated with the active element it should be connected with, this point is rejected because, during its insertion using the Delaunay kernel, the corresponding cavity will not contain the active element to be deleted. The remaining points are then filtered and inserted into the current mesh using the Delaunay kernel.

The final mesh is then optimized in the same way as in the classical approach (using local modifications).

7.5 Isotropic governed Delaunay meshing

Up to now, the mesh generation problem, the so-called classical problem, has been to mesh as best we can a domain using a discretization of its boundaries as the sole data input. We now turn to a different meshing problem.

We consider a domain provided through an adequate boundary discretization and we assume that the sizes (and, possibly, the directional features) of the elements that must be constructed are given. The problem becomes how to construct a mesh conforming to the above specifications. In the case where the information only specifies the element sizes, we have an *isotropic* mesh generation problem whereas in the case where directional information is specified we have an *anisotropic* meshing problem as will be discussed hereafter.

In principle the scheme proposed in Section 7.3.1 is still valid while replacing some of its components. At first, it can be proved that the Delaunay kernel is an adequate technique, without modification, to insert a point in a mesh. Actually, the main difference lies in the way the field points must be created so as to conform to the given size specification. On the other hand, slight variations in the optimization procedures must be also considered. A rather elegant way to achieve the desired solution is to introduce the notion of a *unit mesh* as already seen in the previous chapters and to use this when analyzing the edges. Moreover, some modifications must be made at the optimization phase.

7.5.1 Control space

The specification of a size distribution function enables us to define a control space. Notice that this data, common to most mesh generation algorithms (see, for instance, Chapters 5 and 6) is not so obvious to construct. A popular approach uses a grid or a background mesh whose cells (elements) encode the information about the desired element size. In practice, a size distribution function is associated with this domain covering-up. This simply means that a space control (Chapter 1) is defined in this way. If, from the spatial aspect, the control structure is a background mesh without internal points then the size distribution depends only on the sizes of the boundary entities. In the following section, we will see how to use this control space.

Using a background mesh. Provided with a background mesh[9], the size function is known in a discrete way, say at the vertices of this background mesh. Using this discrete map, a continuous size map can be obtained by means of interpolation.

Let P be an arbitrary point in the domain, the size $h(P)$ at P can be computed using the sizes $h(P_i)$, $i = 1,..d$ at the vertices P_i of the element enclosing P by means of a P^1 type interpolation, *i.e.*, :

- we find the element K within which point P falls,
- we compute $h(P)$ as the P^1-interpolate of the sizes $h(P_i)$ at this element vertices P_i.

In the case of an empty background mesh, the size distribution function is strongly related to the boundary discretization. Indeed, the sizes are computed based on this data alone.

Remark 7.5 *An alternative solution is to define the background mesh by a uniform grid or a quadtree-octree type structure.*

7.5.2 Field point creation

Provided the space control is well defined, it is now possible to turn to the creation of the field points. This is done based on the analysis of the edges of the current mesh. The aim is to construct good quality elements whose size is in accordance with the given specification. This latter requirement is satisfied if the edge lengths are close to one in the corresponding metric (cf. Chapter 1). Bear in mind that a mesh whose elements conform to a size specification is nothing other than a *unit mesh*.

Edge length. Let AB be an edge in the mesh. If, t lying between 0 and 1, $h(t)$ stands for a sizing function defined for this edge ($h(0) = h(A)$, $h(1) = h(B)$), then the length of edge AB with respect to $h(t)$ is :

$$l_{AB} = d_{AB} \int_0^1 \frac{1}{h(t)}\, dt \qquad (7.8)$$

where d_{AB} is the usual (Euclidean) distance.

Then, by definition, an edge AB conforms to the size specification if :

$$\frac{1}{\sqrt{2}} \leq l_{AB} \leq \sqrt{2}\,.$$

The key idea of the point placement strategy is to construct the points so that each segment defined by a pair of neighboring points is of length (close to) one.

One could observe that this notion is consistent. Indeed, if $h(t)$ is a constant function, say $h(t) = h, \forall t$, then $l_{AB} = 1$ means that $d_{AB} = h$ which is the expected value.

[9]For instance, a classical mesh (the mesh obtained at the previous iteration step in an adaptive process, cf. Chapter 21) or the boundary mesh completed by the boundary enforcement phase.

Field point creation. Now, based on this notion, the field points can be created by modifying accordingly the classical point creation method. Note that the edge based method is therefore a rather elegant method to which the above concept applies without major difficulty.

In practical terms, an iterative procedure is used, which aims at saturating the internal edges of the current mesh. The edges are analyzed and their normalized lengths calculated using an approximate method described below. The goal is to construct optimal elements (that are as regular as possible) of unit size. Let δ be a fixed threshold (usually $\delta < 1$, $\delta = 0.5$ for example). If $l_{AB} < \delta$, the edge AB is not split. Otherwise, the middle point Q_1 is introduced, and the process is iterated onto each of the sub-segments AQ_1 and Q_1B. We then find a series of points Q_i such that $l_{Q_iQ_{i+1}} < \delta$ and that :

$$l_{AB} = \sum_i l_{Q_iQ_{i+1}}\,.$$

The total number of points is then known along each edge. The next step consists in finding the location of the points along the edges. To this end, an index i is identified such that :

$$l_{0,i} = \sum_{j=0}^{j=i} \delta_j > 1$$

and the point P_1 is introduced, corresponding to the average value between the points Q_i and Q_{i+1}, weighted by the difference to one of this sum :

$$P_1 = Q_i + \omega \overrightarrow{Q_iQ_{i+1}}\,,$$

where $\omega = \frac{1-l_{0,i-1}}{l_{i-1,i}} d_{i-1,i}$, l_{i-1} denotes the normalized length of segment $Q_{i-1}Q_i$, $d_{i-1,i}$ being the Euclidean distance between the points Q_{i-1}, Q_i. The process is iterated until all edges are saturated.

Similar to the classical case and for the same reason, the points are filtered and inserted into the current mesh using the Delaunay kernel. Moreover, to obtain an optimal mesh, a constraint related to the areas (volumes) of the elements is added. The latter is not strictly required in two dimensions, a triangle having all of its edges of unit lengths is necessarily optimal. However in three dimensions, the same property does not hold, and explicit control of the volumes is thus required[10]

Optimizations. Similar to the classical case, it is generally useful to optimize the resulting mesh. The goal is to achieve well-shaped elements, which are as regular as possible, and whose sizes are compatible with the given specifications. The optimization stage is based on the same procedures as in the classical case, with a size-based criterion. For practical reasons, this procedure is performed in two steps, the mesh elements are first optimized with respect to the size criterion and then optimized with respect to the shape criterion.

In this discussion, the size prescription is assumed to be known analytically, the real case will be discussed in Chapter 21.

[10]A *sliver* is an element of null volume although its edges could be of unit length.

7.6 Extensions

When the information associated with the elements concerns sizes as well as directions, the construction step aims at creating *anisotropic* meshes, that is meshes in which the elements are stretched (for instance, those found in computational fluid dynamic simulations).

In this section, we briefly analyze the construction of anisotropic meshes, the construction of surface meshes and we give some indication of how to use the Delaunay approach for creating meshes in an adaptation scheme.

7.6.1 Anisotropic Delaunay meshing

Similarly, the approach adopted for the classical scheme can be followed but, in addition, it is necessary to modify the method used to create the field points (such as in the previous isotropic governed method), and the Delaunay kernel[11] itself must be extended. These two specific aspects are now briefly presented (see [George,Borouchaki-1997] for a detailed discussion).

Edge length with respect to a metric. First, we give a more subtle form of Relationship (7.8). Again let AB be an edge, which can be defined as : $AB(t) = A + t\overrightarrow{AB}, \quad t \in [0,1]$. Let $\mathcal{M}(M(t))$ be a two by two matrix defined by :

$$\mathcal{M}(M(t)) = \begin{pmatrix} \dfrac{1}{h^2(t)} & 0 \\ 0 & \dfrac{1}{h^2(t)} \end{pmatrix}, \tag{7.9}$$

if we consider a two dimensional problem (and a similar three by three matrix in three dimensions) where $h(t)$ is the specified size expected at point $M(t)$ then the distance between A and B with respect to the metric $\mathcal{M}$ is :

$$l_{\mathcal{M}}(AB) = \int_0^1 \sqrt{{}^t\overrightarrow{AB}\,\mathcal{M}(A + t\,\overrightarrow{AB})\,\overrightarrow{AB}}\,dt\,, \tag{7.10}$$

which is nothing other than a more general expression of Relationship (7.8).

Field point creation. Internal point creation for an anisotropic mesh is performed by replacing the length calculation of the classical scheme. More precisely, we use the previous definition of the normalized lengths, by replacing the matrix $\mathcal{M}(M(t))$ with the matrix (in two dimensions) :

$${}^t\mathcal{R}(t) \begin{pmatrix} \dfrac{1}{h_1^2(t)} & 0 \\ 0 & \dfrac{1}{h_2^2(t)} \end{pmatrix} \mathcal{R}(t)\,. \tag{7.11}$$

[11] See the theoretical remark at the end of this chapter.

In this expression, $\mathcal{R}(t)$ specifies the two directions that are expected at point $M(t)$ while $h_1(t)$ (respectively $h_2(t)$) indicates the desired sizes at the same point following the first (resp. the second) direction previously mentioned. Then the point placement strategy is identical to that of the previous isotropic case.

Point insertion scheme. As mentioned earlier, the classical Delaunay kernel is no longer suitable for an anisotropic mesh generation problem. Therefore, the classical construction (Relationship (7.2)),

$$\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{C}_P + \mathcal{B}_P\,, \tag{7.12}$$

where $\mathcal{C}_P$ is the *cavity* and $\mathcal{B}_P$ is the *ball* associated with point P, must be extended to the present context.

The idea is to define $\mathcal{C}_P$ so as to conform to the anisotropy which is desired. To this end, we detail the condition which implies that a given element is a member of $\mathcal{C}_P$. Indeed, in the classical context, an element K is in the cavity if :

$$\alpha(P,K) = \frac{d(P,O_K)}{r_K} < 1\,, \tag{7.13}$$

(d being the usual distance), with O_K the center of the circumcircle (circumsphere) corresponding to K and r_K the radius of this circle (sphere). We first replace this relation by :

$$\alpha_{\mathcal{M}}(P,K) = \frac{l_{\mathcal{M}}(P,O_K)}{r_K} < 1 \tag{7.14}$$

where now,

- O_K stands for the point equidistant to the vertices of K, in terms of $l_{\mathcal{M}}$ the length related to the matrix (metric) $\mathcal{M}$, and
- r_K is $l_{\mathcal{M}}(P_1, O_K)$, P_1 being one of the vertices of K.

As a result, the cavity is evaluated following the anisotropic context. In practice, a computer implementation of the above relationship is not practical since a non-linear system is involved. Thus, an approximate solution must be found. A possible answer could be to solve the following system :

$$\left\{ \begin{array}{lcl} l_{\mathcal{M}(P)}(O_K, P_1) & = & l_{\mathcal{M}(P)}(O_K, P_2) \\ l_{\mathcal{M}(P)}(O_K, P_1) & = & l_{\mathcal{M}(P)}(O_K, P_3) \end{array} \right. . \tag{7.15}$$

In this way, one obtains O_K (P_i being the vertices of K). Then we check if :

$$\alpha_{\mathcal{M}(P)}(P,K) < 1\,, \tag{7.16}$$

where $\mathcal{M}$ is approached by $\mathcal{M}(P)$, the metric associated with the point under insertion. The following theorem, which holds in two dimensions, shows that the previous characterization is constructive.

Theorem 7.5 *The relationship $\alpha_{\mathcal{M}(P)}(P,K) < 1$ results in a valid Delaunay kernel.*

The proof of the above theorem consists of checking that the so-defined cavity is a star-shaped region with respect to P. Unfortunately, this result does not extend to three dimensions where a more subtle method must be used. Indeed, we consider exactly the same construction but we have to verify explicitly that the current cavity is a star-shaped region (this is completed by an algorithm that corrects a possibly invalid cavity).

To conclude, one should note that other approximate solutions can be used resulting in the desired Delaunay type kernel. These solutions infer point P and some other points naturally present in the context.

Optimizations. Similar to the governed anisotropic case, an optimization stage aims at optimizing the size and shape quality of the resulting mesh. This stage is based on local topological and geometrical operations.

7.6.2 Surface meshing

The generation of surface meshes is generally considered to be a difficult problem, for which two approaches are *a priori* possible, a direct and an indirect approach.
Direct approaches consist of performing a classical meshing technique directly onto the surface, without using any mapping. The element sizes and shapes are controlled by taking into account the local variations of the surface intrinsic properties. The boundary curves are first discretized before meshing the domain using a classical scheme. According to this scheme, a Delaunay type approach does not seem well-suited to performing this task. However, the indirect approach, see below, is widely possible.

Parametric surfaces. The generation of parametric surface meshes can be obtained via the mesh of the parametric space. This technique has been suggested by several authors (see for instance [George,Borouchaki-1997] or Chapter 15 for a detailed overview of this approach). The main interest of this approach is to reduce the problem to a purely two-dimensional problem. Let Ω be a domain in $\mathbb{R}^2$ and let σ be a sufficiently smooth function, then the surface Σ defined by the application

$$\sigma : \Omega \longrightarrow \mathbb{R}^3, \ (u,v) \longmapsto \sigma(u,v)$$

can be meshed using a two-dimensional mesh generation method in Ω, then mapping this mesh, via σ, onto $\mathbb{R}^3$. The sole constraint is to have a method that allows anisotropic meshes to be created. The metric map is indeed based on the intrinsic properties of the surface. The Delaunay method appears to be particularly well-suited to this kind of problem.

7.6.3 Mesh adaptation

The adaptation of meshes to the physical properties of the considered problem makes it possible to improve, theoretically, the accuracy of the numerical results as well as the convergence of the computational scheme. This problem will be

dealt with in Chapter 21. We also mention that the mesh adaptation is based on a governed mesh generation method (the size map is provided here by an *a posteriori* error estimate). Therefore, the Delaunay method can be easily used in this context.

Concluding remarks

Delaunay type mesh generation methods prove to be efficient, robust and flexible enough to handle classical as well as governed mesh creation. In this respect, they are of major importance in solving adaptive computational problems as will be discussed in Chapter 21.

In addition, one could note that anisotropic Delaunay type meshing methods will be an important ingredient for parametric surface mesh generation (see Chapter 15) where some extent of anisotropy is required in the parametric space, based on the intrinsic characteristics of the geometry, if one wants to construct a surface mesh close to the latter.

A theoretical remark. As discussed in the previous section, a Delaunay type method has been advocated to handle isotropic or anisotropic meshing problems. In this respect, the classical Delaunay kernel (as itself or after an approximation) has been used to connect the vertices.

A purely theoretical view of the problem could refute our point of view. In fact, due to the metric map supplied as a sizing (directional) information, the basic notion of a distance between two points is no longer Euclidean. Indeed, the distance measure varies from one point to the other. In terms of the Voronoï diagram that can be associated with a set of points in such a metric context, we don't return to the classical diagram. For instance, the perpendicular bisector of two points (in $\mathbb{R}^2$) is not a straight line but a curve. Thus the Delaunay triangulation, dual of this diagram, is not composed of *affine* (straight-sided) triangles (considering again a two-dimensional problem). Indeed we face a Riemaniann problem where the curve of minimum length between two points, the so-called *geodesic*, is no longer a straight segment.

Nevertheless, in our context, the only objective is to construct an affine triangulation and therefore the purely theoretical discussion above is not an issue. In fact, we use the Delaunay background as a basic scheme that enables us to design a constructive mesh generation method (which is no longer Delaunay).