Chapter 7

Delaunay-based Mesh Generation Methods

Delaunay triangulation and the 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 (CG for short). 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] together with [Ruppert-1995] among various 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 many engineering problems where theoretical results are used or revisited so as to obtain concrete algorithms.

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 1980s and many others in the next decade such as [Weatherill-1985], [Mavriplis-1990], together with [George, Borouchaki-1997].

This chapter includes six parts. The first recalls some theoretical issues regarding Delaunay triangulation. The second discusses the notion of a constrained triangulation. We then show how to develop a Delaunay-type mesh generation method. The fourth part briefly introduces several variants. Finally, extensions are proposed. We explain how to complete a mesh conforming to a pre-specified size map and how to generate anisotropic meshes (used, in particular, when dealing with parametric surface; see Chapter 15). Comments are added about weighted

and anisotropic diagrams and triangulations together with potential applications.

7.1 Voronoï diagram and Delaunay triangulation

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

The Voronoï diagram

Let $\mathcal{S}$ be a finite set of points $(P_i)_{i=1,\ldots,n}$ 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 $\mathbb{R}^d$.

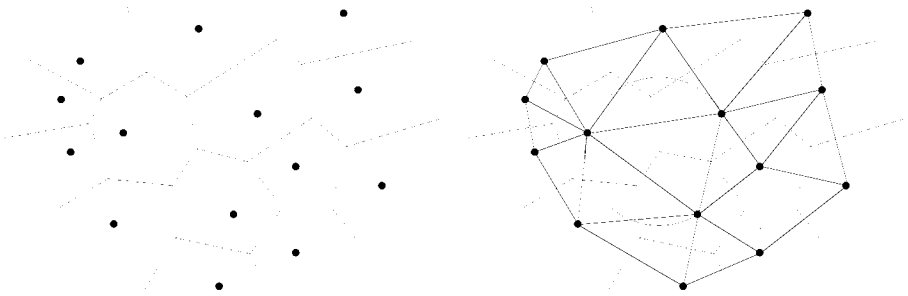

Figure 7.1: *Left-hand side: Voronoï diagram (in two dimensions). Right-hand side: corresponding Delaunay triangulation.*

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 according to d) provided the points in $\mathcal{S}$ are locally in general position (cf. Chapter 1 for this notion). 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).

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.*

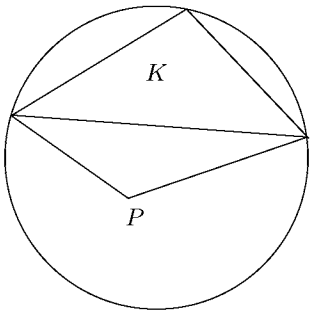

Figure 7.2: *B.N. Delaunay and his famous criterion. In this two-dimensional example, the "empty sphere" criterion is violated as the open disc of triangle K encloses point P. Note that this example is special as the point P is the vertex of a triangle adjacent to K which is opposite the common edge.*

The empty sphere criterion. In two dimensions, this definition refers to the open disk circumscribing a triangle while in three dimensions it concerns the open ball circumscribing a tetrahedron. This criterion is also referred to as the *Delaunay criterion.* Note that the criterion is referred to as the empty sphere criterion while would be better referred to as the empty ball criterion.

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

[1]Published in French in 1934, see [Delaunay-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*

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 provides 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 for 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 locally 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:

- cavity $\mathcal{C}_P$ is the set (the union) of elements in $\mathcal{T}_i$ whose open circumballs contain point P and
- 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.

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 (a proof, using the Voronoï duality, can be found in various references and in [George, Borouchaki-1997]).

---

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."

[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.

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 Relation (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 Relation (7.2), meaning that $\mathcal{C}_P$ and $\mathcal{B}_P$ are exactly the same as 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 us assume that there exists one element in $\mathcal{R}_P$ without P as one of its vertex, this element is then 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 completes $\mathcal{T}_{i+1}$, 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, we consider then 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 $\mathcal{T}_i$). Thus, 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 and thus 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 makes possible 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, which is 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, that 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. □

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, or five or more co-spherical points are in the initial set with corresponding empty circumballs, which is not likely to be plausible for realistic engineering applications, apart from a case where all the points in the set are co-spherical).

Then, after 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 star-shapedness 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.*

The correction step is typically due to the numerical problems necessarily encountered when encoding the triangulation algorithm (and is representative of the difficulty of encoding geometric algorithms). This merits the following comments.

A typical numerical problem. The cavity construction is done by adjacency given the base as initialization (note that other methods exist). However, this solution offers a guarantee about the connectivity of the set; indeed, it prevents the obtaining of a multi-connected set.

The question is to decide if a given element is a member of the cavity of the current point P. Let K be the visited element, let O_K be its circumcenter (i.e.,

the center of its circumcircle (circumsphere)) and let r_K be the corresponding circumradius. Theoretically speaking, it is merely necessary to consider the ratio $\alpha(P,K) = \frac{d(P,O_K)}{r_K}$, (called the Delaunay measure) and to check if

$$\alpha(P,K) < 1\,.$$

The relevant check leads to comparing $d(P,O_K)$ and r_K. As these two quantities are not precisely valued, this check may be inaccurate, specifically, if the region in which P falls is close to the boundary of the disk (the ball) C_K of K. This uncertainty may result in dramatic results and the Delaunay kernel, (Relation 7.2), may result in a non-valid triangulation. These ambiguous configurations can fall in two classes:

- the cavity is not empty, meaning that there exists at least one vertex of a previously created element inside the cavity. This default is usually due to a proximity problem,
- the cavity is not a connected set. In general, this denotes a cocyclic (co-spherical) configuration.

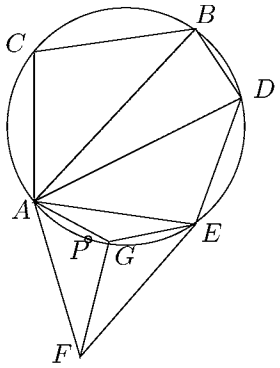

Figure 7.3: *The two ambiguous configurations.*

The first case leads to a vertex being missed (the resulting triangulation is still valid albeit wrong in this respect). The second case leads to a triangulation having overlapping regions. Figure 7.3 depicts these two situations.

- The first case of failure, due to imprecise computations, corresponds to the case where all the triangles in the figure are picked; point G is then strictly included in the cavity.
- The second case of failure, also caused by imprecise calculations, corresponds to the case where all the triangles of the figure except triangle (ADB) are selected, thus resulting in a non-connected cavity.

To overcome these problems, several solutions have been investigated. They consist of

- not introducing any point causing a problem,
- (slightly) perturbing all points leading to a problem,
- introducing a threshold value, ε, in the comparisons,
- performing exact computations,
- or, finally, suppressing the ambiguity using a different formulation of the algorithm.

The first solution requires that the current point is placed on a stack, such that its insertion will be done later when the local context is modified.

The second approach moves the point upon insertion and modifies the quantities involved in the construction, thereby expecting to remove the ambiguity.

The third solution, which introduces a threshold value ε in the comparison, has been investigated by numerous authors but does not lead to satisfactory results. An adequate value ε for a given case is not suitable for other cases.

The fourth approach implicitly assumes integer-type coordinates for the vertices and is not based on the Delaunay measure (meaning that the circumcenters and the circumradii are not computed or updated). Instead, it is related to the equivalent formulation (let us consider the two-dimensional case)

$$\Delta_K(x_P, y_P) < 0$$

where $\Delta_K(x, y)$ is the *inCircle* predicate of Chapter 2. This inequality includes quantities in the range of a length to the power $d + 2$ which involve additions (subtractions) and multiplications only. Consequently, a restriction is imposed on the vertex coordinate's range. In other words, the minimal distance between two points is limited. Indeed, if b is the number of bits of the mantissa of a double memory word, the largest value (denoted as l) that can be expressed in the above expression must satisfy the following relation

$$l \leq 2^{\frac{b}{d+2}} .$$

Assuming that the vertex coordinates start from the origin, this relation states that these coordinates must range from 0 to $l \leq 4096$ in two dimensions and from 0 to $l \leq 1024$ in three dimensions and, on the other hand, that the distance between two points is at least 1 with a typical computer[3] for which $b = 50$. These limits give both the maximal possible number of points according to the d directions as well as the minimal distance between two points. We have introduced, *de facto*, the *separating power* or the *resolution* of the method. The limit resulting from this discussion is obviously too restrictive and, consequently, while *a priori* elegant, this method is not adequate in general.

A determinant evaluation method can be found in [Avnaim *et al.* 1994], which overcomes this limit at the expense of increased complexity.

[3]Double precision words are employed, with *a priori* 51 significative bits. For safety reason, we limit ourselves to 50 bits. It should be noted that this limit depends on the technology; actually, 128 bit computers are widely used.

Another way to avoid this limit is to introduce an extended arithmetic and, more specifically, to use infinite precision[4] in the computations. See for instance, [Guibas *et al.* 1989], [Fortune, Van Wyk-1993] (among others) or [Perronnet-1988b] and [Peraire, Morgan-1997] for meshing applications.

The fifth method is the one we would like to recommend. We assume that the vertex coordinates are of integer type and we propose a new formulation for the Delaunay kernel resulting in a robust and exact algorithm in this context. The discussion of this method is the aim of the following paragraph. Briefly, the assumptions about the coordinates allow us to find the base exactly. This base enables us to define an approximated cavity which is furthermore corrected so as to ensure the expected properties (emptyness, connecteness and star-shapedness). This method will result in a valid triangulation which will not strictly be Delaunay.

Cavity correction A way to prevent a failure in the construction is to explicitly check what is needed in terms of properties. This motivated the following so-called cavity correction algorithm.

The problem centers on expressing the Delaunay kernel in such a way as to obtain an efficient constructive algorithm despite the round-off errors that may occur in the actual computation scheme. As already mentioned, the given coordinates are assumed to be of integer type ensuring exact surface (or volume) evaluations (obviously, to this end, we compute twice the surface area or six times the volume so as to avoid the division needed for an exact value). In this context, a two part algorithm is proposed. This algorithm includes the above method serving to initialize the cavity, the latter being wrong in some cases. The process is completed by a new algorithm, referred to as the *correction algorithm.* Let P be the current point to be inserted and let $\mathcal{T}_i$ be the current triangulation; the first stage of the method leads to

- using the Delaunay measure to construct the cavity associated with P, $\mathcal{C}_P$, by adjacency, given the base.

As this algorithm can result in a non-valid cavity, a correction step is needed as the second part of the process. This correction relies in removing some elements from $\mathcal{C}_P$ to meet the desired properties again. Thus, the correction algorithm can be described as follows

- if a vertex of $\mathcal{T}_i$ falls in the cavity, find one of the simplices[5] in $\mathcal{C}_P$, not in the base, having this point as vertex and remove this element from $\mathcal{C}_P$,
- if there is a $(d-1)$ boundary face of $\mathcal{C}_P$ not visible by P, pick and remove the simplex having this face from the cavity,

[4]This approach requires some comments. Indeed, if we consider the example of a surface of a triangle strictly positive when valued in infinite precision, it is not obvious that the same surface will be computed in the same way when used in a different software package.

[5]We can select as a simplex candidate the first element found that can be removed or select one of the possible simplices, enjoying a desired property.

- repeat this process as long as the number of elements in the cavity changes. One iteration results in starting the whole analysis of the elements remaining in the cavity again. This is done either from the base and proceeding by adjacency or this can be done by considering the last element not affected by the actual process.

Note that this correction algorithm converges. Indeed, in the worst case, the cavity is reduced to the base thus leading to the convergence. Also, using adjacency relationships in the process ensures that the cavity is a connected set; as the base is necessarily included in the cavity, the latter contains point P. Finally the visibility checks (surface or volume computations according to d) guarantee the star-shapedness property of the cavity.

In summary, the proposed algorithm is constructive and the computations are integer in nature (and thus are exact) or such that only surface (volume) evaluations, or equivalent computations, have been used. Thus, it is possible to obtain a computationally efficient and robust algorithm with a limit of application, as discussed above, partly extended. Indeed, the limit is now $l \leq 2^{\frac{b}{d}}$, leading to $l \leq 33554432$ in two dimensions and $l \leq 65536$ in three dimensions. Obviously, an order of magnitude has been obtained and the separation power of the method is increased. Hence, this method is usually well-suited. The l value gives the separation power of the method and indicates the maximal number of points in each direction (the minimum distance from point to point being 1).

Actually while assuming integer coordinates in the discussion, the same idea of using explicit validation works well with real coordinates.

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 (Relation (7.2)) can be constrained so as to preserve a specified edge (a face) 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 visiting the elements by adjacency, we do not pass through the specified edges (faces) 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 Relation (7.2) determines $\mathcal{T}_{i+1}$, a valid triangulation having P as element vertex.*

This theorem just means that Relation (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 Enforcing a constraint

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 7.4 where a simple example is depicted.

The problem we face is then to enforce[6] 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 delete all the edges

[6]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.

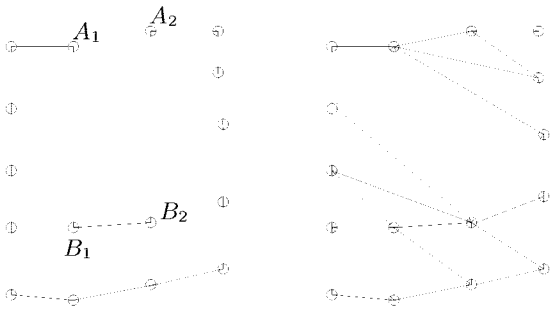

Figure 7.4: *In this simple two-dimensional example, we have displayed the triangles which are intersected or close to two missing edges (in particular, some triangles, part of the triangulation of the convex hull, are not shown). Actually, edges A_1B_1 and A_2B_2 are missing in the triangulation although their endpoints are element vertices.*

that intersect a specified segment (actually, an edge that must be constructed) and results in the desired solution.

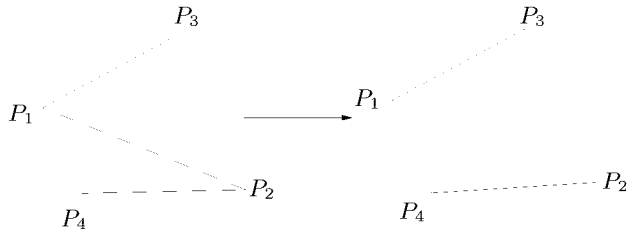

Figure 7.5: *Diagonal swapping. The quadrilateral formed by this pair of adjacent triangles is a convex region, so its diagonal can be swapped.*

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

Two theoretical issues can be invoked 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). Nevertheless, the previous theorems hold in two dimensions only.

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 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[7] 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 more 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.

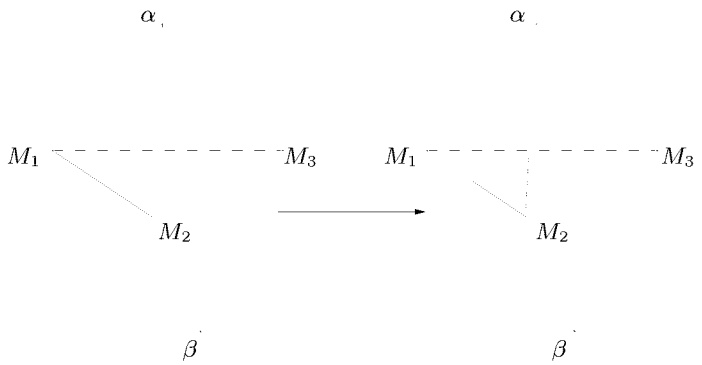

Figure 7.6: *The polyhedron consisting of the two tets $M_1M_2M_3\alpha$ and $\beta M_1M_2M_3$ (right) is convex and can be re-meshed using the three tets $M_1\alpha\beta M_2$, $M_2\alpha\beta M_3$ and $M_3\alpha\beta M_1$. Conversely, this three element configuration can be re-meshed by means of two elements. In the first transformation, a face has been removed while in the second an edge has been removed.*

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* (Chapter 2). A shell is the set of tetrahedra sharing a given edge. Then, the

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

three-dimensional version of the swap operator can be seen as remeshing this polyhedron by suppressing the common edge (Figures 7.6, 7.7 and Chapter 18).

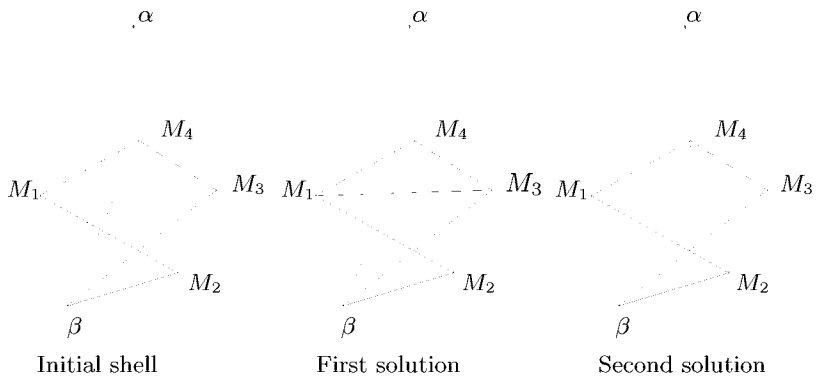

Figure 7.7: *The initial pattern is the shell associated with edge $\alpha\beta$. Two alternate remeshings of this polyhedra are possible if it is convex.*

Steiner points. A rather obvious example (Figure 7.8) 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, we 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.

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.

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 thus 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).

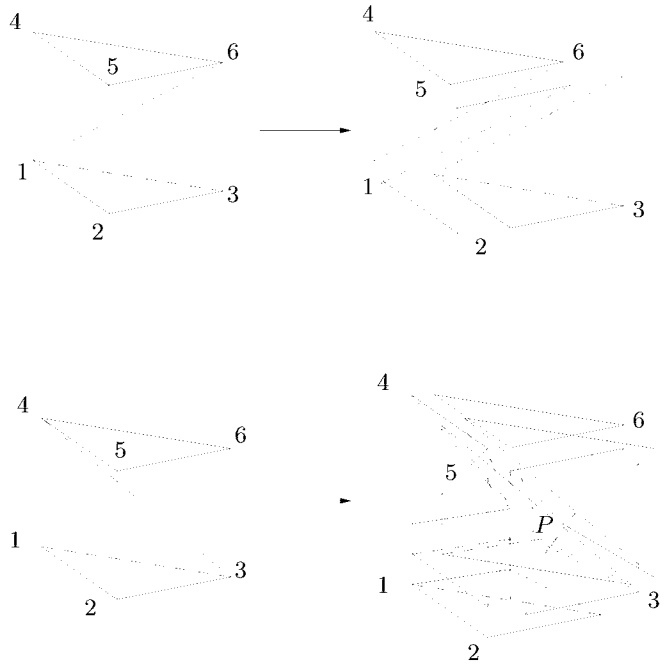

Figure 7.8: *The prism at top can be partitioned by simply using three tetrahedra. The prism at the bottom, the so-called Schönhardt polyhedron, cannot be split with three tetrahedra. This prism differs from the previous one in the way in which its quadrilateral faces are decomposed into triangular faces. 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.*

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 numerically proved as, theoretically speaking, we use arguments like "there exists a non-empty visibility kernel", has proved to work well in most concrete 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 is necessary 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, etc.),

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 from 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.

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:

- 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.

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).

7.3.1 Simplified Delaunay type triangulation method

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

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

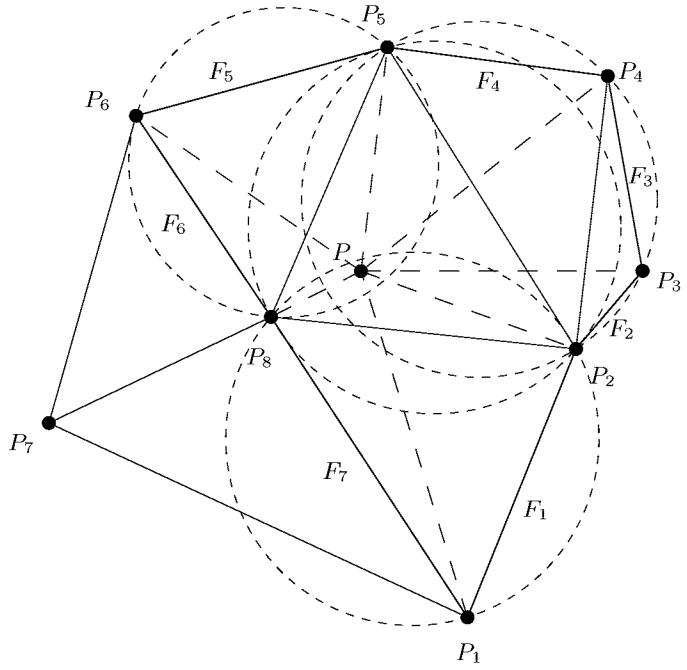

Figure 7.9: *Inserting point P (P included in the current mesh). In this two-dimensional example, only the triangles close to point P are displayed. The base is reduced to the triangle $(P_2P_5P_8)$. The cavity is formed by the triangles in solid lines. The external faces of this cavity are denoted by $F_1, F_2, ..., F_7$. The ball consists of the elements in dotted lines. It is formed by joining P with the F_is.*

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.

7.3.2 Boundary integrity and domain identification

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.

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 (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;
   - 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;
   - 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.

7.3.3 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[8], 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).

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.

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

[8]The necessary Steiner points.

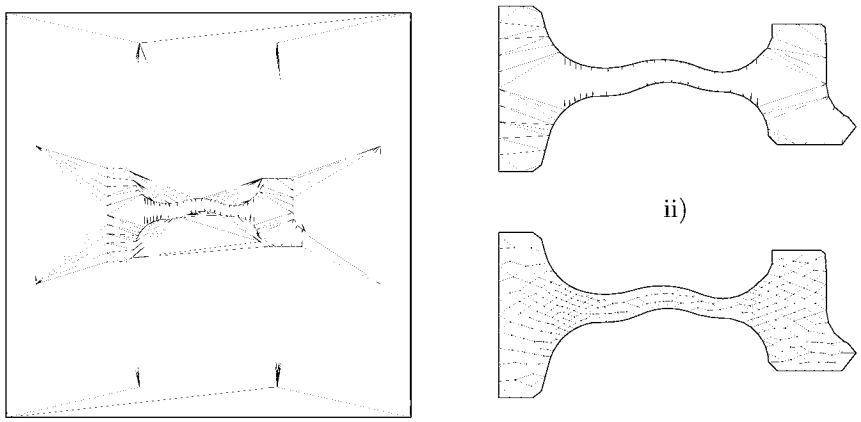

Figure 7.10: *i) Mesh of the box enclosing a domain for a mesh generation problem in two dimensions. The four corners used to define the box enclosing the domain can be seen together with some extra points defined between the box and the domain for efficiency reasons. ii) Corresponding empty mesh. This 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.*

- $h(0) = h_A$ 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,

we can define a sequence α_i (and thus the corresponding P_is) as:

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

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:

$$\begin{cases} \sum_{i=0}^{n} \alpha_i &= d \\ \alpha_{i+1} &= \alpha_i + r \end{cases} \tag{7.5}$$

to find both r and n. The solutions are:

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

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_is 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[9] 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, simply using 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 (Relation (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.

7.3.4 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). Indeed, while being Delaunay (in most of the domain, in specific far from the boundaries), the mesh quality is not necessarily what is needed. This means that the Delaunay criterion (i.e., a bound about the angles) is not, *stricto sensu*, a quality criterion.

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[10]). In terms of sizes, we have very little information about what is expected, so 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 (Chapter 18), a "natural" measure for the quality of a simplex is:

$$\mathcal{Q}_K = \alpha \frac{h_{max}}{\rho_K} \tag{7.6}$$

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:

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

[9]The control space (see Chapter 1) is the current mesh considered together with the hs of its vertices.

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

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, mainly due to slivers.

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 (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 make it possible 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 edges and faces (generalized edge swapping, in three dimensions).

7.3.5 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; see 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[11].

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.

[11] Note that other more subtle strategies can be employed to minimize the overall cost of this point insertion process.

Remark 7.4 *In the previous computational issues, we have not really mentioned accuracy problems or round-off errors. 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 as already discussed or that an element is valid (at the boundary or optimization step) which return to the same type of control.*

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,
- some extra resources (for instance, for the grid used for the above filter).

7.3.6 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 Relation (7.7), close to one can be expected regardless of the polygonal discretization of the domain boundary (meaning that equilateral triangles can be constructed[12] 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 can be constructed 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, recorded in 1997, 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 $\mathcal{Q}_M$ 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 at 100 MHz) required to achieve the mesh (including i/o).

It should be noted 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.

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