HERMES Science Publishing
9 Park End Street
Oxford, OX1 1HH
United Kingdom

hermes_science@BTinternet.com
http://www.hermes-science.com

A catalogue record of the book is available from the British Library
British Library Cataloguing in Publication Data

ISBN 1-903398-00-2

Mesh Generation

application to finite elements

Pascal Jean Frey
Paul-Louis George

Hermes Science publishing
Oxford & Paris

Contents

Introduction

General introduction

Mesh generation techniques are now widely employed in various engineering fields that make use of physical models based on partial differential equations (P.D.E.). Numerical simulations of such models are intensively used for design, dimensioning and validation purposes. One of the most frequently used methods, among many others, which is used in this respect is the *finite element* method. In this method, a continuous problem (the initial P.D.E. model) is replaced by a discrete problem that can actually be computed thanks to the power of currently available computers. The solution to this discrete problem is then an approximate solution to the initial problem whose accuracy is based on the various choices that were made in the numerical process.

The first step (in terms of actual computation) of such a simulation involves constructing a mesh of the so-called computational domain (*i.e.*, the domain where the physical phenomenon under interest occurs) so as to replace the continuous region by means of a finite union of (geometrically simple and bounded) elements such as triangles, quads, tetrahedra, pentahedra, prisms, hexahedra, etc., based on the spatial dimension of the domain. For this reason, mesh construction is an essential pre-requisite for any numerical simulation of a P.D.E. problem. Moreover, mesh construction could be seen as a bottleneck for a numerical process in the sense that a failure in this mesh construction step renders any subsequent numerical simulation impossible.

While there are numerous textbooks, monographs, handbooks, thesis dissertations and journals as well as conferences, symposia and workshops devoted to both finite element theory and practical issues, it appears that this is not so clearly the case for meshing technologies. Indeed, finite element aspects are considered worldwide to be an important topic of study at university whereas meshing technology aspects are frequently neglected as a full topic of interest. Thus, in the best cases, this aspect is briefly mentioned as a (purely) technical point that is possibly non-trivial (but, doubtless, one which can be successfully handled by commercial software packages !).

This situation deserves some comments. Actually, mesh construction for numerical simulation purposes involves several different fields and domains. These include (classical) geometry, so-called computational geometry and numerical simulation (engineering) topics coupled with advanced knowledge about what is globally termed computer science. The above classification in terms of disciplines which can interact in mesh construction for numerical simulation clearly shows why this topic is not so straightforward. Indeed, people with a geometrical, a computational geometry or a purely numerical background do not have the same perception of what a mesh (and, *a fortiori*, a computational mesh) should be, and subsequently do not share the same idea of what a mesh construction method should be.

To give a rough idea of this problem, we mention, without in any way claiming to be exhaustive, some commonly accepted ideas about meshes based on the background of those considering the issue.

From a purely geometrical point of view, meshes are mostly of interest for the properties enjoyed by such or such geometrical item (and probably not the full mesh itself), for instance, a triangle. In this respect, various issues have been investigated regarding the properties of such an element including aspect ratios, angle measures, orthogonality properties, affine properties and various related constructions (centroids, circumcenters, circumcircles, incircles, particular (characteristic) points, projections, intersections, etc.).

A computational geometry point of view mainly focuses on theoretical properties about triangulation methods including a precise analysis of the corresponding complexity. In this respect, Delaunay triangulation and its dual, the Voronoï diagram, have received much attention since nice theoretical foundations exist and lead to nice theoretical results. However, triangulation methods are not necessarily suitable for general meshing purposes and must, to some extent, be adapted or modified.

Mesh construction from a purely numerical point of view (where, indeed, meshes are usually referred to as triangulations) tends to reduce the mesh to a finite union of (simply shaped) elements whose size tends towards 0 :
" Let $\mathcal{T}_h$ be a triangulation where h tends to 0, then ..., "
the above $\mathcal{T}_h$ being provided in some way or other (with no further details given on this point), the construction of $\mathcal{T}_h$ is no longer a relevant problem if a theoretical study is envisaged (such as a convergence issue for a given numerical scheme).

In contrast to all the previous aspects, people actually involved in mesh construction methods face a different problem. Provided with some data, the problem is to develop methods capable of constructing a mesh (using a computer) that conforms to what the "numerical" and more generally the "engineering" people need. With regard to this, the above h does not tend towards 0, the domain geometry that must be handled could be arbitrary and a series of requirements may be demanded based on the subsequent use of the mesh once it has been constructed. On the one hand, theoretical results about triangulation algorithms (mainly obtained from computational geometry) may be not so realistic when viewed in terms of actual computer implementation. On the other hand, engineering requirements may differ slightly from what the theory states or needs to assume.

As a brief conclusion, people involved in "meshing" must make use of knowledge from various disciplines (mainly geometry and computational geometry) then combine this knowledge with numerical requirements (and computational limitations) to decide whether or not such or such *a priori* attractive aspect (for a particular discipline) is relevant for a meshing process. In other words, good candidates for mesh construction activities must have a sound knowledge of various disciplines so as to be able to select from these what they really require for a given goal.

Having made these "philosophical" remarks about meshing technologies, we should point out that this topic is becoming increasingly recognized as a subject of interest in its own right, not only in engineering but also in some universities as well. In practice the subject is being addressed in many places all over the world, and a large number of people are spending a great deal of time on it. A few specialized conferences and workshops do exist and papers on meshing technologies can be found in various journals. Up to now, however, it has not been so very difficult to enumerate all the available books[1] entirely (or substantially) devoted to meshing technologies, which indicates that there is still a shortage of textbooks about this topic. The reader could well find here one reason which motivated this work, together with the fact that the technology has developed very rapidly over the last few years. The present work could also be seen as a successor to the reference [George-1991].

Purpose and scope

The scope of this book is multiple and thus the potential categories of intended readers are various. As a first remark, we like to think that the theoretical background that is strictly necessary to understand the book is anything but specialized. We are confident that a reasonable knowledge of basic geometry, a touch of computational geometry and an idea of what a numerical simulation is (for

[1] Probably the very first significant reference about mesh generation is the book by Thompson, Warsi and Mastin, [Thompson *et al.* 1985], authored in 1985, which mainly discussed structured meshes. A few years after, in 1991, a book by George, [George-1991], was written which aimed to cover both structured and unstructured mesh construction methods. More recently, a book authored in 1993 by Knupp and Steinberg, [Knupp,Steinberg-1993], provided an updated view of structured meshes while, in 1998, a book fully devoted to Delaunay meshing techniques, [George,Borouchaki-1997], appeared. Among books that contain significant parts about meshing issues, one can find the book authored by Carey in 1997, [Carey-1997].

Thus, it is now possible to find some references about mesh technology topics. In this respect, one needs to see the recent publication (at the time of writing this book), of the Handbook of Grid Generation, edited by Thompson, Soni and Weatherill, [Thompson *et al.* 1999], which, in about 37 chapters by at least the same number of contributors, provides an impressive source of information. To end, notice the scheduled publication of another collective work, "Meshing", in the MIM (Mécanique et Ingénierie des Matériaux) series published by Hermès, Paris.

instance some basic notions about the finite element method) provide a sufficient basis for the reader to profit from the material discussed. With regard to this, one of our objectives has been to make most of the discussions self-contained.

One issue underlying some of the discussions developed in the book was what material the reader might expect to find in such a book. A tentative answer to this point has led us to incorporate some material that could be judged trivial by readers who are already familiar with some meshing methods, yet we believe that its inclusion may well prove useful to less experienced readers.

We have introduced some recent developments in meshing activities, even if they have not necessarily been well validated (at least at the industrial level), so as to permit advanced readers to initiate new progress based on this material.

It might be said that constructing a mesh for such or such a purpose (academic or industrial) does not strictly require knowing what the meshing technologies are. Numerous engineers confronted daily with meshing problems, as well as graduate students facing the same problem, have been able to complete what they need without necessarily having a precise knowledge of what the software package they are familiar with actually does. Obviously, this point of view can be refuted and clearly a minimum knowledge of the available meshing technologies is a key to making this mesh construction task more efficient. Finally, following the above observations, the book is intended for both academic (educational) and industrial purposes. In this respect, various categories of persons are targeted including graduate and post-graduate students, researchers from universities, public and private laboratories, developers as well as users and, why not, managers.

Synopsis

Although we could have begun by a general purpose introduction and led on to a presentation of classical methods, followed by a discussion of advanced methods, specialized topics, etc., we chose to structure the book in such a way that it may be read sequentially. Relevant ideas are introduced when they are strictly necessary to the discussion, which means that the discussion about simple notions is made easy while when more advanced discussions are made, the more advanced ideas are given at the same time. Also, some almost identical discussions can be found in several sections, in an attempt to make each section as self-contained as possible.

The book contains 23 chapters. The first three chapters introduce some general purpose definitions (Chapter 1) and basic data structures and algorithms (Chapter 2), then classical mesh generation methods are briefly listed prior to more advanced techniques (Chapter 3). The following chapters provide a description of the various mesh generation methods that are in common use. Each chapter corresponds to one type of method. We include discussions about algebraic, P.D.E.-based or multi-block methods (Chapter 4), quadtree-octree based method (Chapter 5), advancing-front technique (Chapter 6), Delaunay type methods (Chapter 7), mesh generation methods for implicitly defined domains (Chapter 16) and other mesh generation techniques (Chapter 8) not covered by the previous cases. The next chapter (Chapter 9) deals with Delaunay-admissible curve or surface meshes and then discusses medial axis construction along with the various applications that can be envisaged based on this entity. Prior to a series of five chapters on lines, curves and surfaces, a short chapter concerns the metric aspects that are encountered in mesh generation activities (Chapter 10). As previously mentioned, Chapters 12 to 16 discuss curves and surfaces while Chapter 11 recalls the basic notions regarding differential geometry for curves and surfaces. One chapter presents various aspects about mesh modification tools (Chapter 17), then, two chapters focus on optimization issues (Chapter 18 for planar or volumic meshes and Chapter 19 for surface meshes). Basic notions about the finite element method are recalled in Chapter 20 before looking at a more advanced mesh generation problem, namely how to construct an adapted mesh (Chapters 21 and 22). Parallel aspects are discussed in Chapter 23. To conclude, an extensive index is given to help the readers to quickly locate the information they require.

Acknowledgements

This work was carried out at the Gamma project, INRIA-Rocquencourt and was contributed to by many people in many ways. The authors would like to thank some of these people here. Friends and colleagues at INRIA, Frédéric Hecht, Patrick Laug, Éric Saltel and Maryse Desnous (who helped us for most of the figures that illustrate the discussion) were a great source of support and provided much of the technical material and illustrations for this work. Their comments and suggestions contributed greatly to its technical content.

The authors are especially indebted to Frédéric Noël (Université Joseph Fourier and Laboratoire des Sols Solides et Structures, Grenoble) who took the responsibility for the chapters devoted to curve and surface definitions and meshing. Bruce Simpson (Waterloo University, Ontario, Canada) must be thanked for his generous and significant contribution to this book in checking the consistency of the discussion. Mark Loriot (Simulog, France) kindly accepted to help us with the part devoted to partitioning methods. Frédéric Eichelbrenner (ibid) helped us with surface meshing methods and Philippe Pébay (INSA, Lyon), through his PhD work, helped us for the chapter about Delaunay admissibility and medial axis construction. To end, we would like to thank Frédéric Cazals (INRIA, Sophia Antipolis) for his help on the fundamentals of data structures and basic algorithms.

In addition to these people, several contributors were always willing to help us on various topics : Laurent Francez (Simulog), Loïc Marechal (CNAM-INRIA), Éric Séveno (previously at INRIA), Rachid Ouatchaoui (EDF-INRIA) as well as Houman Borouchaki (TUT).

This book is a translation of "Maillages. Applications aux éléments finis", published by Hermès Science Publications, Paris, 1999. The translation was carried out by the two authors with the valuable help of Richard James whom we would like to thank here.

INRIA-Rocquencourt - 1999.

Symbols and Notations

Notations and abbreviations are generally explained when they occur in the text. However, the list provided below describes the most frequent notations, symbols and abbreviations.

Notations.

d	refers to the spatial dimension
$\mathbb{N}, \mathbb{R}$	set of integers, set of reals
Ω	refers to a closed geometric domain of $\mathbb{R}^d$
$\partial\Omega$	refers to the (discretized) boundary of Ω
$\Gamma(\Omega)$	refers to the boundary of Ω
Γ, Σ	refers to a curve, a surface
γ, σ	refers to the parametrization of a curve, a surface
$\mathcal{T}, \mathcal{T}_h, \mathcal{T}_r$	refers to a triangulation or a mesh
$\mathcal{V}$ or $\mathcal{S}$	refers to a set of vertices
$Const$	refers to a constraint (a set of entities)
$Conv(\mathcal{V})$	refers to the convex hull of $\mathcal{V}$
(Δ, H)	refers to a control space
K	refers to a mesh element
S_K, V_K	refers to the surface area, the volume of element K
$\mathcal{Q}_K$	shape quality of mesh element K
d_{AB}, $d(A,B)$	(Euclidean) distance between A and B
$\|\overrightarrow{PQ}\|$	Euclidean length of segment PQ
l_{AB}	(normalized) length of edge AB

Symbols.

∇	gradient operator
$\mathcal{H}$	Hessian tensor
$\|a\|$	absolute value
$\lfloor . \rfloor$	integer part or restriction
$\|\| . \|\|$	Euclidean length of a vector
$[a,b]$	a closed interval
$\langle u, v \rangle$	dot product of two vectors
$(. \wedge .)$	cross product of two vectors
${}^t u$	u transposed (also u^t)

Abbreviations.

BRep, F-Rep	Boundary Representation, Function Representation
C.A.D.	Computer Aided Design
C.S.G.	Constructive Solid Geometry
M.A.T	Medial Axis Transform
F.E.M.	Finite Element Method
P.D.E.	Partial Derivative Equation
NURBS	Non Uniform Rational B-Splines
LIFO	Last In First Out
FIFO	First In First Out
BST	Binary Search Tree
AVL	Adelson, Velskii and Landis tree

Finally, let us mention, *at this time*, two websites devoted to meshing technologies :

`http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html`
`http://www.andrew.cmu.edu/user/sowen/mesh.html`

Chapter 1

General definitions

Introduction

Before discussing the various methods used for mesh construction, optimization, modification and more globally for mesh manipulation (described in the following chapters), it seems important to clarify the terminology and to provide some basic definitions together with some notions of general interest. Thus, this chapter sets out some definitions and basic notions related to triangulation and meshing problems. First, we define the *covering-up* of a bounded domain, then we present the notion of a *triangulation* before introducing a particular triangulation, namely the well-known *Delaunay triangulation.*

A domain covering-up simply corresponds to the naive meaning of this word and the term may be taken at face value. On the other hand, a triangulation is a specific covering-up that has certain specific properties. Triangulation problems concern the construction, generally by means of simplicial elements, of a covering-up of the convex hull of a given set of points. Based on the space dimension, a triangulation consists of simplicial elements (triangles in two dimensions, tetrahedra in three dimensions) such that certain properties hold.

If, in addition to the set of vertices, the boundary of a domain (more precisely a discretization of this boundary whose vertices are in the above set) is specified or, simply if any set of required edges (faces) is provided, we meet a problem of *constrained triangulation.* In this case, the triangulation of the convex hull must contain these required items.

In contrast, the notion of a *mesh* may now be specified. Given a domain, namely defined by its boundary (actually a discretization of this boundary), the problem comes down to constructing a "triangulation" that accurately matches this precise domain. In a way, we are dealing with a constrained triangulation but, now, we no longer face a convex hull problem and, moreover, the mesh elements are not necessarily simplices.

After having established triangulation and mesh definitions, some other aspects are discussed, including element definition (as an element is the basic component of both a triangulation and a mesh), finite element definition as well as mesh data

structure definition which are the fundamental ingredients of any computational step with the finite element method (F.E.M.) and which are also useful in other types of problems. In addition, we introduce some definitions related to certain data structures which are widely used in mesh construction and mesh optimization processes. To conclude, we give some views on mesh quality and mesh optimality.

Obviously this chapter cannot claim to be exhaustive. While it includes many definitions and basic notions, others are not provided. However, specific ideas will be introduced and discussed as required throughout the book.

1.1 Covering-up and triangulation

Let $\mathcal{S}$ be a (finite) set of points in $\mathbb{R}^d$ ($d = 2$ or $d = 3$), the convex hull of $\mathcal{S}$, denoted as $Conv(S)$, defines a domain Ω in R^d. Let K be a simplex[1] (triangle or tetrahedron according to d, always considered as a connected and closed set). Then a covering-up $\mathcal{T}_r$ of Ω by means of such elements (simplices) corresponds to the following :

Definition 1.1 *$\mathcal{T}_r$ is a simplicial covering-up of Ω if the following conditions hold*

- $(H0)$ *The set of element vertices in $\mathcal{T}_r$ is exactly S.*
- $(H1)$ $\Omega = \overline{\bigcup_{K \in \mathcal{T}_r}{}^{\circ} K}$, *where K is a simplex.*
- $(H2)$ *The interior of every element K in $\mathcal{T}_r$ is non empty.*
- $(H3)$ *The intersection of the interior of two elements is an empty set.*

Here is a "natural" definition. With respect to condition $(H1)$, one can see that Ω is the open set corresponding to the domain that means, in particular, that $\overline{\Omega} = \bigcup_{K \in \mathcal{T}_r} K$. Condition $(H2)$ is not strictly necessary to define a covering-up, but it is nevertheless practical with respect to the context and, thus, will be assumed. Condition $(H3)$ means that element overlapping is proscribed.

[1] Let us briefly recall the definition of a d-simplex : we consider $d+1$ points $a_j = (a_{ij})_{i=1}^d \in \mathbb{R}^d, 1 \leq j \leq d+1$, not all in the same hyper-plane, *i.e.*, such that the matrix of order $d+1$:

$$\mathcal{A} = \begin{pmatrix} a_{11} & \cdots & a_{1,d+1} \\ \cdots & \cdots & \cdots \\ a_{d1} & \cdots & a_{d,d+1} \\ 1 & 1 & 1 \end{pmatrix},$$

is invertible.A d-simplex K whose vertices are the a_j is the convex hull of these points a_j. Every point x in $\mathbb{R}^d$, with Cartesian coordinates x_i is fully specified by the data of $d+1$ scalar values $\lambda_j = \lambda_j(x)$ that are solutions of the linear system :

$$\begin{cases} \sum_{j=1}^{d+1} a_{ij}\lambda_j = x_i \text{ with } \sum_{j=1}^{d+1} \lambda_j = 1, \end{cases}$$

whose matrix is $\mathcal{A}$. The $\lambda_j(x)$ are the *barycentric coordinates* of point x with respect to the points a_j.

Similarly, we will consider conforming coverings-up, referred to as triangulations.

Definition 1.2 *$\mathcal{T}_r$ is a conforming triangulation or simply a triangulation of Ω if $\mathcal{T}_r$ is a covering-up following Definition (1.1) and if, in addition, the following condition holds :*

- $(H4)$ *the intersection of two elements in $\mathcal{T}_r$ is either reduced to*
  - *the empty set or to*
  - *a vertex, an edge or a face (for $d = 3$).*

More generally, in d dimensions, such an intersection must be a k-face[2], for $k = -1, ..., d-1$, d being the spatial dimension.

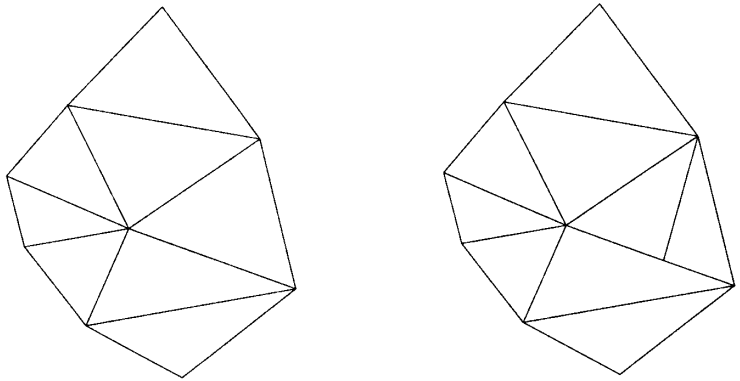

Figure 1.1: *Conformal triangles (left-hand side) and non conformal triangles (right-hand side). Note the vertex located on one edge in this case.*

Remark 1.1 *For the moment, we are not concerned with the existence of such a triangulation for a given set of points. Nevertheless, a theorem of existence will be provided below and, based on some restrictive assumptions, the case of a Delaunay triangulation will be seen.*

Euler characteristics. The Euler formula, and its extensions, the Dehn-Sommerville relationships, relate the number of k-faces ($k = 0, ..., d-1$) in a triangulation of Ω. Such formula can be used to check the topological validity of a given mesh or also for other purposes (such as the determination of the genus of a surface).

[2] A (-1)-face is the empty set, a 0-face is a vertex, a 1-face is an edge, a k-face is in fact a k-simplex with $k < d$, d being the spatial dimension.

Definition 1.3 *The Euler characteristics of a triangulation $\mathcal{T}_r$, is the alterned summation :*

$$\chi = \sum_{k=0}^{d} (-1)^k n_k \,, \tag{1.1}$$

where $n_k, k = 0, .., d$ denotes the number of the k-faces in the triangulation.

When the triangulation is homeomorphic to the topological ball, its characteristic is 1. If the triangulation is homeomorphic to the topological sphere, its Euler characteristic is $1 + (-1)^d$.

In two dimensions, the following relation holds :

$$ns - na + ne = 2 - c \,,$$

where ns, na and ne are respectively the number of vertices, edges and triangles in the triangulation, c corresponds to the number of connected components of the boundary of Ω. More precisely, if the triangulation includes no holes, then $ns - na + ne = 1$.

In three dimensions, the above formula becomes :

$$ns - na + nf - ne = 2 - 2g \,,$$

where g stands for the *genus* of the surface (*i.e.,* the number of holes). Thus, a triangulation of a closed surface is such that $ns - na + nf = 2$.

Delaunay triangulation. Among the different possible types of triangulations, the Delaunay triangulation is of great interest. Let us recall that $\mathcal{S}$ is a set (or a cloud) of points and that Ω is $Conv(S)$, the convex hull of $\mathcal{S}$.

Definition 1.4 *$\mathcal{T}_r$ is the* Delaunay triangulation *of Ω if the open (balls) discs circumscribed associated with its elements contain no vertex of $\mathcal{S}$.*

This criterion, the so-called *empty sphere criterion* or *Delaunay criterion*, means that the open balls associated with the elements do not contain any vertex (while the closed balls contain the vertices of the element under consideration only). That is a characterization of the Delaunay triangulation. The Delaunay criterion leads to several other characteristics of any Delaunay triangulation. Figure 1.2 shows an example of an element K which does not meet the Delaunay criterion.

A basic theoretical issue follows :

Theorem 1.1 *The Delaunay triangulation of a set of points exists and, moreover, it is unique.*

The proof is evident by involving the duality with the Voronoï diagram associated with the set of points (cf. Chapter 7). The existence is then immediate and

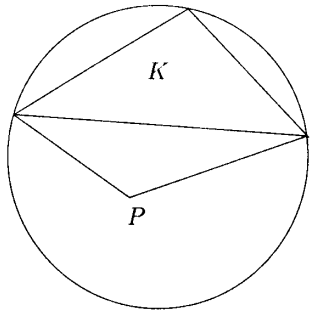

Figure 1.2: *The empty sphere criterion is violated, the disc of K encloses the point P. Similarly, the circumdisc of the triangle with vertex P includes the vertex of triangle K opposite the common edge (the criterion is symmetric for any pair of adjacent elements).*

the uniqueness is clear since the points are in general position[3] if one wishes to have a simplicial triangulation. Otherwise, the following remark holds.

Remark 1.2 *In the case of more than three co-circular (four co-spherical) points, a circle (sphere) exists to which these points belong. In such a case, the Delaunay triangulation exists but contains elements other than simplices. Indeed, polygons (polyhedra) may exist.*

Hence, the uniqueness holds if non simplicial elements are allowed while if the latter are subdivided by means of simplices, several solutions can be found. Nevertheless, while it may be excessive, we will continue to speak of **the** Delaunay triangulation by observing that all any partitions of a non simplicial element are equivalent after swapping a k-face (as we shall see below).

A brief digression. The notion of a Voronoï diagram (though it had yet to be called as such !) first appeared in the work of the French philosopher R. Descartes (1596-1650) who introduced this notion in 1644 in his *Principia Philosophiae*, which aimed to give a mathematical description of the arrangement of matter in the solar system. In 1850, G. Dirichlet (1805-1859) studied this idea in two and three dimensions and this diagram came to be called the *Dirichlet tessellation* [Dirichlet-1850]. However, its definitive name came after M.G. Voronoï (1868-1908), who generalized these results in d dimensions [Voronoï-1908].

Nature provides numerous examples of arrangements and quasi-regular paving which bear a strange resemblance to Voronoï diagrams.

[3] A set of points is said to be in general position if there is no configuration of more than three points that are co-circular (more than four co-spherical points).

Figures 1.3 and 1.4 illustrate some of these typical arrangements.

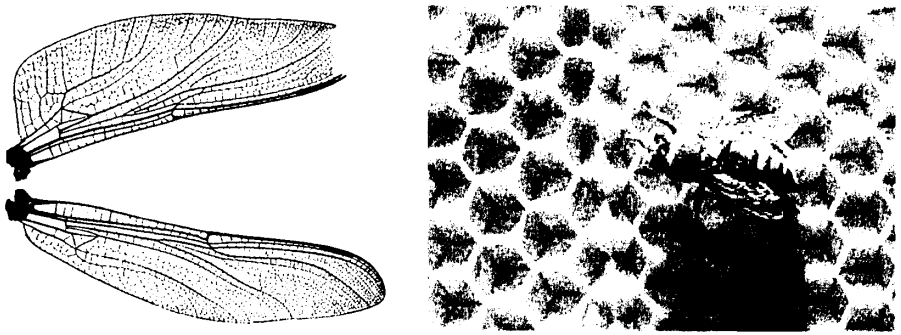

Figure 1.3: *The wings of a dragonfly (doc. A. LeBéon) show an alveolar structure apparently close to a Voronoï diagram (left-hand side). One of the more representative examples of regular paving (consisting of hexagonal cells) is that of a bee nest (right-hand side).*

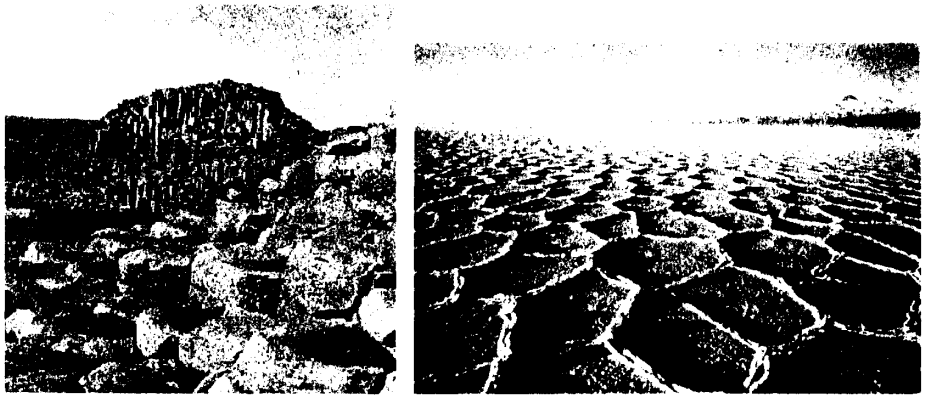

Figure 1.4: *Two examples of natural arrangements. Left-hand side : the basaltic rock site of the Giant's Causeway, Co Antrim, Northern Ireland (photo credit : John Hinde Ltd.). Right-hand side, desert region of Atacama in the north of Chile, the drying earth forms patterns close to Voronoï cells.*

Constrained triangulation. Provided a set of points and, in addition, a set of edges and faces (in three dimensions) , an important problem is to ensure the existence of these edges (in two dimensions) or these edges and faces (in three dimensions) in a triangulation. In the following, *Const* denotes a set of such entities.

Definition 1.5 *$\mathcal{T}_r$ is a constrained triangulation of Ω for Const if the members of Const are entities of $\mathcal{T}_r$.*

In particular, a constrained triangulation[4] can satisfy the Delaunay criterion locally, except in some neighborhood of the constraints.

Remark 1.3 *As above, provided a set of points and a constraint, we are not concerned here with the existence of a solution triangulation.*

1.2 Mesh

Now we turn to a different problem. Let Ω be a closed bounded domain in R^2 or R^3. The question is how to construct a conforming triangulation of this domain. Such a triangulation will be referred to as a *mesh* of Ω and will be denoted by $\mathcal{T}_r$ or $\mathcal{T}_h$ for reasons that will be made clear in the following. Thus,

Definition 1.6 *$\mathcal{T}_h$ is a mesh of Ω if*

- $(H1)$ $\Omega = \overline{\bigcup_{K \in \mathcal{T}_h} K}^{\,o}$.
- $(H2)$ *The interior of every element K in $\mathcal{T}_h$ is non empty.*
- $(H3)$ *The intersection of the interior of two elements is empty.*

Condition $(H2)$ is clearly not verified for a beam element for instance. Condition $(H3)$ avoids element overlapping. In contrast to the definition of a triangulation, Condition $(H0)$ is no longer assumed, which means that the vertices are not, in general, given *a priori* (see hereafter) and, in $(H1)$, the K's are not necessarily simplices.

Most computational schemes using a mesh as a spatial support assume that this mesh is conforming (although, this property is not strictly necessary for some solution methods).

Definition 1.7 *$\mathcal{T}_h$ is a conformal mesh of Ω if*

- *Definition (1.6) holds and*
- $(H4)$ *the intersection of two elements in $\mathcal{T}_h$ is either,*
  - *the empty set, a vertex, an edge or a face (when $d = 3$).*

Clearly, the set of definitions related to a triangulation is met again. The fundamental difference between a triangulation and a mesh is that a triangulation is a covering-up of the convex hull of a set of points (the latter being given) which, in general, is composed of simplicial elements while a mesh is a covering-up of a

[4] Whereas a *constrained Delaunay triangulation* (for example in two dimensions) is a triangulation which satisfies the empty sphere criterion, where a ball can contain a point in the case where the latter is not seen, due to a constrained edge, by all the vertices of the considered element. In other words, a constrained entity exists which separates the above points and the others.

given domain defined, in most of the applications, via a given discretization of its boundary, this covering-up being composed of elements that could be other than simplices. On the other hand, at least two new problems occur. These are related to :

- the respect or *enforcement*, in some sense, of the boundary of the domain so that the triangulation is a constrained triangulation,
- the necessity of *constructing* the set of points which will form the vertices of the mesh. Usually the boundary points of the given boundary discretization are given as sole input and field points must be explicitly created.

Remark 1.4 *For a given domain (a boundary discretization) the existence of a mesh conforming to this discretization holds in two dimensions but is still, at least from a theoretical point of view, an open and delicate question in three dimensions. In this respect, there is a series of simple counter-examples (see below).*

Remark 1.5 *In the finite element method, the meshes*[5] *are generally denoted by* $\mathcal{T}_h$*, where the index* h *of the notation refers to the diameters of the elements in the mesh, these quantities being used in error majoration theorems*

As previously mentioned, a mesh can be composed of elements of different geometric natures. A mesh consists of a finite number of segments in one dimension, segments, triangles and quadrilaterals (quad in short) in two dimensions and the above elements, tetrahedra (tet), pentahedra and hexahedra (hex) in three dimensions. The elements constituting the mesh must generally satisfy some specific properties according to the application involved (see the next section).

Meshes can be classified into three main classes according to their *connectivity*.

Definition 1.8 *The connectivity of a mesh is the definition of the connection between its vertices.*

Then, following this definition

Definition 1.9 *A mesh is called structured if its connectivity is of the finite difference type.*

Such a mesh can be termed as a *grid*[6]. In two dimensions, a grid element is a quadrilateral[7] while, in three dimensions, a grid consists of hexahedra. The connectivity between nodes is of the type (i, j, k), *i.e.*, assuming the indices of a given node, the node with indices (i, j, k) has the node with indices $((i-1), j, k)$ as its "left" neighbor and that with indices $((i+1), j, k)$ as its "right" neighbor; this kind of mesh is convenient for geometries for which such properties are suitable, *i.e.*, for generalized quadrilateral or hexahedral configurations.

[5] It should be noted that people with a finite element background use the term triangulation and use the term mesh synonymously.

[6] Note that some authors use the term "grid" to refer to any kind of mesh whatever its type of connectivity may be.

[7] Also simply referred to as a quad.

Remark 1.6 *Peculiar meshes other than quad or hex meshes could have a structured connectivity. For instance, one can consider a classical grid of quads where each of them are subdivided into two triangles using the same subdivision pattern.*

Definition 1.10 *A mesh is called unstructured if its connectivity is of any other type.*

Such a mesh is usually composed of triangles (tetrahedra) but can also be a set of quadrilaterals (hexahedra) or, more generally, a combination of elements of a different geometric nature. Note that quad or hex unstructured meshes are such that the internal vertices may be shared by more than 4 (8) elements (unlike the case of structured meshes).

For completeness, we introduce two more definitions.

Definition 1.11 *A mesh is said to be* mixed *if it includes some elements of a different geometric nature.*

Definition 1.12 *A mesh is said to be* hybrid *if it includes some elements with a different spatial dimension.*

A mixed mesh, in two dimensions, is composed of triangles and quads. A hybrid mesh, again in two dimensions, is clearly a mixed mesh but, for instance, includes some triangles together with some segments.

To complete this classification, a mesh may be *manifold* or not. This point concerns the surface meshes.

Definition 1.13 *A surface mesh is called manifold if its internal edges are shared by exactly two elements (one element in the case of a boundary edge for an open surface).*

Otherwise, the surface mesh is said to be *non-manifold*. This is the case of surface meshes which include stiffeners or which have two or more connected components.

1.3 Mesh element

The elements are the basic components of a mesh. An element is defined by its geometric nature (triangle, quadrilateral, etc.) and a list of vertices. This list, enriched with some conventions (see hereafter), allows the complete definition of an element, including the definition of its edges and faces (in three dimensions).

Definition 1.14 *The connectivity of a mesh element is the definition of the connections between the vertices at the element level.*

This connectivity, the local equivalent of the mesh connectivity (see a previous definition), makes the description of the *topology* of the element possible.

Definition 1.15 *The topology of a mesh element is the definition of this element in terms of its faces and edges, these last two being defined in terms of the element's vertices.*

Triangle connectivity and topology The (local) numbering of vertices and edges is pre-defined in such a way that some properties are implicitly induced[8]. This definition is only a convenient convention resulting in implicit properties. In particular, the oriented numbering of the vertices enables us to compute the surface area of a triangle with a positive, or directional, sense. It also allows us to evaluate directional normals for each edge.

In the case of a triangle (in $\mathbb{R}^2$) with connectivity $[1,2,3]$, the first vertex (1) having been chosen, the numbering of the others is deduced counterclockwise. Then the topology can now be well defined by means of the edge definition :

- edge [1] runs from vertex (1) to vertex (2),
- edge [2] : (2) $\rightarrow$ (3),
- edge [3] : (3) $\rightarrow$ (1),

or

- edge [1] is opposite vertex (1), it runs from vertex (2) to vertex (3),
- edge [2] : (3) $\rightarrow$ (1),
- edge [3] : (1) $\rightarrow$ (2).

The question is to define the topology, one or the other, and to conform to this definition. Such an implicit definition will be a source of simplicity hereafter (avoiding explicit definitions at the element level during the computational step as mentioned earlier).

Usual element connectivities and topologies Elements other than triangles are now defined in terms of the two above definitions.

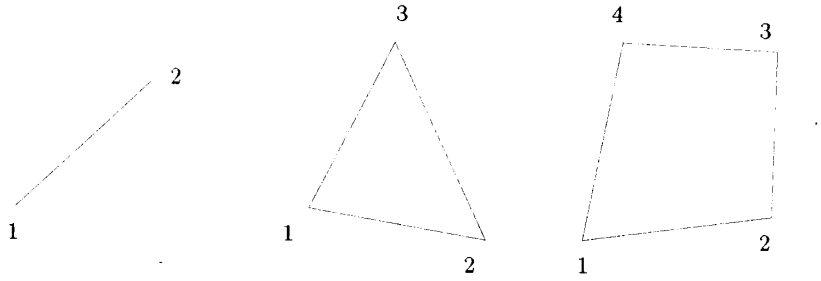

Figure 1.5: *Segment, triangle and quadrilateral (local vertex numbering, given the first vertex index).*

- The segment : $[1,2]$, (1) $\rightarrow$ (2).

[8] Given a vertex numbering (index) based on an implicit definition results in implicit definitions for both the edges and the faces, thus avoiding an explicit definition of these entities at the element level, which would be tedious and memory consuming.

- The quadrilateral : $[1,2,3,4]$ with a numbering as for the triangle,

edge [1] : (1) $\rightarrow$ (2) edge [2] : (2) $\rightarrow$ (3)
edge [3] : (3) $\rightarrow$ (4) edge [4] : (4) $\rightarrow$ (1)

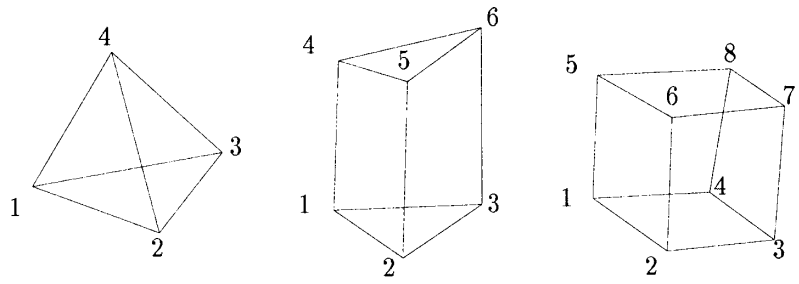

Figure 1.6: *Tetrahedron, pentahedron and hexahedron.*

- The tetrahedron[9]: $[1,2,3,4]$ with $(\vec{12}, \vec{13}, \vec{14})$ assumed to be positive with, for the edges :

edge [1] : (1) $\rightarrow$ (2) edge [2] : (2) $\rightarrow$ (3) edge [3] : (3) $\rightarrow$ (1)
edge [4] : (1) $\rightarrow$ (4) edge [5] : (2) $\rightarrow$ (4) edge [6] : (3) $\rightarrow$ (4)

and, for the faces :

face [1] : (1) (3) (2) face [2] : (1) (4) (3)
face [3] : (1) (2) (4) face [4] : (2) (3) (4)

- The pentahedron: $[1,2,3,4,5,6]$ with $(\vec{12}, \vec{13}, \vec{14})$ assumed to be positive, with, for the edges :

edge [1] : (1) $\rightarrow$ (2) edge [2] : (2) $\rightarrow$ (3) edge [3] : (3) $\rightarrow$ (1)
edge [4] : (1) $\rightarrow$ (4) edge [5] : (2) $\rightarrow$ (5) edge [6] : (3) $\rightarrow$ (6)
edge [7] : (4) $\rightarrow$ (5) edge [8] : (5) $\rightarrow$ (6) edge [9] : (6) $\rightarrow$ (4)

and, for the faces :

face [1] : (1) (3) (2) face [2] : (1) (4) (6) (3)
face [3] : (1) (2) (5) (4) face [4] : (4) (5) (6)
face [5] : (2) (3) (5) (6)

- The hexahedron: $[1,2,3,4,5,6,7,8]$, $(\vec{12}, \vec{14}, \vec{15})$ assumed to be positive, with

edge [1] : (1) $\rightarrow$ (2) edge [2] : (2) $\rightarrow$ (3) edge [3] : (3) $\rightarrow$ (4)
edge [4] : (4) $\rightarrow$ (1) edge [5] : (1) $\rightarrow$ (5) edge [6] : (2) $\rightarrow$ (6)
edge [7] : (3) $\rightarrow$ (7) edge [8] : (4) $\rightarrow$ (8) edge [9] : (5) $\rightarrow$ (6)
edge [10] : (6) $\rightarrow$ (7) edge [11] : (7) $\rightarrow$ (8) edge [12] : (8) $\rightarrow$ (5)

[9] Similarly to the triangle, an alternative definition also suits well where face [i] is opposite vertex (i). Actually, the latter convention leads to greater simplicity.

and, for the faces :

face [1] : (1) (4) (3) (2) face [2] : (1) (5) (8) (4)
face [3] : (1) (2) (6) (5) face [4] : (5) (6) (7) (8)
face [5] : (2) (3) (7) (6) face [6] : (3) (4) (8) (7)

Other types such as pyramid may be defined. Actually, this type of element allows some flexibility for mixed meshes. This is the case when structured meshes (*i.e.*, hex meshes) must be combined with unstructured meshes composed by tets.

1.4 Finite element mesh

So far, we have considered meshes as geometric entities. Now we turn to the notion of *finite element meshes* since we are mainly interested in finite element computation as an application of great importance whose spatial support is a mesh enriched by some ingredients. As will be discussed in Chapter 20, finite elements will be constructed based on the mesh element. To this end, it will be necessary to properly define the nodes, degrees of freedom, interpolation schemes, etc. so as to compute what is required (stiffness matrix, right-hand side, etc.) in order to obtain the solution to the problem under investigation.

Let us briefly recall for those not so familiar with a finite element style computation, that the classical scheme of such calculus includes the following steps (for a simple case) :

- a mesh construction step whose purpose is to complete the list of the (geometric) elements of this mesh,
- an interpolation step which, from the mesh elements, constructs the finite elements,
- a matrix and right-hand side construction step to complete the system corresponding to the discretization of the initial equations,
- a solution step which computes the solution of the above system.

This being clear, we can proceed by giving some definitions related to the finite element meshes.

Node definition. The finite elements will be associated with the mesh elements at the computational step. For the moment, a finite element is a geometric element supplied with a list of *nodes*.

Definition 1.16 *A node is a point supporting one or several unknowns or degrees of freedom (d.o.f.).*

The nodes are defined according to the interpolation used in the computation. For a given geometric element, several finite elements may be exhibited as a function of the interpolation step. The "simplest" finite element is the Lagrange P^1 finite element whose nodes are the element vertices. A Lagrange P^2 finite element

includes as nodes the element's vertices and a point on each of its edges (in the usual case, these nodes are the edge midpoints). Other finite elements may involve several nodes for each edge, nodes located on faces or inside the element while the element vertices may be nodes or not (see Chapter 20 where various examples of finite elements are given).

Once the node location has been established, it is a case of defining a local numbering for the nodes of the finite element. This task is trivial when the only element vertices are the nodes as the node numbering follows the vertex numbering. If the nodes are defined elsewhere, the local numbering must be well defined (implicitly as for the vertex or explicitly in some cases where an implicit definition is not possible, for instance when the number of nodes varies from one edge to the other as it does for some finite elements). If we consider a Lagrangian P^2 triangle, it is natural to define the three first nodes as the three vertices and then to define as fourth node the node located along the first edge of the triangle (and so on for the other nodes). See also Chapter 20 and Chapter 17 for node (re)numbering issues.

Physical attributes. At the solution step, the finite elements are the support of various computations or specific treatments. The mesh, through its elements must contain information making it possible the selection of a set of elements, a set of faces, set of edges or a set of nodes, thus making possible any processing concerning these entities, in particular to take into account the loads, flows, pressures, boundary conditions, graphic requirements at the time the solutions must be displayed, etc., related to the problem considered. This will allow to carry out the adequate assignment (*i.e.*, associate the relevant value with such or such an item) or the proper computation (such as the computation of the useful integrals over such or such entities).

It is then convenient to associate a *physical attribute* with all the mesh entities (elements, faces, edges, nodes). The way in which this task can be done is not unique and, at the computer level, can be implemented in various ways.

Geometric attributes. For similar reasons, a geometric attribute (provided in some way) proves to be useful for some operations such as the proper definition of the nodes in the case of a finite element other than a P^1 element. Thus, it is also convenient to associate a geometric attribute with all the element entities (faces, edges, nodes).

1.5 Mesh data structures

In general, a data structure is a way of organizing the information (values) needed in a given process. In this respect, a mesh data structure is a way to store all the values that will be useful for a further processing. Following this idea, two categories of structures can be distinguished. One is used when constructing the mesh. In other words, such a structure, referred to as the *internal mesh data structure* is the organization of the values describing the mesh which are required

inside the mesh generation method chosen for this task. On the other hand, the structure referred to as the *output mesh data structure* is the mesh organization used at the computational step, *i.e.*, outside the mesher. Thus the values stored in it as well as the way in which they are organized can be slightly different from those in the previous context. The possible differences between the two mesh structures depend both on the type of the mesh generation method and the solution method that are used. However, for the output structure, it could be desirable to have a "universal" structure or, at least, a more generic structure (see below).

1.5.1 Internal mesh data structure

The key-idea is to define as simple a structure as possible which is well suited to the problem. Various reasons can justify this policy. Among them, a given mesh generation method, due to the algorithm which is used and the nature of the meshes it is capable of processing, may require a given data organization that is different from another method. Hence, a "universal" structure is not, *a priori*, a solution (as it would be unnecessarily complex in certain cases). Thus, for a given method, one has to find what is needed specifically and what is not strictly required. In this respect, efficiency as well as memory occupation reasons can be invoqued to justify such or such a choice.

The internal mesh structure is only used within the algorithm and, when the mesh is created, this structure is transformed so as to complete the output data structure which is the natural (and unique) interface with the computational step.

Some values and data items are specific to one structure or the other. Some others must be included in both. The point is to make this requirement precise and to define the structure accordingly. In the next chapter, we will try to give some indication about what such an internal structure could be.

1.5.2 Output mesh data structure

Defining a suitable, general purpose and reasonably simple output data structure for mesh storage is not a trivial task. A number of issues must be addressed in order to make various tasks possible. These include :

- the proper definition of the nodes (when defining the finite elements at the so-called interpolation step),
- the definition of loads, the computation of the relevant matrices and right-hand sides,
- the solution of the resulting system(s),
- the visualization of the results,
- and many others allowing for other types of processing depending on the nature of the problem.

In this respect, it is clear that a classic computational scheme involving a mesh generation step, an interpolation step, the computation of the system to be solved and the solution step is less demanding than an adaptive scheme requiring a loop of such operations where, for instance, the mesh (and the geometry itself, in some cases) is modified at each iteration step of the loop.

Thus, a mesh data structure must be defined in such a way as provide easy access to :

- the vertex coordinates,
- the element vertices,
- the physical attributes of the mesh entities,
- the geometrical attributes of these entities,

in order to make the previous computational requirements possible.

These basic principles being stated, it is beyond the scope of this book to discuss further what a data structure could be. Nevertheless, it must be observed that, at this time, existing *norms* do not give a satisfactory answer to the question of knowing what a mesh data structure should be. However, one can find in [George,Borouchaki-1997], Chapter 10, a proposal for such a structure. On the other hand, in Chapter 2, one can gain some indications about data structures when seen from a more abstract point of view.

1.6 Control space

It is useful to introduce the notion of a *control space* for various purposes (as will be seen in the chapters devoted to the construction of adapted meshes). Indeed, this space serves to determine the current background.

An *ideal* control space is simply the specification of a function $H(x, y)$ defined at any point $P(x, y)$ of $\mathbb{R}^2$ (for the sake of simplicity, we consider a two-dimensional problem). In other words, the function is defined analytically and specifies the size and the directional features that must be conformed to by the mesh elements anywhere in the space. However, from a practical point of view, the control spaces will not be ideal in the sense that the above function H will be provided only in a discrete manner. Indeed, formally speaking, a control space can be defined as follows, [George-1991] :

Definition 1.17 *(Δ, H) is a* **control space** *for the mesh T of a given domain Ω if :*

- *$\Omega \subseteq \Delta$ where Δ covers the domain Ω,*
- *a function $H(P, \vec{d})$ is associated with every point $P \in \Delta$, where $\vec{d}$ is the direction of the disc S^1 (or the sphere S^2 in three dimensions):*

$$H(P, \vec{d}) : \Delta \times S^1 \rightarrow R.$$

Thus a control space includes two related ingredients. First a covering triangulation Δ, is defined (by means of such or such a method). Second a function H, whose support is a covering triangulation Δ, is posed. This pair allows for the specification of some properties or criteria to which the elements of the mesh should conform.

In terms of geometry, Δ is an arbitrary covering triangulation. For example, it can be one of the following types :

type 1 : a regular partition, such as a finite difference type,

type 2 : a quadtree or octree-based partition (see Chapters 2 and 5),

type 3 : an arbitrary user-constructed mesh or the current mesh, for instance in an iterative process, the last mesh which then acts as a background mesh.

In addition to this partitioning aspect, (Δ, H) contains, by means of the function H, the global information related to different aspects. One could be the geometry of the domain, the other could be the physics of the problem. These values allow us to determine whether the mesh T, under construction, conforms to the function everywhere.

To construct H, one can consider one of the following approaches :

- compute, from the data, the local stepsizes h (the value h being the desired distance between two points) related to the given points. A generalized interpolation then enables us to obtain H (this process is purely geometric in the sense that it relies on geometric data properties: boundary edge lengths, etc.);
- manually define the value of H for every element of Δ. A desired size is then given everywhere in the space for *isotropic* control, or the desired sizes according to specific directions are given for *anisotropic* control;
- manually specify H by giving its value for each element of the covering triangulation constructed for this purpose (we return here to a *type 3* space as introduced above);
- use the cell sizes (in the above *type 2* case), where this size is used to encode the value of H. This then leads to the construction of the (Δ, H) space so as to satisfy this requirement;
- define H from an a posteriori error estimate. We are then in an iterative adaptive process. A mesh T is constructed, the corresponding solution is computed and the error estimate analyzes this solution so as to complete H. Then Δ is set to T and the pair (T, H) forms the control space used to govern the new mesh construction (cf. Chapter 21).

For each of the different types, this definition results in one or the other control space types. In what follows, we will show how to create the internal points in accordance with the specifications contained in this space (for a mesh construction method) or to optimize a mesh (for a mesh optimization method).

Definition 1.18 *When the geometric locus of $P + H(P, \vec{d})$ is a circle (a sphere in three dimensions), with P in Δ and $\vec{d}$ varying, the control space is isotropic. When this locus is an ellipse (ellipsoid), the control space is anisotropic.*

Only these two cases will be discussed hereafter, leading to the definition of the metric maps which are used to govern the mesh construction process.

Remark 1.7 *The popular notion of a* background mesh, *extensively used for instance in adaptive mesh construction processes, is nothing more than a control space. In this case the covering triangulation is the mesh of the previous iteration step while the function results from the analysis of the current solution by means of an* a posteriori *error estimate.*

Remark 1.8 *The above function H is nothing other than a metric as will be seen later. In this context the discrete meaning of $H(P, \vec{d})$ will be seen as a $d \times d$ symmetric definite matrix denoted, at point P, by $\mathcal{M}(P)$.*

Note that some authors, while using different approaches, return to this notion of a control space. For instance, source points can be introduced and this information is used to complete an equivalent of the above H function.

1.7 Neighborhood space

Close to the previous idea of control space, the notion of a *neighborhood space* can be introduced. Such a space acts to help any neighboring information or queries. For instance, this is a way to facilitate the search for all the vertices located in some regions as such a query could be of great interest for various purposes.

The neighborhood space structure is similar to that of a control space. It includes two components, a spatial support together with some information related to this support. The spatial support spatial may be as above with, however, a special interest for the simple type of covering-up (grid, tree structure such as quadtree in two dimensions) in which localization problems are evident. Along with this support, the space includes some information of a topological nature. For instance, for a grid, one encodes in its cells the fact that a point, an edge, etc., exists or not. Thus, when such a cell is queried, we know whether a point, an edge, etc., exists within a certain neighborhood. This is a simple example of what could be encoded in this kind of space.

Developing a binary tree (for instance, an alternate tree) is also a way of giving access to the neighboring entities, in some sense, of a given entity.

1.8 Mesh quality and mesh optimality

The purpose of constructing a mesh is not simply a question of creating a mesh (a covering-up) of the domain of interest but to obtain a good quality mesh. Therefore, the immediate question is : " what is a good quality mesh ?" or, similarly : " how can mesh optimality be defined ?".

In response to these questions, the literature contains a number of tentative definitions which are more or less naive (and, in some cases, fanciful to the point of eccentricity !). In this respect, let us mention some widely held ideas. A nice looking mesh is a good mesh. However, this raises the problem of what "nice looking" actually means. A good mesh is a mesh whose elements have the same size and conform to a nice aspect ratio. In addition, some peremptory assertions can be found such as "a Delaunay mesh is optimal". The list goes on, but our aim here is to find some reasonable criteria that are well suited to qualifying a mesh.

A preliminary and obvious remark helps us in this discussion. The mesh is constructed with a precise purpose : solve a given problem. Therefore, the true quality or optimality problem is related to the solution that can be computed with the mesh as a support. In this respect, it makes sense to claim that the mesh quality is good if the resulting solution quality is good. As a consequence and following the same approach, optimality is obtained if the mesh size is minimal (in some sense, for instance, if the number of elements or, more precisely, the number of nodes and vertices is minimal) resulting in a minimal cost when computing the problem solution. This characterization is then related to the nature of the problem and, moreover, implicitly assumes that a suitable way to qualify the solution quality is given.

In a numerical simulation by means of finite elements, an error estimate is the sole judge. A nice mesh is one which leads to the best possible numerical accuracy. For other cases, for instance in a surface visualization problem based on a mesh of the surface of interest, the quality must be measured with regard to the approximation quality or the graphic aspect of the surface mesh with respect to the real surface. For other problems, the quality aspect may be related to different objectives.

At first, this computational process is based on an approximation of domain Ω where the problem is formulated. Thus, a first condition regarding the mesh quality is to make this approximation precise. In fact, we want to solve the given problem in domain Ω and not in an approximate domain different from the real domain. This being established (that results in conditions about the boundary elements of the mesh leading to a good boundary approximation), the mesh quality is related to the solution and thus to the nature of the problem under investigation. Using an error estimate enables us to see the quality of the mesh and, more precisely, to know if its elements conform to a desirable size (a nice directional aspect).

When such an appreciation is not available (*i.e.,* no error estimate is used), one can only judge in advance the mesh quality notion and to translate this point in terms of a quality function about the mesh elements. For a problem of an isotropic nature, it seems attractive to consider the equilateral triangle as optimal while for a quad the square is *a priori* optimal. When no sizing information (about the element size or the desired edge length) is provided, the only thing that can be envisaged is to take advantage of the boundary (assumed to be correct) and to complete regular elements whose size follows at best the size of the boundary items (edges and faces). A fine discretization in a boundary region results in small elements while a coarser discretization may lead to larger elements. Then, these sizing features will be used to find a reasonable size value for the elements located at some distance from the domain boundaries. In particular, a progressive gradation of the sizes is often a desirable feature when considering the elements in a given neighborhood.

When a size map (control space) is provided, it is clear that a unit length in the metric associated with this control space is the targeted value. This leads us to the notion of a unit mesh which will be the good mesh we like to have (after some additional remarks).

Definition 1.19 *A* unit *mesh is a mesh with unit edge lengths.*

Computing the edge length in accordance with the metric as defined in the control space provides the expected actual values. This point will be discussed later and, for the moment, it is enough to say that, in an isotropic context, a unit length in a metric map specifying a value h actually gives a length h in the usual sense and that the same is true in an anisotropic context.

Remark 1.9 *A triangle with unit edge lengths is necessarily a good quality triangle. This is not the case of a tetrahedron that may have an inconsistent volume (quite small) while its edges are (about) unit edges.*

After this remark about simplicial elements and due to the other various geometries that can be envisaged, edge length control must be coupled with other criteria to make sure that an optimal quality is obtained.

Notice that the unit value is related to the underlying metric map and that the latter could be the combination of a map of a geometric nature (due to the domain geometry) with one or several other maps of a physical nature depending on the behavior of the problem under investigation.

To conclude this brief overview, let us indicate that numerous detailed discussions will be found in the following chapters to clarify the notions introduced in this section and to indicate how to compute a length in a given metric map.