Chapter 21

Mesh generation and adaptation (h-methods)

Introduction

In this chapter, we consider mesh construction from the perspective of a finite element style computation. The objective is to introduce some mesh generation methods or mesh modification methods resulting in a mesh that conforms to some pre-specified requirements in terms of sizes (isotropic problem) or in terms of sizes and directional properties (anisotropic problem). The basic principle for adapted mesh construction is to collect information about the sizes, the directions and the related sizes using an adequate data structure, a *control space* (or a background mesh), and then to use this information to construct a mesh conforming (as far as possible) to these specifications.

As pointed out in Chapter 20 for finite element convergence and accuracy, theoretical error estimates involve the parameter h, the size of the mesh elements. Thus, at least for isotropic situations, the necessity of adapting the h's in the mesh is rather natural[1]. On the other hand, anisotropic cases are not so trivial. Actually, numerical experiments (mainly in two dimensions) resulting in nice solutions could lead us to think that h-adaptation is well suited to the problem, and yet theoretical proofs are not fully available at this time (except for simple problems).

Hence, we discuss a mesh adaptation method related to the element size (direction), in other words a *h-method.* This type of method is, as will be seen, the basic ingredient that makes it possible to compute the solution by means of adaptive solution methods. In brief, such a method is an iterative process where, at each iteration step (or after a small number of steps), an adapted mesh is constructed which is used to calculate the solution. Then this solution is analyzed in order to decide whether the iterations should continue or not.

[1] Intuitively, adapting the element sizes, according to the problem, comes down to constructing elements which are smaller or larger in some regions, the size h then being changed from one region to another (problems involving mechanics or fluids, etc.) or to adjust this h while keeping it as constant as possible (for wave propagation problems, for example).

Several categories of h-adaptive methods may be considered. One class of methods is based on the local modification of the current mesh so as to fit the sizing (directional) requirements. Another class of methods involves reconstructing the whole mesh of the domain from its boundary discretization while conforming to the sizing (directional) requirements. On the other hand, one can find adaptive methods that belong to neither of these above classes. For instance, hierarchic methods, multigrid type methods and non conforming or overlapping methods are examples of such approaches (and are not covered in this book). Finally, adaptation methods by means of degree adaptation also exist, the so-called *p-methods* which are discussed in the next chapter.

In what follows we discuss the first two types of methods with special emphasis on the global approach where the mesh is entirely recreated. Indeed, issues discussed throughout the previous chapters are now reviewed with the present objective in mind. This means that most of the aspects covered in this chapter have been already discussed to some extent in various parts of the book[2]. However, to keep the chapter as self-contained as possible, some notions will be recalled within the context of a h-adaptive mesh generation method.

★
★ ★

To this end, this chapter first recalls what a control space is and how it can be used to give the information that is required to control the mesh generation process. In this respect, localization as well as interpolation problems are discussed. Then we turn to h-adaptation by means of the local modification of a given mesh. Afterwards, h-adaptation based on the whole construction of a mesh is presented. Finally, computational schemes suitable for adaptive methods are given, together with details about the various steps involved in these schemes. The metric aspect (*i.e.*, the error estimate problem) is assumed to be known and thus will only be touched upon briefly.

21.1 Control space (background mesh)

A *control space* (Chapter 1) is a structure whose purpose is to govern a mesh modification or a mesh generation process by providing the required information about the adaptation. Such information may be specified in various ways. Among these, and based on the adaptive strategy, one can find element refinement (coarsening) demands, desired sizes (or densities) or desired directions and related directional sizes.

A demand at the element level mainly corresponds to an isotropic adaptation problem addressed by means of local mesh modifications. On the other hand, isotropic or anisotropic specifications expressed in terms of sizes, densities, directions, etc., are mostly related to an adaptation problem based on complete reconstruction of the mesh.

[2]Therefore, in principle, it is just necessary to dip into those parts of the book where the necessary information may be found.

In the first case, a list of elements (vertices) is specified which must be processed locally. As an example, if a refinement is demanded (at the element level) then the elements in question are subdivided into smaller elements or, if the demand concerns vertices, the elements sharing such a vertex are processed similarly. Then, if h stands for the element size, the refined elements will have a size smaller than their initial h (for instance, $\frac{h}{2}$ if we consider the edge lengths. This leads to a ratio of 4 (or 8) in terms of surface (volume) variation). This means that a subdivision is made if h is too long but irrespective of the initial range of h. In this way, we go from a value h to a value $\frac{h}{2}$ (for example) while this new value, closer to the desired value, may be still too large.

On the other hand, if a size map is specified (generally at the vertices of a mesh serving as a background mesh), then the affected elements must conform to this map meaning that the h's of the adapted elements must be as close as possible to the specifications. In this way, the adaptation method allows the creation not only of smaller elements (in the case of a refinement as above) but also elements with a prespecified h.

As seen in Chapter 1, a general control space comprises two kinds of data including a spatial description of the domain and metric information provided in some way based on this spatial description. Thus, a control space is a pair *(background mesh-metric specification)* and, for the sake of simplicity, we may use the term background mesh with the same meaning.

21.1.1 Definition of a background mesh

A background mesh serves as a control space for the mesh modification or generation method.

Background mesh : spatial aspect. As already mentioned, a background mesh, in terms of its spatial aspect, can follow various types.

Let us consider that the current mesh (under modification or construction) is a simplicial mesh[3] (triangular or tetrahedral mesh according to the spatial dimension). Then, following Chapter 1, the background mesh could be :

- a simple (uniform or not) grid (*type 1*),
- a tree-type structure (such as a quadtree or an octree) (*type 2*),
- an arbitrary mesh (*type 3*), for instance, a simplicial mesh in our case.

The point is that the background mesh encloses the domain where the adapted mesh will be established. As will be seen, this property which is a source of simplicity, does not always hold (see the discussion about localization problems below).

[3]we mainly consider the case of simplicial meshes as they offer great flexibility in terms of adaptation. Nevertheless, obviously, non simplicial meshes can also be used for adaptive purposes.

Isotropic background mesh : metric aspect. Based on the type of background mesh and related to the adaptation method, the metric aspect could be defined in various ways. The most popular methods consist in the following :

- a refinement (derefinement) demand at the element level,
- a refinement (derefinement) demand at the vertex level,
- a adaptation demand expressed in terms of sizes at the vertex[4] level.

The first two cases are mostly those found in adaptation methods based on the local modification of a given mesh while the third case is generally that used when the adaptation method consists in completely reconstructing the mesh.

For a local modification based method, the background mesh is, in general, the current mesh. The mesh elements contain the adaptation requests and are processed accordingly (see below).

For a global adaptive method, the background mesh could follow one of the three above types. For a *type 1* background mesh, the cells of the grid contain the adaptation requests. These could be related to the cells themselves (for instance, their sizes or a *scalar* value associated with the cells which indicates the desired h in the corresponding regions or again a *scalar* value defined at the cell vertices which is furthermore used to define the desired h in the related regions). For a *type 2* background mesh, we return to the same discussion. Either the cells (quadrants or octants) have been constructed so as to have a size conforming to the desired sizing specification and these sizes govern the adaptation process or a point representative of the cell (its centroid for instance) or the cell corners support the metric information (a *scalar* value). In the case of a *type 3* background mesh, the most popular method consists in associating a *scalar* value with the element vertices. Then, in this case and for an adaptive loop of computation consisting of several iteration steps, the background mesh at iteration step i could be the current mesh of iteration $i-1$ enriched with the sizing (directional) specifications, these values resulting from an *a posteriori* analysis of the solution in the current mesh.

Anisotropic background mesh : metric aspect. Unlike the isotropic situation, a background mesh for anisotropic mesh control contains information with both directional and sizing aspects. In this respect, a *type 3* background mesh appears to be a suitable solution to define the adaptation directives. With each element vertex of the background mesh is associated a *tensor* value representing both the desired directions and the related edge sizes in the neighborhood of the vertices.

[4]Actually, a demand for element refinement or coarsening can be expressed at the element level (for instance, it can be demanded to refine twice a given element). Nevertheless, it appears to be simpler to have the refinement demand expressed at the element vertex level, *i.e.*, it can be demanded to refine twice around a given vertex. However, these two kinds of formulation are closely related.

Such a specification consists of $d \times d$ matrices provided at some points. Such a matrix is denoted, given a point P, as

$$\mathcal{M}(P) = \begin{pmatrix} a_P & b_P \\ b_P & c_P \end{pmatrix}$$

if we consider a two dimensional problem (*i.e.*, d the spatial dimension is 2). Above point P could be a mesh vertex when the background mesh is itself a mesh or a cell corner (or again a point representative of a cell) when the background mesh is a grid or a tree structure. From the metric point of view, the reader is referred to Chapter 10 for the meaning of the coefficients involved in the above matrix.

Remark 21.1 *The isotropic case is a peculiar occurrence of the anisotropic situation where a matrix $\mathcal{M}(P)$ reduces to $\frac{\mathcal{I}}{h_P^2}$ where $\mathcal{I}$ stands for the identity matrix and h_P is the size which is desired around point P.*

Remark 21.2 *In essence, the sizing (directional) information contained in a background mesh is of a discrete nature[5] which means that attention must be paid on how to obtain consistent continuous information (see below).*

21.1.2 Use of a background mesh

Whatever the nature of the background mesh, the actual mesh modification or (re)construction is governed by this structure to which a series of queries is made so as to collect the required information.

Various problems must be carefully addressed including mainly localization problems and, the localization problems being completed, interpolation problems must also be addressed so as to extract the metric information related to the region under consideration.

Localization problem. We face a localization (or searching or point-location) problem since a frequently demanded operation is to find which element (cell) of the background mesh contains a given point of the current mesh.

An adaptive method based on the local modification of the current mesh does not lead to difficulties since there is no localization problem (indeed, the localization is implicit). In contrast to this method, a global adaptive method strictly requires the localization of points in the background mesh. As mentioned earlier, the background mesh could be a grid, a quadtree-octree type mesh or an arbitrary simplicial[6] mesh.

The localization of a point in a cell of a grid or a quadtree-octree type background mesh has been discussed in Chapters 2 and 5 and does not lead to serious difficulties.

[5]Except for rapid testing purposes where analytical sizing (directional) functions can be used for the sake of simplicity and thus where an "ideal" control space is used.

[6]A non-simplicial mesh can be used as well which results in the same analysis but requires a more precise process. Indeed, splitting the non-simplicial elements in terms of simplices allows us to return to the simplicial case (see Chapter 18) for the searching problem.

On the other hand, a simplicial background mesh must be considered in a more subtle way. We can face cases where the current mesh, as well as the background mesh, are not convex, include holes and even are not strictly coincident while they approach the same domain (*i.e.*, the (approximate) domain covered by the background mesh is not exactly that covered by the current mesh due to the fact that the boundary of the current mesh is not meshed in the same way). Thus regions of the current mesh may fall outside the background mesh and conversely.

Therefore, depending on the (global) mesh generation method, localization or searching problems may be of greater or lesser difficulty. As a first remark, it could be observed that a (uniform or not) grid (or a quadtree-octree structure, see for instance [Krause,Rank-1996]) can be constructed to help the searching operation in a given (arbitrary) mesh. Using such a structure, it is easy and fast to find the cell within which a given point falls.

Then, the point is to find the element(s) of the background mesh corresponding to this box which is an easy task. Actually, with each background vertex is associated a background element. Then, a background point in the box within which the current point falls is found and thus an element is found as well which can serve as initial element for the searching procedure. Given this background element, the searching problem, in general, becomes a very local problem that can be successfully solved using a classical searching method (return to Chapter 2). Nevertheless, for non-convex domains, an element in a given box (enclosing the point under consideration) is not necessarily close (topologically speaking) to this point, meaning that a searching procedure initialized with this element may lead to meeting the boundary of the domain. Figure 21.1 illustrates three possible situations that can be encountered in such a localization process.

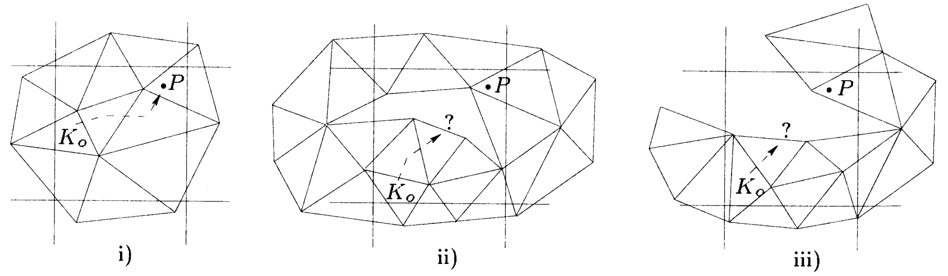

Figure 21.1: *Localization in a two-dimensional domain. We can see one box of the grid associated with the mesh and the mesh elements included or near this box. Three local patterns are visible : i) the local context is convex, ii) a hole exists while point P and element K_0, from which the localization search starts, are separated by this hole, iii) it is demanded to pass through the boundary to reach the element containing point P.*

A natural idea to overcome a difficulty due to a non convex local situation is to visit the boxes neighboring the initial box so as to reach the right part of the domain, then a classical searching method easily finds the solution. Note that this

solution could demand more effort in terms of CPU time.

Remark 21.3 *For a Delaunay type method, a different solution can be envisaged as it is possible to take advantage of the current mesh which is necessarily convex (return to Chapter 7 where a method based on an enclosing box is discussed for which a convex situation always holds). Thus the background mesh is convex or not but the mesh under construction is convex and a subtle writing of the searching procedure easily results in the desired solution.*

Localization problem in a surface mesh. Searching on a surface could be a more tedious problem. Actually, two cases can be considered.

First the surface is the boundary of a domain and a mesh of the latter can be helpful to find the location of a point. In this case, in principle, we return to the previous paragraph where a d-dimensional entity (a point in $\mathbb{R}^3$) must be located in a d-dimensional entity (the background mesh).

In contrast to the previous case, if only a surface mesh is given, we face a localization problem where a $d+1$-dimensional entity (point P) must be located on a d-dimensional domain entity (the surface mesh being seen as a topological entity). It is then easy to construct a grid (or an octree) which encloses the surface mesh. Then, given a point, the cell within which it falls is found[7]. After which, an element in this cell is selected and a classical searching procedure is used. In this way we return to the same difficulties as in the previous paragraph and the same solutions apply.

Interpolation problem. Once a point of the current mesh has been properly localized in the background mesh, we have to collect the sizing (directional) information related to this point (say, related to a "small" region surrounding the point in question). Since the sizing is known in a discrete way, we have to define an interpolation scheme about discrete values so as to extract consistent information at the considered point (region).

Recall that the sizing information is assumed to be defined at the background mesh vertices, then several interpolation problems are encountered based on the type of background mesh and the entity within which the point under investigation falls (background vertex, background edge, background face or element or again background cell).

First, let us consider an isotropic problem. We assume that the background mesh is a simplicial mesh and that point P has been located in a given element (a triangle in the example figure), then using the available metric data (the h's at the element vertices) and the way in which the h function varies, we want to find h_P the corresponding h at P. Then the first case is obvious, point P is coincident with a background point. The second case can be successfully dealt with using the material discussed in Chapter 10 where various interpolation schemes can be found that interpolate h along an edge from the h's at its two endpoints. Indeed, given the h's at the edge endpoints, the desired h at the given point is obtained based on

[7]For the sake of simplicity, we consider that point P lies on a triangle in the surface. If not, the projection of P onto the plane of the triangles must be used in this search.

one of these interpolation functions. The two other cases are more tedious. Using a classical interpolation scheme (between two points) leads to splitting the problem into two parts as can be seen in Figure 21.2 (parts ii) and iii)). First, point P_{12} is found in edge P_1P_2 and the h of this point is obtained by interpolation from those at P_1 and P_2. Then, using segment P_3P_{12}, a solution can be found based on the same type of interpolation. Hence, the scheme is not necessarily commutative and thus the resulting h may depend on the way the known h's are used. To overcome this, see part i) of the same figure, a classical interpolation scheme can be used (for instance a P^1-type interpolation) and the solution depends on what type of variation is desired. We can use a scheme like :

$$h_P = \omega_1(P)h_1 + \omega_2(P)h_2 + \omega_3(P)h_3 \,,$$

with evident notations or something like :

$$h_P = \left(\frac{\omega_1(P)}{h_1} + \frac{\omega_2(P)}{h_2} + \frac{\omega_3(P)}{h_3}\right)^{-1} .$$

In the case of quad elements, tet elements or other classical elements, the same approach can be retained provided the corresponding interpolation function is used.

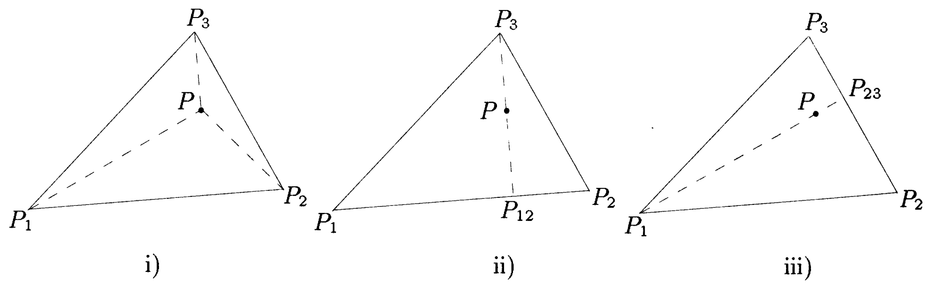

Figure 21.2: *Interpolation in a two-dimensional domain. The three interpolations are equivalent in the case of a linear function while cases ii) and iii) result in a different solution at point P depending on whether P_{12}, constructed using P_1 and P_2, or P_{23}, related to P_2 and P_3, is used in the interpolation scheme.*

Remark 21.4 *Note that a grid or tree-type background mesh can be considered in a rather similar way. Point P is located on this structure and the h's associated with the box (the cell) including P are interpolated as above (for instance, using a Q^1-type interpolant in the case of a quadrilateral cell, Chapter 5).*

Obviously, the same problem when an anisotropic context is desired leads to the same kind of discussion. Now, we need to interpolate both the sizing values (the h's) and the corresponding directions. The material in Chapter 10 allows solution when the interpolation is based on two points (as when P belongs to an edge). However, when the interpolation scheme must use three (or more) points (point P falls within a triangle or within a cell), the above scheme is no longer valid.

Heuristics can be used to complete the solution. Let us consider the case where point P falls within a triangle for which the metric information is known at the three vertices. Let $\mathcal{M}_i$, $i = 1,3$, be the three corresponding metric matrices then a possible interpolation scheme could be as follows :

- compute $\mathcal{M} = \sum_{i=1}^{3} \omega_i(P)\,\mathcal{M}_i$ where $\omega_i(P)$ is a weight function (that of a P^1-interpolation in this case),
- find the eigenvectors of matrix $\mathcal{M}$, *i.e.*, $e_1, ..., e_d$ (d being the spatial dimension, *i.e.*, $d = 2$ in this example)
- compute $h_i^j = \left(\sqrt{{}^te_j\,\mathcal{M}_i\,e_j}\right)^{-1}$, for $i = 1,3$ and $j = 1,d$,
- finally, compute $h^j = \sum_i \omega_i(P)\,h_i^j$, for $j = 1,d$,

then the matrix at P is the matrix whose eigenvectors are the above e_j's while the size h^j, related to direction e_j, is that obtained by the above interpolation scheme.

Remark 21.5 *Other interpolation schemes can be developed leading to a different variation in both the directions and the sizes. In this way, emphasis can be placed on a particular entity depending on what is expected.*

21.2 H-adaptation by local modifications

In this adaptive strategy, the current mesh is locally modified so as to construct an adapted mesh. Thus, the background mesh is naturally the current mesh and the directives for refinement or coarsening are known at the vertices of this mesh.

Local modification of the elements, the basic ingredient of this adaptive strategy, is mainly based on the general tools discussed in Chapters 17 and 18. Two categories of local modifications can be demanded. One corresponds to element refinement so as to increase the density of the mesh in a given region, while the other corresponds to element coarsening leading to a coarser mesh in a specified region. In the following discussion, we recall some tools that ensure the level of refinement (coarsening) which is demanded while maintaining a conforming mesh.

Due to the local tools involved in this type of adaptive approach, one could note that local modification based methods are mainly suitable for *isotropic* adaptation.

21.2.1 Refinement of mesh elements

Element refinement demands can be expressed either at the element level or at the element vertex level.

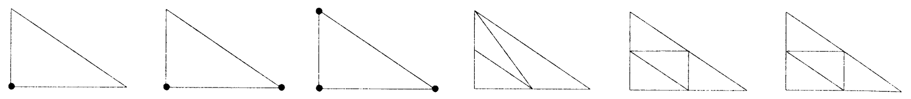
Figure 21.3: *The element must be refined around one, two or three of its vertices. Possible partitions are depicted where only triangles exist.*

Refinement around vertices. In this section, we assume that the refinement demand is expressed at the element vertex level (and not at the element level). In this way, it will be easy to maintain a conforming mesh as will be seen below.

Let P be a vertex in the mesh, if a refinement is prescribed around P then all the edges emanating from P are subdivided into two edges using the edge midpoint. Based on this subdivision, the element is partitioned into sub-elements. The case of a triangular element is shown in Figure 21.3 where, left-hand side, the refinement demand concerns one, two or three vertices. One can see on the right-hand side of the figure the corresponding partitions of the initial triangle.

Obviously all the elements sharing a point where a refinement is required are subdivided in such a way that a conforming refined mesh is automatically obtained.

Remark 21.6 *When a boundary edge must be subdivided, it is necessary to return to the boundary geometry so as to properly locate the midpoint involved in the process.*

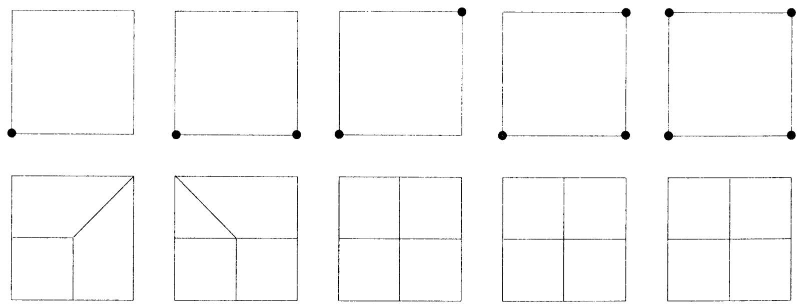
Figure 21.4: *The element must be refined around one, two, three or four of its vertices leading to the partitions depicted. The refinement procedure is a* 2-level *procedure. Note that a triangle is formed in case two (when this is undesirable, the pattern defined for three or four vertices can be used).*

The same procedure for a quad element leads to five cases (Figure 21.4) depending on both the number of affected vertices and their relative locations. Note that this local procedure leads to dividing an affected edge twice, the process is a *2-level* process. A variation, proposed by [Schneiders-1996b], considers different templates as depicted in Figure 21.5. In this case, three subdivisions can be employed based on the number of vertices where a refinement is demanded. Thus the procedure is a *3-level* procedure. As before, conformity is automatically maintained[8] as can be easily observed in the figure (just consider the cases of adjacent elements and apply to them the corresponding templates).

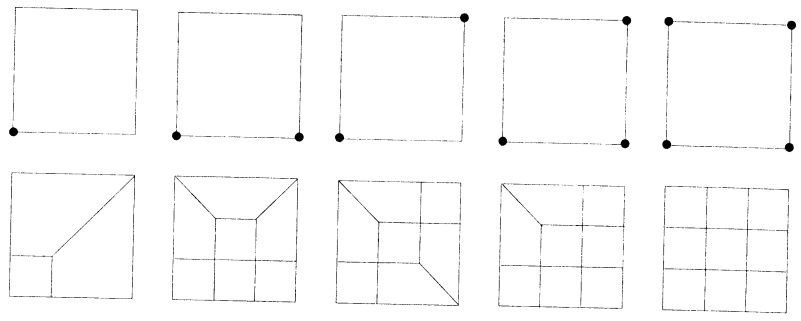
Figure 21.5: *The element must be refined around one, two, three or four of its vertices leading to the partitions depicted. The refinement procedure is now a* 3-level *procedure. Note that no triangles are formed, whatever the case.*

Three-dimensional elements can be dealt with in the same vein. Nevertheless, subdivision procedures are not so obvious. Some examples of possible partitions are given in Figures 21.6 and 21.7.

The templates of Figure 21.6 are now fully described. Let i and j be two vertex indices, we note ij the index of the midpoint of points i and j. With this concise notation, for a given element $[1,2,3,4]$, the four series of templates depicted in the figure correspond to the following vertex enumeration :

- $[1,2,3,4]$ gives $[12,2,23,24]$, $[1,12,3,4]$, $[12,23,3,4]$ and $[12,23,24,4]$ when we refine around vertex 2. Indeed, we define a "small" tet around vertex 2 and, removing this tet, the remaining polyhedron is a pentahedron which, in turn, is split into 3 tets.
- $[1,2,3,4]$ gives $[1,2,3,24]$ and $[1,3,24,4]$ when the edge $[2,4]$ is subdivided.
- $[1,2,3,4]$ gives $[13,1,12,14]$, $[12,2,23,24]$, $[23,3,13,34]$, three "small" tets, along with $[12,23,13,4]$, $[12,13,14,4]$, $[12,24,23,4]$ and $[23,34,13,4]$ when we refine around vertices 1, 2 and 3 meaning that the 6 mid-edges are introduced in the partition.
- $[1,2,3,4]$ gives, introducing again the 6 mid-edges, $[13,1,12,14]$, $[12,2,23,24]$, $[23,3,13,34]$, $[34,4,14,24]$, four "small" tets at the corners and $[13,23,14,12]$, $[12,14,24,23]$, $[23,14,34,13]$ and finally $[34,23,24,14]$.

Exercise 21.1 *Show that the first refinement scheme admits an alternate a priori valid subdivision. Hint : examine the remaining prism (on the fly, retrieve and discard the Schönhart case which is no longer valid).*

[8]Given an adequate definition of the templates used to refine a triangle in the case of a mesh comprising triangular and quad elements.

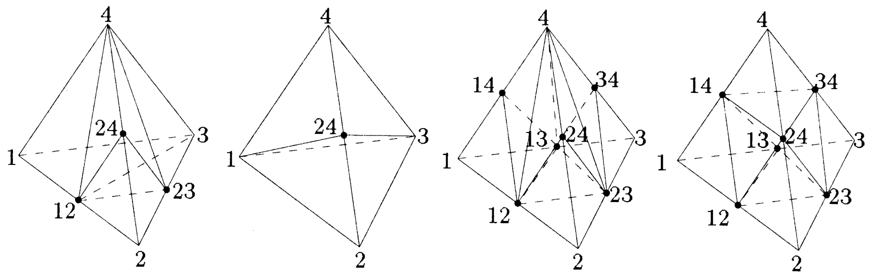

Figure 21.6: *Selected examples of three-dimensional partitions for tetrahedral elements.*

Exercise 21.2 *Show that the last refinement scheme is only one of the three possible subdivisions of a given tet. Hint : examine the polyhedron formed when the four corner based tets are removed. Look at the six possible diagonals based on the six remaining vertices.*

Exercise 21.3 *Examine how the tet subdivision patterns interact with mesh conformity.*

The templates, as depicted in Figure 21.7, concern the case of a 2-level subdivision. More complex and flexible templates can be found when a 3-level procedure is used. In this case, we return to Figure 21.5 for the patterns corresponding to the face subdivision. Using these templates, no conformity problems may be expected.

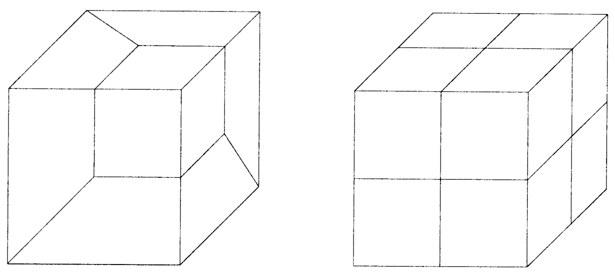

Figure 21.7: *Two examples of three-dimensional partitions for hexahedral elements.*

Local refinements can be coupled with classical mesh optimization tools (such as node balancing, (generalized) edge swapping, etc.) so as to increase (to preserve) the mesh quality.

Remark 21.7 *In many of the above examples, it could be noted that if the edge lengths of the initial element are about unity then the edge lengths of the refined elements range from unity to half (or from unity to a third). Thus, it is not so easy to achieve a size variation following a different variation for two adjacent elements.*

Remark 21.8 *Multiple refinements can be easily obtained by repeating the above element partitioning. Nevertheless, a repeated series of refinements may alter the angles in the elements if some care is not taken thus resulting in ill-shaped elements.*

With regard to what an optimal mesh is (see Chapter 18) and following the two above remarks, it appears that local adaptation methods necessarily induce some extent of rigidity.

Element refinement. In this section, we assume that the refinement demand is expressed at the element level. Thus, maintaining mesh conformity must be made explicitly. Indeed, when an element is refined, one has to propagate the refinement to some neighboring elements if the interfaces (edges or faces) between the refined element and some of its neighbors are affected by the refinement procedure.

Basically, we return to the above templates. However, simplicial elements offer alternative solutions. In this respect, a triangular element can be subdivided by the so-called *longest-side subdivision* method or one of its variations. Advocated in numerous papers including, in a recent issue, [Rivara-1997], the basic idea of this method is to split a triangle by introducing the midpoint of its longest edge and then to propagate (for conformity reasons) the subdivision to some of its neighbors. Nice theoretical issues indicate that the path of propagation is finite and thus mesh conformity is easily obtained and the number of elements still remains reasonable. In addition, this strategy results in a refined mesh where the angles are bounded.

Exercise 21.4 *In two dimensions, show that if a triangle is split by introducing the midpoint of its longest edge, then the two resulting triangles are such that the minimum angle is greater or equal to the minimum angle in the initial configuration. Therefore, the angles do not alter even when repeating this splitting procedure.*

Also, many variations have been investigated including coupling with the Delaunay criterion. The three-dimensional extension of this idea to tet elements can be found in [Rivara,Levin-1992] and, more generally, there are numerous papers by the same author(s) discussing simplicial mesh refinement.

Remark 21.9 *It could be of interest to store the history of a refinement procedure. This can be achieved by storing the item genealogy. Such information could be useful for the inverse process, i.e., a derefinement procedure as will be seen below.*

21.2.2 Element coarsening

Constructing a coarser mesh, based on local modifications of a given mesh, is more tedious. Actually, two approaches can be envisaged that address the derefinement problem in a rather different way. First, the coarsening procedure is seen as an inverse algorithm based on a previously applied refinement algorithm. Second, the coarsening of a mesh is seen as a autonomous procedure that applies for arbitrary meshes however they may have been created.

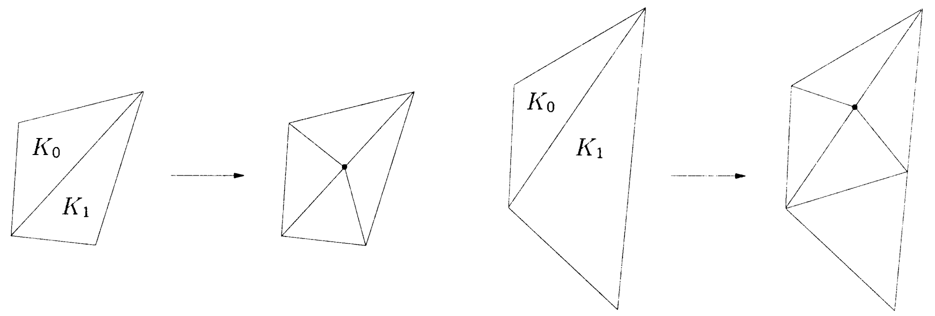

Figure 21.8: *Some occurrences for the longest-side subdivision refinement method in two dimensions. Left, triangle K_0 is candidate for refinement and the edge common to K_0 and its neighbor K_1 is the longest edge of both of these triangles, then the refinement is confined in this pair of elements. Right, triangles K_0 and K_1 are candidates for refinement and the refinement path also propagates to some other (not depicted) elements.*

Derefinement as the inverse of refinement. As it is based on the inverse of a refinement procedure, such an approach is only suitable in regions of the mesh where a refinement has been applied in a previous stage of the computational process. Given a list of candidate nodes for the derefinement procedure, we have to check whether or not they can be removed from the mesh. This point concerns both maintaining mesh conformity and the fact that the genealogy of the investigated node allows the procedure. The genealogy mainly depicts relationships like *children-parent* for the mesh entities (vertices, edges, faces and elements). Based on this history, derefinement consists in rolling backward the refinement procedure which led to the current mesh.

This inverse process either has an immediate effect or else is more subtle to be processed as it must consider not only a transformation (to be processed backward) but also a series of transformations whose inverse implies several operations.

Arbitrary derefinement method. Such a method considers an arbitrary mesh and ignores the way in which it was created. Thus, this derefinement method is based on classical optimization tools (see Chapter 18) that lead to vertex removal (and, consequently, edge, face and element removal). Apart from some peculiar local pathologies (for instance, a vertex in a triangular mesh only shared by three triangles) where an obvious solution can be obtained, the basic tool for mesh coarsening is the edge collapsing operator. Due to this, simplicial meshes are more flexible than other types of meshes as they are more readily candidates for successful edge collapsing.

21.2.3 *R*-method

A *r-method* is a method which adapts the sizes (the *h*'s) while maintaining the connections between the mesh vertices, thus this connectivity remains unchanged.

The principle lies in moving the mesh vertices, in such a way as to increase or decrease the vertex density in certain regions in the domain in accordance with the behavior of the physical phenomenon in question. This can be seen as a mesh deformation.

While well suited to some classes of problems, this method suffers from a certain degree of rigidity. From a numerical point of view, it must be carefully checked that the deformed elements (once their vertices are moved) remain valid and have an acceptable quality. The deformation process (by attracting the vertices in a given region or by repulsing them) is in general an iterative process, the vertex shifts are processed step by step.

Remark 21.10 *It is often very efficient to combine an r-method with some of the previous methods for mesh enrichment or coarsening.*

Note that this method is a local processing (based on vertex moving) but has a global effect on the mesh.

21.2.4 Remarks about local methods

Local modification based adaptive methods are one solution for completing adapted meshes. Nevertheless, a precise examination of the different points discussed in this section leads to some comments. Indeed, this adaptive approach offers both a series of positive aspects and some weaknesses.

Let us mention first some apparent weaknesses of a local method. Nevertheless, note that a tricky implementation could be, in some cases, one way to overcome some of these weaknesses.

- In essence, local methods appear to be a solution for isotropic adaptive problems and seem unlikely to be suitable when anisotropic features are expected.
- The mesh conformity is more or less automatically insured.
- Mesh derefinement is not so obvious.
- Continuous variation in size is not easy to obtain (indeed, we start with an edge length h and obtain a length $\frac{h}{2}$ and thus obtaining another type of gradation is not so simple. In other words, such an approach is far from optimal when the desired size is not in a ratio 2. or 0.5).

On the other hand, positive features of a local method include the following.

- Localization problems are not an issue as the background mesh is, in general, the mesh itself.
- The use of predefined templates allows an easy implementation.
- Also, templates lead, in principle, to a reduced effort in terms of CPU

Note that other computer issues, such as boundary management, are common to both the local approach and the global method discussed in the next section and, thus, do not influence the appreciation of the local approach.

Thus, based on these observations, the user must decide whether or not a local method is a suitable solution to a particular problem.

21.3 Global isotropic adaptation method

Automatic mesh generation methods, as discussed in the previous chapters, are natural candidates for completing adapted meshes. Thus we now discuss how to use or to modify these methods to serve this objective.

In Chapters 5, 6 and 7, we have described *quadtree-octree*, advancing-front and Delaunay type methods. In what follows, we successively return to each of these methods to see if they can be suitably used for the construction of meshes conforming to a metric map specified in advance.

Quadtree-octree based methods can be used to obtain graded (isotropic) meshes as the quadtree-octree structure can be constructed in such a way as to conform to some sizing properties. In this case, the control space could be the quadtree-octree structure itself provided a suitable criterion has been used to complete this structure.

Advancing-front type methods can result in the same kind of meshes by locating the created point properly (say at a location such that the resulting edges are of the desired size).

A Delaunay based method as previously described is also a possible solution that can be used to obtain governed meshes.

In the following sections, we give some details about these three classes of automatic mesh generation methods when an isotropic adaptive problem is considered (the case of an anisotropic situation will be discussed in subsequent sections).

Remark 21.11 *To make the discussion of each method independent, some materials, common to several methods, may, to some extent, be repeated in what follows. Note also that a good knowledge of the methods in their classical version is assumed.*

21.3.1 H-method based on a quadtree-octree approach

A quadtree-octree type mesh generation method (Chapter 5) allows for some flexibility in the mesh creation process leading to the creation of meshes conforming to pre-specified isotropic requests.

A quadtree-octree type mesh generation method can be decomposed into several steps where the first concerns the construction of the underlying quadtree-octree structure. Then, based on this spatial structure, field points are inserted and elements are constructed prior to some degree of optimization.

Underlying tree construction. The tree construction is the basis of the mesh construction. In the classical meshing method, the tree is constructed based on the domain boundary discretization provided as data. Recall that this tree construction is a recursive process in which several criteria are used to decide whether or not the current tree is fine enough or must be refined again.

In the context of a h-method, the same idea can be retained but now the tree is constructed based on the boundary discretization of the domain and the sizing function which is desired. Let $h(P)$ be the size at point P where P is a vertex of the background mesh serving as control space, then the tree must take into account this information. Thus, a reasonable method resulting in an adequate tree structure could be as follows :

- construct the tree structure based on the domain boundary discretization (which is assumed to conform to the size map),
- refine, if necessary the above tree, by checking if it conforms to the h's included in the background mesh which means that the size of a terminal cell corresponds to the desired element size. To this end, for each cell of the tree, one can find the vertices of the background mesh which fall within this cell. Then, examine if the cell size conforms to the sizing values. If not, pursue the tree decomposition until a satisfactory matching has been reached, while applying classical rules (such as the 2:1 rule) to balance the tree. Figure 21.9 depicts how the h's data interact with the tree construction.

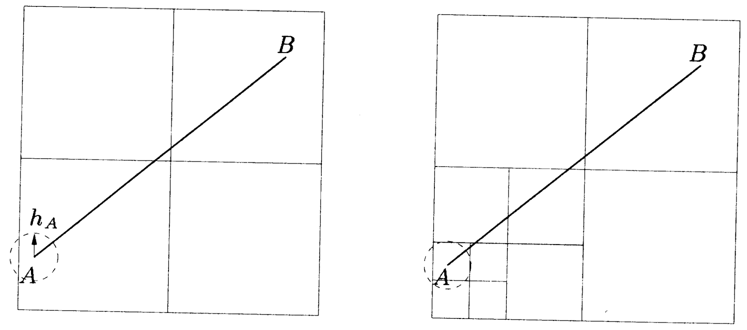

Figure 21.9: *Tree decomposition using a prescribed size. The size at point A forces the initial tree box containing A (left-hand side) constructed without explicit sizing prescription to be decomposed two levels deeper (right-hand side) to adjust to the size h_A represented by the dashed circle.*

Remark 21.12 *The tree level l is then related to the size of a mesh edge, h, as $l = \log_2(b/h)$ where b is the length of a side of the root cell.*

Remark 21.13 *Note that, in this method, the distance between two connected mesh vertices (i.e., an edge length) is controlled by the cell sizes. Hence, the edge size control is not obtained explicitly but results from the tree decomposition.*

Remark 21.14 *Also, using the [2:1] rule results in a ratio of one, one half or two between two adjacent cells (in terms of size).*

Element construction. Once the tree has been completed, the element creation follows the same aspect as in a classical quadtree-octree type method (return to Chapter 5).

Mesh optimization. Mesh optimization, taking place at the end of the meshing process, is slightly different than in the classical meshing problem. Unlike this case where only the element shapes were considered, we have now to take into account two (possibly antagonistic) aspects. Actually, what is needed are well-shaped elements conforming a specified size. Thus these two aspects must be the objective of the optimization stage. We refer the reader to a further section on this point as optimization purposes in this quadtree-octree based method are similar to those of the two other meshing techniques described below.

In conclusion, adaptation using a *quadtree-octree* method, such as that proposed above, is essentially based on a particular construction of the tree structure so that the cells in this tree reflect through their sizes the desired metric specifications. Therefore, the control is achieved by means of the tree.

21.3.2 H-method based on an advancing-front approach

An advancing-front type mesh generation method (see Chapter 6) allows for some flexibility in the mesh creation process leading to the creation of meshes conforming to pre-specified isotropic requests.

An advancing-front type mesh generation method includes the construction of the field points and their connection with the current front and a final step that corresponds to some extent of optimization.

Point creation and point connection. One way to control the mesh creation with regard to a given sizing map is to analyze the length of the mesh edges. Let us assume a three-dimensional case. Then, given a front face, say ABC, we examine the context to decide whether an existing point or a new point must be created that can be combined with face ABC so as to form an element. In the case where this candidate element is indeed formed, we will have introduced one, two or three new edges depending on the situation. As a consequence, a control on this (these) new edge(s) allows us in principle to complete what is needed.

Thus, let us examine a (potentially new) edge of this candidate element, say AP (we assume that P is known, see below regarding a way to find such a point). If $h(t)$ stands for a sizing function defined for this edge (with $h(0) = h(A)$ and $h(1) = h(P)$), then the length of edge AP with respect to $h(t)$ is :

$$l_{AP} = d_{AP} \int_0^1 \frac{1}{h(t)}\,dt \qquad (21.1)$$

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

Actually, $h(t)$ is only known in a discrete way (thanks to the background mesh). So, if only $h(A)$ and $h(P)$ are provided, we have (using an approximation based on a rather simple quadrature formula) :

$$l_{AP} = d_{AP} \frac{\dfrac{1}{h(A)} + \dfrac{1}{h(P)}}{2} \qquad (21.2)$$

while if edge AP intersects a series of elements in the background mesh and if M_i stands for the i^{th} intersection point, then we can define the length of AP as :

$$l_{AP} = \sum_{i=0}^{i=n} l_{M_iM_{i+1}} \qquad (21.3)$$

where (using the same quadrature formula)

$$l_{M_iM_{i+1}} = d_{M_iM_{i+1}} \frac{\dfrac{1}{h(M_i)} + \dfrac{1}{h(M_{i+1})}}{2}. \qquad (21.4)$$

In this relationship, $h(M_i)$ is obtained by interpolation on the background mesh. Using these definitions, the (potentially new) edges of the analyzed element are examined. This means, Figure 21.10, that we may be interested in computing the lengths of edges AP, BP and CP, given the face ABC. Based on these values, as compared with the unity, point P is moved towards or away from the face (see Chapter 6) and the same analysis is repeated until (a satisfactory) convergence is obtained.

In theory, we need to find a point, P, the solution of :

$$l_{AP'} = l_{BP'} = l_{CP'} = 1\,,$$

which is, in practice, unrealistic. Therefore, the above approximate method (based on iterations) is a reasonable solution.

The above discussion assumed that P was found in some neighborhood of face ABC meaning that an existing point can be used or, on the other hand, that P is constructed somewhere. This being done, point P is easily adjusted using the previous material. The point is then to find an initial (and not too bad) candidate point. One method could be, based on face ABC, to define point P_{opt} such that element K is well-shaped. Then, among the three edges $P_{opt}A$, $P_{opt}B$ and $P_{opt}C$, that or those which will be added in the mesh in the case where element K is retained, are examined based on the metric map. If P_{opt} is a free-point, its location is adjusted to meet as closely as possible a unit length for the edges of element K. Then P is assigned to be the resulting adjusted point. Note that all these operations are coupled with the classical validity checks as performed in a classical advancing front type method.

Remark 21.15 *Note that the radii used for the searching operations for the candidate points must take into account the desired element sizes.*

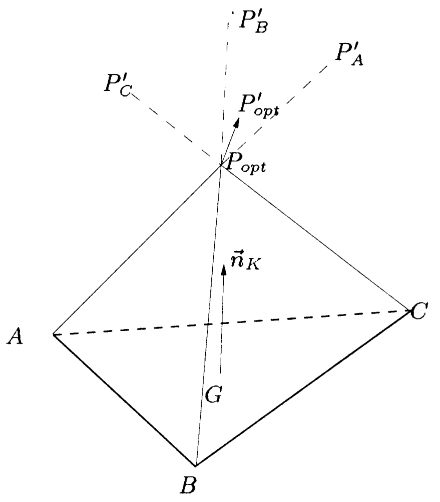

Figure 21.10: *Optimal point P creation, given a front face ABC the resulting tetrahedron K is considered optimal as it has edge lengths conforming to the desired sizing map.* P_{opt} *is first constructed, then* P'_A, P'_B *and* P'_C *are defined and an adequate combination of these (virtual) points is used to define* P'_{opt}. *Finally, the new location of P is prescribed to be this* P'_{opt}.

Remark 21.16 *In [Rassineux-1995], the internal points are constructed using an* octree, *in three dimensions. Thus, provided this tree structure is developed in accordance with the size map (cf. the previous approach), we obtain a priori a mesh which is reasonably satisfactory with regard to this size map.*

Mesh optimization. As above, we refer the reader to a later section on the optimization step of the method.

In conclusion, adaptation by using an advancing-front method, such as that proposed here, basically relies on an appropriate location of the points with regard to the front faces. The global control, each point being well located for its face, is due to the fact that the candidate points are selected in a neighborhood of the faces in question and thus these points are globally well located.

21.3.3 *H*-method based on a Delaunay-type method

A Delaunay type mesh generation method also appears to be flexible enough to carry out the creation of meshes conforming to pre-specified isotropic requests.

As we saw in Chapter 7, a Delaunay type mesh generation method can be decomposed into several steps that concern the insertion of the boundary vertices, the boundary enforcement, the creation of a suitable set of field points prior to their insertion and some extent of optimization.

Delaunay kernel. Recall that, under appropriate assumptions, the Delaunay kernel is the basic ingredient that makes it possible to insert a point in a given triangulation. This Delaunay kernel is simply expressed by the relationship

$$\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{C}_P + \mathcal{B}_P\,, \tag{21.5}$$

where the cavity and the ball of P are involved. Then, whatever the size map, this basic algorithm remains valid.

Boundary enforcement. Obviously, the *a posteriori* enforcement of the boundary entities when the boundary vertices have been inserted, remains unchanged in this adaptive context.

Internal point creation. Following the classical point creation method (Chapter 7) using as a support the edges of the current mesh, we have to examine how this method could be used in the present adaptive context.

Note initially that the proposed method is rather similar to that used in the above advancing-front approach since it is based on a control related to the edge lengths.

Thus, let us examine an edge, say AB. If $h(t)$ stands for a sizing function defined for this edge (where $h(0) = h(A)$ and $h(1) = h(B)$ hold), then the length of edge AB with respect to $h(t)$ is

$$l_{AB} = d_{AB} \int_0^1 \frac{1}{h(t)}\, dt \tag{21.6}$$

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

Actually, $h(t)$ is only known in a discrete way (thanks to the background mesh). So, if only $h(A)$ and $h(B)$ are provided, we have (using an approximation based on a rather simple quadrature formula) :

$$l_{AB} = d_{AB}\, \frac{\dfrac{1}{h(A)} + \dfrac{1}{h(B)}}{2}\,, \tag{21.7}$$

while if edge AB intersects a series of elements in the background mesh and if M_i stands for the i^{th} intersection point, then we can define the length of AB as :

$$l_{AB} = \sum_{i=0}^{i=n} l_{M_i M_{i+1}} \tag{21.8}$$

where $l_{M_i M_{i+1}}$ is approached using Relationship (20.4). Using these definitions, the edges of the current mesh are analyzed so as to create some points along them.

This can be done using the very simple algorithm that follows (where n is the number of the M_i's) :

```
$l = 0$. (where $l$ stands for a length)
DO FOR $i = 0, n$
```

```
    $l = l + l_{M_i M_{i+1}}$
    (A) - IF $l > 1$
        create one point between $M_i$ and $M_{i+1}$ at a unit distance from the
        previously created point ($M_0 = A$ at the first step),
        $l = l - 1$ and return to (A).
END FOR $i$.
```

This process is repeated for all the edges of the current mesh leading to the creation of a series of points. These are inserted using the Delaunay kernel resulting in a new mesh. The process is then repeated until all the edges are satisfactory (in terms of length). Actually, the unit value is replaced by an appropriate value close to one (as it is for all of these length based methods).

Mesh optimization. As above, we refer the reader to the next section about the optimization step of the method.

In conclusion, adaptation for a Delaunay type method mostly relies (in the proposed approach) on the proper position of the points in the mesh edges. The global control, every point being well located on its supporting edge, is due to the fact that the points are filtered before insertion, this leading to a global consistence.

21.3.4 Mesh optimization tools (isotropic case)

The optimization stage met in all automatic mesh generation methods (for simplicial meshes) is, in practice, the same for all of them (*quadtree-octree*, advancing-front or Delaunay). As already seen, this stage is the last in the mesh construction process. The optimization is governed by a quality objective (at the element level) with includes two criteria, a *shape* criterion and, at the same time, a *size* criterion. The point is what mesh quality is expected and, based on this objective, what tools can be used and how to combine them (*i.e.*, how to define a mesh optimization strategy) so as to improve the mesh resulting from the first steps of the mesh generation (modification) method.

Mesh evaluation. Provided with a metric map, the mesh quality analysis must naturally take into account this map to examine whether or not the mesh elements conform to it. In addition, as we are considering an isotropic situation, the element shapes (aspect ratios) must be as good as possible. Indeed, we have discussed all these aspects in Chapter 18 where aspect ratio measures as well as length efficiency index have been introduced and which form the basic ingredients we need to evaluate a mesh quality.

However, the above appreciation is not directly related to the main reason for constructing the presumably adapted mesh and thus, this appreciation is only formal. In fact, in the context of an adaptive loop of finite element computation (our purpose), the right tool to analyze the mesh quality is the error estimate which analyzes the quality of the solution of the P.D.E. problem under consideration. Since the error estimate is not known in the meshing process, we assume that the above formal appreciation is valid (and consistent). Thus, at the meshing stage of the entire process, we still use this kind of appreciation.

As detailed in Chapter 18, various local tools can be used to optimize a given mesh. In our context, the same tools are used in the optimization step included in the mesh generation methods.

Shape optimization. Classical optimization tools (Chapter 18) can be used such as node relocation, edge swapping, generalized swapping, etc.

Size optimization. Edge collapsing or edge splitting can be performed when a given edge is too short (in terms of l measured according to the discrete metric map) or too long. Prior to actually apply such an operator, one has to check whether or not it results in a better solution. For instance, an edge that is too long could be retained if splitting it into two shorter edges results in two violations of the sizing criterion (instead of only one violation in the initial configuration). Apart from these two processes, node relocation proves to be useful. A free vertex, namely P, can be moved as follows :

$$\overline{P}_j = P_j + \frac{P_jP}{||\overrightarrow{P_jP}||}\,\overline{h}_j \qquad (21.9)$$

where $\overline{h}_j$ is such that $l_{P_jP} = 1$ in the metric. Then, the new position of point P can be the centroid of the $\overline{P}_j$'s.

Optimization strategy. First, it could be observed that respecting a size map while maintaining well-shaped elements could be two conflicting objectives, based on what the size map is. Thus, the optimization strategy must take into account this possible conflict. Following our experience, we suggest firstly optimizing the length criterion and then, this objective being satisfied, optimizing the element shapes (although this may alter the length criterion to some degree).

21.3.5 Remarks about global methods (isotropic)

As for the local approach, we give some preliminary observations about global adaptive methods. The aim is not to pass judgment on the various mesh generation methods that may be employed but to give a general impression of a global approach.

Indeed, this adaptive approach offers both a series of positive aspects and some weaknesses. Let us mention first some apparent weaknesses of a (isotropic) global method.

- Localization problems could be an issue and could be time consuming.
- CPU requirements could be large, particularly when the difference between the initial mesh and the adapted mesh concerns only a small part of the computational domain.

Now, positive features of a global method are briefly indicated.

- Essentially, global methods appear to be a good solution for isotropic adaptive problems.
- Mesh conformity is not an issue.
- Mesh derefinement is obvious since it is no different from a refinement problem.
- Continuous variation in size is easy to obtain.

Note that other computer issues, such as boundary management, are common to the local method discussed in a previous section and, thus, do not count in the appreciation of the approach.

Thus, based on these observations, one has to decide whether or not a global method is a suitable solution to carry out a given problem.

21.4 Global anisotropic adaptation method

Anisotropic meshes or meshes with anisotropic meshed regions are of great interest for some numerical simulations (one can consider the construction of boundary layers or the case where the problem induces some shocks).

As for the previous meshing problem, anisotropic mesh generation can *a priori* be based on any of the classical mesh generation methods. Nevertheless, each of them must be precisely examined so as to see if it can be easily extended to this particular meshing problem. In what follows, we review the *quadtree-octree*, the advancing-front and the Delaunay type methods with this precise objective.

21.4.1 H-method based on a quadtree-octree approach

Basically, the mesh elements resulting from a quadtree-octree type method, are strongly related, in terms of size as well as in terms of shape, to the underlying tree structure. More precisely, the mesh elements are created based on the cells of the tree structure. Thus, since the cells are in essence isotropic, it is not so easy to find a method resulting in the construction of anisotropic elements. Up to now and as far as we know, there is no available literature about this point and no attempts to construct an anisotropic quadtree-octree structure.

Remark 21.17 *Nevertheless, it could be observed that a global anisotropy (*i.e., *where the anisotropy is globally defined and aligned with the usual coordinate axis and also does not vary from regions to regions) can be obtained by constructing the tree structure accordingly.*

Despite the above observations, it could be of interest to examine how it is possible to construct a quadtree (octree) structure using a repeated subdivision of the root cell not aligned with the sides of this cell. This being completed, one could examine whether or not such a spatial partition can be exploited for anisotropic mesh construction.

21.4.2 H-method based on an advancing-front approach

An advancing front type mesh generation method is more flexible than the previous type of method since point connection as well as point placement can be guided by anisotropic criteria.

In an anisotropic context, directional features and sizes varying in these directions are expected at the mesh element level. This leads to introducing what the length of an edge is, with respect to a given metric map.

Length of an edge. In Chapter 10, we have seen that computing a length consists in using the dot product related to a quadratic form. Here we are concerned with the curve Γ joining two points A and B. If $\gamma(t)$ is a parameterization of Γ of class C^k $(k \geq 1)$ such that $\gamma(0) = A$ and $\gamma(1) = B$, the *length* $L(\gamma)$ of the arc Γ in the metric $\mathcal{M}_\gamma$ is defined by the expression :

$$L(\gamma) = \int_0^1 ||\gamma'(t)||dt = \int_0^1 \sqrt{{}^t\gamma'(t)\,\mathcal{M}_\gamma\,\gamma'(t)}\,dt\,. \tag{21.10}$$

Therefore, the restriction of the parameterized arc Γ to the vector $\overrightarrow{AB}$ (*i.e.,* the edge under examination) with the parameterization $\gamma(t) = A + t\overrightarrow{AB}$, $t \in [0,1]$ and $\gamma(0) = A, \gamma(1) = B$ enables us to write the length $L(\gamma)$ of the line segment AB as :

$$L(\gamma) = \int_0^1 \sqrt{{}^t\overrightarrow{AB}\,\mathcal{M}_\gamma\,\overrightarrow{AB}}dt\,, \tag{21.11}$$

where M_γ is the metric specification in γ. Once the metric does not vary with the position ($\mathcal{M}_\gamma = \mathcal{M}$), we obtain :

$$L(\gamma) = \sqrt{{}^t\overrightarrow{AB}\,\mathcal{M}\,\overrightarrow{AB}}\,.$$

Having this definition, it becomes possible to compute the lengths of the mesh edges is the case where an anisotropic metric is given.

Internal point creation. As for the isotropic case above, creating an adapted mesh is based on the edge length control with respect to the metric map. Given a front face ABC, we want to know whether a point P in the current mesh exists which is suitable for the construction of elements whose edge lengths are compatible with the sizing and directional specifications. If such a point is found, it is used so as to create a mesh element. Otherwise, the above method is used to find the location of an *optimal* point. A point P_{opt} is first constructed in such a way that the element $ABCP$ has a nice shape quality. Then, the position of P_{opt} is adjusted so as to meet unit edge lengths in the metric.

The element constructed based on ABC and P_{opt} is an element whose size conforms to the size specification and whose stretching and orientation follow the directional features included in the anisotropic metric map.

Metric interpolation. In practice, the metric at a vertex is known in a discrete manner (in fact at the background mesh vertices). Therefore, it is necessary to use a metric interpolation procedure so as to obtain the metric value(s) at a vertex P in the current mesh. To this end, the element K in the background mesh in which point P falls is identified (this point is that resulting in an optimal element). Using the metric information at the vertices in K, the metric at P is obtained by interpolation and then serves to adjust the position of P, so as to obtain unit length for the edges AP, BP and CP.

Remark 21.18 *The advancing-front approach is also able to construct some boundary layers in the vicinity of a given boundary. This boundary forms a front which is then "pushed" in order to define the first layer, this then forms a new front and the same method is repeated [Kallinderis* et al. *1995].*

Remark 21.19 *A combination with a Delaunay type method is also a solution for completing anisotropic elements. The optimal points are created using an advancing-front strategy and are then connected using a Delaunay method (see [Mavriplis-1992], for instance).*

21.4.3 H-method based on a Delaunay-type method

Given the final theoretical remark of Chapter 7, the Delaunay based method as described in this chapter is probably also a possible answer for anisotropic mesh generation. As previously seen, a Delaunay type method is mainly based on the notion of a distance between two points. Indeed, the Delaunay kernel, the field point creation phase and the optimization phase all make extensive use of this kind of information[9].

Edge length. Before going further, we recall how the necessary lengths are defined. Given a curve Γ, joining two points A and B, we consider $\gamma(t)$ a parameterization of Γ, at least of class C^1 (t ranging from 0 to 1) such that $\gamma(0) = A$ and $\gamma(1) = B$. Then, $l_{\mathcal{M}}(A,B)$, the distance, following the metric map characterized by the matrices $\mathcal{M}$'s, between A and B is $l(\Gamma)$, the length of Γ defined as follows :

$$L(\gamma) = l_{\mathcal{M}}(A,B) = \int_0^1 \sqrt{{}^t\overrightarrow{AB}\,\mathcal{M}_\gamma\,\overrightarrow{AB}}\,dt\,, \tag{21.12}$$

or, if metric $\mathcal{M}$ does not depend on the position :

$$L(\gamma) = l_{\mathcal{M}}(A,B) = \sqrt{{}^t\overrightarrow{AB}\,\mathcal{M}\,\overrightarrow{AB}}\,.$$

Now, using this length definition, we can return to the scheme of the isotropic case and compute the length of AB using metric information, the $\mathcal{M}$ matrices, collected on the background mesh.

[9]While the other phase (boundary enforcement), part of the whole method, is, in principle, not affected by the present meshing context.

Point insertion method (Delaunay kernel). Actually, the classical Delaunay point insertion algorithm is no longer suitable. Indeed, following the method proposed in the isotropic case while modifying the way in which the points are located (*i.e.*, by computing the unit lengths using Relation (21.12) instead of Relation (21.6)) results in *a priori* well located internal points. Inserting these points via the classical Delaunay kernel is possible, but the underlying proximity notion (based on a Euclidean distance) does not match the way in which the distances from point to point have been evaluated. Thus, the local procedure (already described elsewhere) :

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

enabling us to insert point P in triangulation $\mathcal{T}_i$ so as to complete $\mathcal{T}_{i+1}$ must be extended to the present context.

Note that $\mathcal{B}_P$ is the set of elements formed by joining P with the external edges (faces) of $\mathcal{C}_P$, these being the series of elements in $\mathcal{T}_i$ whose circumcircle (circumsphere) encloses P. Thus, now, this notion of a circle (sphere) is defined according to the metric associated with the problem.

The key-issue is to construct a proper *cavity*, *i.e.*, the set $\mathcal{C}_P$. To this end, we consider a two-dimensional meshing problem and, at first, we return to the classical case. Let K be an element of $\mathcal{T}_i$, let O_K be its circumcentre and let r_K be its circumradius. Then, element K will be a member of $\mathcal{C}_P$ if

$$\alpha(P,K) = \frac{d(P,O_K)}{r_K} < 1 \tag{21.13}$$

where d is the usual distance. To take into account a metric map, this classical characterization is replaced by :

$$\alpha_{\mathcal{M}}(P,K) = \frac{l_{\mathcal{M}}(P,O_K)}{r_K} < 1 \tag{21.14}$$

where, now,

- O_K stands for the point equidistant to the vertices of K. This means that this point is the center of the circumcircle of K according to the metric defined by $\mathcal{M}$ (in general, this circle is an ellipse in the usual Euclidean space) and,
- r_K is the radius of this circle according to the Euclidean space defined by $\mathcal{M}$, *i.e.*, r_K is the length between the point equidistant to the three vertices of K, the above O_K, and one of these points.

Computing O_K as well as r_K is not, in general, possible, as $\mathcal{M}$ varies from one point to another. Thus, *approximate* solutions must be involved leading back to a Euclidean problem. The simplest one consists in approaching $\mathcal{M}$ by the value of $\mathcal{M}$ at point P. This approximation results in a possible construction of $\mathcal{C}_P$ (after a correction step) and the point insertion method applies in an anisotropic context (return to Chapter 7).

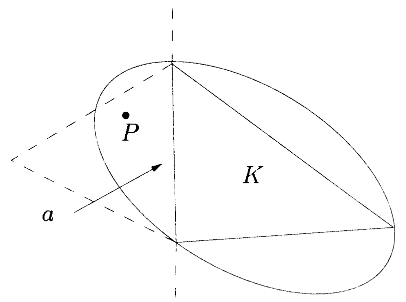

Figure 21.11: *The circumcircle associated with element K once the metric is fixed. The construction of the cavity of P by means of adjacencies relies, starting from one triangle in this set (in dotted line), in examining its neighboring triangles, here that share the edge denoted by a,* i.e., *element K.*

It has been proved (thanks to the correction step) that the resulting cavity has the desired properties and thus that replacing it by the ball of P results in what is expected : an anisotropic point insertion method.

Remark about the boundary enforcement. In practice, as the elements in manipulation can be strongly anisotropic, numerical problems[10] may arise.

Internal point creation. This step is similar to the isotropic case previously discussed while the lengths are evaluated using the above (anisotropic) formula. We return to the isotropic scheme where queries about the background mesh together with interpolation of the collected information are used to compute the length of a given edge. As for the Delaunay kernel, an approximation is used in this length computation.

21.4.4 Mesh optimization tools (anisotropic case)

Common to all automatic methods resulting in simplicial elements, an optimization phase forms part of the meshing process. The optimization here is based on a shape quality criterion coupled with a size (directional) quality criterion. The delicate issue is, as in the isotropic case, to define a optimization strategy that produces what is desired.

Mesh evaluation. The analysis of the resulting mesh must take into account the specified metric map. The aspects about optimization itself have been discussed in Chapter 18. It should just be borne in mind that the efficiency index allows for a global appreciation of the lengths of the mesh edges. Moreover, using an error estimate may provide information (in the form of a metric map) that makes

[10] While in principle, this stage is similar to how it was in a classical case.

the mesh analysis possible. The tools for optimization as described in Chapter 18 apply here with no restriction.

Shape optimization. Among the classical optimization tools for element shape improvement, we have edge swaps, point moves (with a unit length), etc.

Length optimization. The edge lengths having been computed in the given metric, the collapsing (resp. splitting) operators deal with the edges that are too small (resp. too long). The point relocation procedure consists in trying to obtain unit length edges (in the metric at the considered vertex P).

Optimization strategy. As in the isotropic governed case, an efficient strategy leads to optimizing the mesh by firstly considering the size criterion. Then, a quality (shape) criterion is considered.

21.4.5 Remarks about global methods (anisotropic)

In this section, we give some remarks about anisotropic global mesh adaptation methods. As in the isotropic case, a series of (relative) weaknesses can be found :

- localization problems may be tedious,
- the time required for a remeshing may be relatively great, specifically when the adaptation is rather local,
- numerical problems may arise due to the stretching of the elements (when computing surface areas or volumes for instance).

Nevertheless, a certain number of positive features can be mentioned :

- the global remeshing approach is a likely solution for anisotropic adaptive problems,
- a mesh gradation is, in general, relatively easy to achieve (or to maintain).

As in the isotropic case, mesh derefinement (coarsening) is a trivial task since it does not differ from a refinement method. Note that it is always the user's responsibility to decide whether the problem at hand requires global anisotropic approach.

21.5 Adaptation

Adaptation is a key-issue for automatic simulations where the purpose is to insure a given accuracy of the solution. Given a tolerance threshold, the problem is to compute, for a given P.D.E. problem, a solution whose accuracy conforms, in some sense, to this threshold.

There are several approaches suitable for adaptation purposes. As discussed up to now in this chapter, these include the h-method where the mesh, the support for the computational step, is adapted in terms of element size (density) or sizes and directions. In an isotropic meshing problem, the desired mesh must be coarser or finer in such or such region as specified by an error estimate that analyzes the quality of the solution computed using the current mesh as a spatial support. In an anisotropic context, the elements must be aligned in the directions specified by such an error estimate which is assumed to have a directional aspect and, at the same time, these elements (indeed their edges) must have the required lengths.

Following the previous discussion, the mesh generation aspect can be envisaged in two ways so as to design a h-method. The first is based on local modifications of the current mesh while the other involves a entire mesh (re)construction at each step of the process (or after a few iteration steps).

The ingredients needed in this context include local mesh modification tools (local approach) or fully automatic mesh generation processes (global approach) resulting in adapted meshes along with solution methods coupled with *a posteriori* error estimates able to provide mesh specifications used, in turn, to repeat the full process until the tolerance threshold has been achieved.

In what follows, we indicate what a global computational scheme could be for both a local and a global meshing approach in the case of an adaptive loop of F.E.M. style computation. The example concerns a three-dimensional case from which it is easy to infer what a two-dimensional case could be.

21.5.1 General framework of a local adaptation method

The general framework of a h-method adaptive loop of computation when the adaptation is based on local mesh modifications at each step is illustrated in Figure 21.12 (in three dimensions).

The general scheme includes two parts. Left, we find a scheme similar to that in a classical mesh generation method. Right, we see the part directly related to the adaptation phase.

More precisely, for the classical part, we start from a CAD system (box "CAD" in the figure), to define a first surface mesh ($j = 0$), then a 3D mesh generator creates the mesh of the domain and this part (left-hand side of the figure) is nothing other than a *classical mesh generation problem*. Then the solution of the P.D.E. problem is computed and analyzed using an *ad-hoc* error estimate. The latter provides a metric map (box "Metric"). Based on this metric map, the process is iterated ($j = j + 1$) thus consisting in the second part of the figure (right-hand side) and corresponding to the *governed mesh modification problem*. Note that the geometry of the surface (box "Geometry definition") is now strictly necessary as will be discussed below. The local mesh modification step consists in modifying the mesh so as to complete a mesh conforming to the data included in the metric map.

The metric map is indeed a simple request associated with each element of the type *element to be subdivided* or, conversely, *element to be coarsened*. In practice, the same type of requests may be given at the vertices of the current mesh (for example, refine (once or several times) around a given vertex).

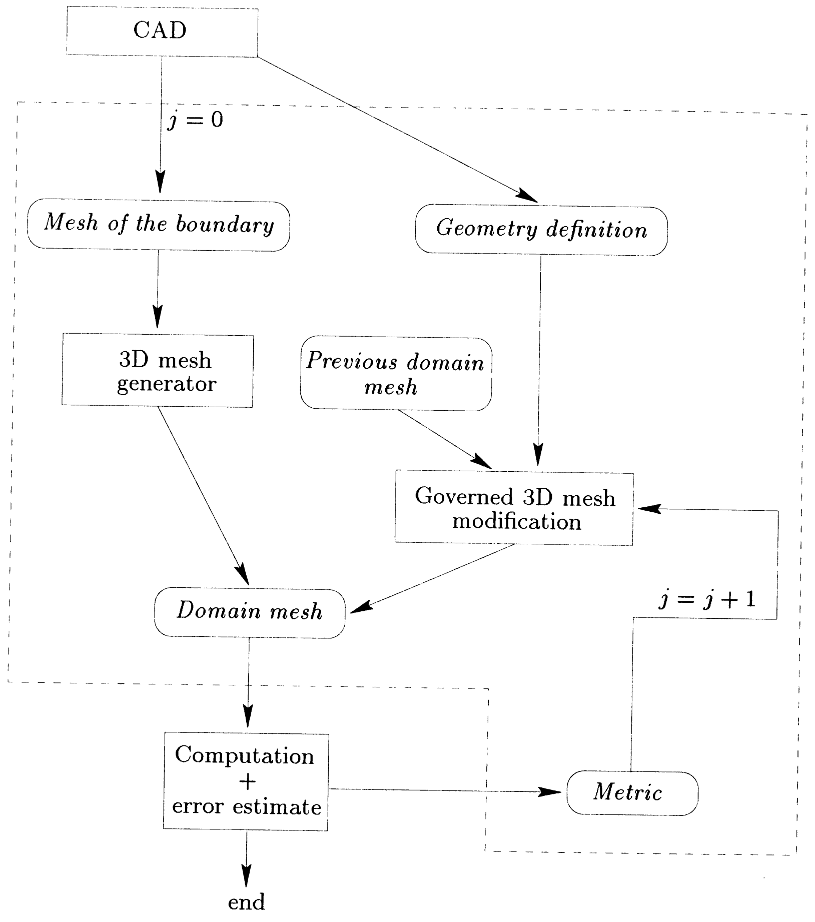

Figure 21.12: *General framework for a local adaptation method.*

21.5.2 General framework of a global adaptation method

The general framework of a h-method adaptive loop of computation when the adaptation is based on the entire mesh (re)construction at each step is depicted in Figure 21.13.

As for the previous approach, we can see two parts as shown in the previous figure. Indeed, the first part (left-hand side, the *classical mesh generation problem*) remains unchanged while the second part (right-hand side, the *governed mesh creation problem*) is rather different.

The classical part in the scheme is similar to that in the previous case and completes an initial mesh of the domain using an initial mesh of its surface as data.

In contrast, the adaptive part in the scheme differs slightly from that included in the previous scheme. Now this part comprises two steps. One concerns the surface mesh processing while the other, based on an automatic mesh generation method, considers as input the above surface mesh (assumed to be conform to the metric specifications) and completes the domain mesh accordingly.

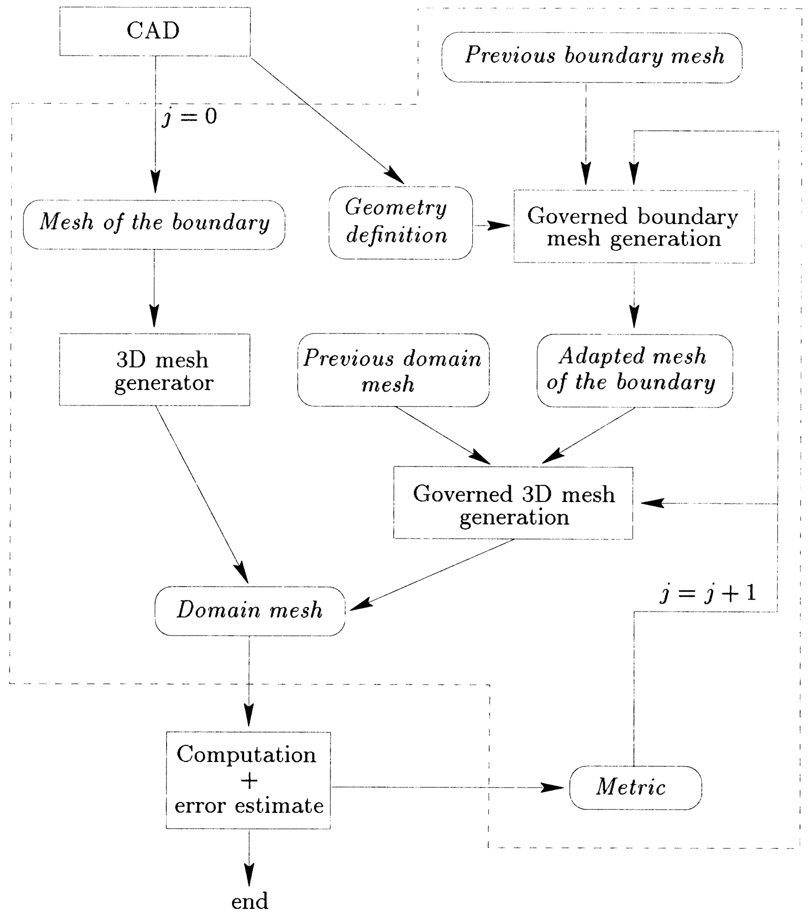

Figure 21.13: *General framework for a global adaptive method.*

21.5.3 Remarks about an adaptive scheme

The two above general frameworks include a series of processes which, in turn, involve various types of input and data flows. In what follows, we give some ideas and comments about these aspects.

Geometry definition. The precise definition (in an analytical way, for instance) of the geometry of the domain surface to be meshed is not strictly required in a classical meshing process[11]. In fact, the domain mesh is generally obtained using a discretization of its boundary as input data. Therefore, this boundary mesh is assumed to be known and is sufficient for the definition of the domain in question.

In contrast, in an adaptive meshing process, the discretization (the mesh) of the domain surface may vary during the iterations, depending on the metric specifications. It is then necessary to have access to a definition of the geometry (the boundaries) of the domain. In practice, there are two ways to obtain the information related to the geometry :

- using a direct access to a geometric modeler (a CAD system) to which queries are made by the mesher so as to know the geometrical or topological information which is needed (corners, ridges, curve definitions, surface definitions, normals, tangents, curvatures, etc.);
- using an indirect access, which means using a mesh, the so-called *geometric mesh* which serves as the geometric definition of the domain (Chapter 19).

When a geometric mesh serves as a support for the geometry definition, the properties of the boundary curve (resp. surface) are obtained using this discretization. In principle, such a mesh is constructed by the CAD system and the geometric approximation it forms is assumed to accurately reflect the geometry of the curve (resp. surface) it models. Therefore, a mesh whose element density indicates the curvature well is a suitable candidate. On the other hand, it must be noted that such a mesh is not generally a suitable mesh for a finite element computation (in particular, the mesh gradation is not necessarily controlled).

Given such a geometric support, we return to the previous case in which queries are addressed to the geometry in order to obtain the required information.

About the boundary mesh construction. A mesh of the domain boundary is the natural data input[12] of a mesh construction or a mesh modification procedure. In practice, there are two types of problem regarding surface meshes.

The classical part of the adaptive mesh construction scheme consists of constructing an initial mesh of the domain. This is done with no special knowledge, *i.e.*, of a metric nature related to the physical behavior of the problem. The sole properties used are those related to the geometry. The surface mesh results from a geometric modeler or by using an appropriate surface mesh method. However, this mesh must be reasonable (meaning that it is representative of the underlying geometry, Chapter 19), but it is probably not ideal with regard to the physics included in the P.D.E. problem in hand, which therefore justifies the use of an adaptive approach.

In a local approach, the surface is remeshed so as to take into account the given metrics and the geometric metric. The basic idea is always the same, and aims at

[11] This is strictly the case for an advancing-front or a Delaunay type method but this is not necessarily the case for a classical *octree* method where the surface mesh and the volume mesh can be constructed at the same time.

[12] See the previous note.

constructing unit length edges. The remeshing process, discussed in Chapter 19, makes use of geometrical operations (point relocations) or topological operations (edge swaps, edge collapsing, point insertion, etc.). In brief, remeshing a surface is seen as an optimization procedure. The information which is then pertinent concerns :

- the proper location of a point on the surface,
- the access to the properties of the surface (normals, tangents, principal radii of curvature, etc.).

Note that the local surface remeshing can be made at the same time that the local remeshing of the volume is made.

In a global approach, the surface mesh is constructed in a stand-alone step with no connections with the volume mesh. See Chapters 14 and 15 for a detailed discussion on mesh generation methods for curves and surfaces.

About the mesh construction. Using an initial surface mesh as data, the volume mesher constructs an initial mesh in the domain. It is clear (following the mesh generation methods described in this book) that the quality of this three-dimensional mesh is strongly related to the quality of this surface mesh. If a metric map (obtained using an error estimate after the solution analysis) is available, it is possible to numerically evaluate whether the current mesh is satisfactory or not. If it is not judged good enough, the mesh generation process is iterated thus leading to the construction of a new mesh taking into account the geometric metric map together with the physical metric map. Then, the current mesh becomes the background mesh for the next iteration step.

As for the surface meshing case, two types of methods can be envisaged to obtain adapted volume meshes. A local approach makes use of the optimization operators, as we have already seen. In a global approach, the new mesh is entirely recreated using the metric map defined at the vertices in the background mesh. The mesh generation is then governed, meaning that we aim to construct a mesh with unit length edges.

About the solution step. The solution step comprises a solution method (the method used to solve the resulting matrix system) and an error estimate to access the quality of the solution that is computed. This estimate also serves to translate this analysis in terms of directives that are directly usable by the mesh generation method.

In a global adaptation scheme, it is advisable to use an iterative solution method in which the solution which is sought is initialized by the solution obtained at the previous iteration step. Therefore, interpolation methods must be used to interpolate the solutions from mesh to mesh. This raises the problem of how to *transport the solution* from the background mesh to the current mesh. These methods involve finding the position of a vertex in the current mesh in the background mesh and are thus based on localization procedures (Chapter 18).

About the metric map. As previously indicated, the metric map is a discrete set of values (tensors) which are usually known at the vertices in the background mesh. When several metrics are specified at these vertices, an intersection procedure (Chapter 10) is used to find a unique metric. A continuous map is then obtained by (linear) interpolation using the vertex values.

21.6 Application examples

In this section, we show some examples of adapted meshes corresponding to various iteration steps in an adaptive process. In all these examples, the approach is a global one for the domain meshing (planar or volume domain) and a local approach (by remeshing) for the boundary (curves or surfaces) meshing.

The first series of examples (Figure 21.14) corresponds to a mechanical device in two dimensions. The mesh generation method is a *quadtree* type method. In this example, the metric map is isotropic, the sizing function is defined in an analytic manner using two real valued functions :

$$h1(x,y) = 4|r_1 - ||\overrightarrow{C_1P}||| + 0.05 \quad \text{and} \quad h2(x,y) = 2|r_2 - ||\overrightarrow{C_2P}||| + .02\,, \quad (21.15)$$

where $C_1 = (-3,5)$ and $C_2 = (10,-5)$ are the center-points of two discs whose radii are respectively $r_1 = 7$ and $r_2 = 13$. Ten iteration steps for mesh construction were necessary to capture the analytical map in a satisfactory manner.

mesh - iter	np	ne	Q_T	$\overline{Q}$	l_{min}	l_{max}	l_{avg}	τ
part - 0	2,204	3,794	2.9	1.29	0.24	8.33	0.75	0.88
part - 1	2,808	4,975	2.9	1.28	0.21	21.13	0.77	0.89
part - 3	5,090	9,502	3.64	1.27	0.24	29.9	0.75	0.90
part - 10	7,772	14,847	2.9	1.27	0.24	1.78	0.74	0.92

Table 21.1: Statistics about the numerical evaluation of adapted meshes by means of a quadtree type method.

Table 21.1 gives the results of the adaptation for this example. The values np and ne are the number of vertices and the number of elements in the mesh, Q_T and $\overline{Q}$ note the worst and the mean qualities of the triangles. The minimum, maximum and mean edge lengths are respectively denoted by l_{min}, l_{max} and l_{avg} while τ is the efficiency index.

Remark 21.20 *Note that this type of method (based on a tree) favours the creation of small edges (in the metric) and, more specifically, in this analytic example in which the mesh gradation (by means of the [2:1] rule) is not really compatible with the given metric map. Nevertheless, the spatial tree decomposition has captured the required map, as can be seen by observing the value of index τ.*

The second example concerns a C.F.D. case, in two dimensions. The problem deals with a viscous calculus around a NACA012 type profile at Mach 0.95 with

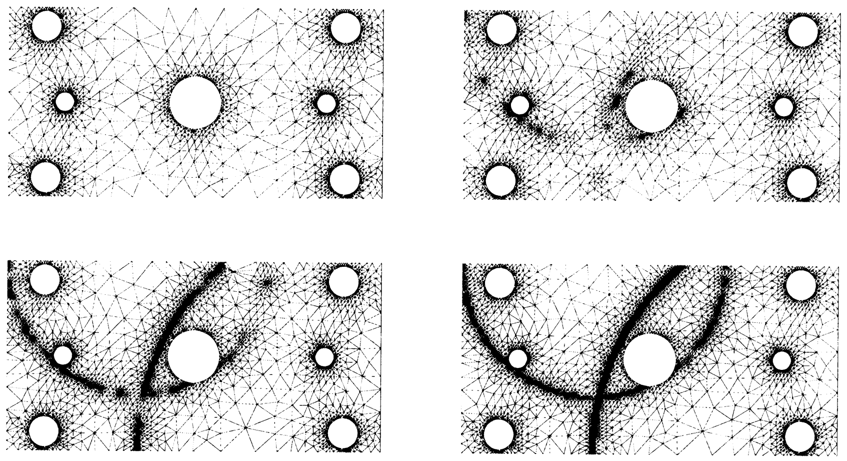

Figure 21.14: *Adapted meshes of a mechanical device using a quadtree type method (iteration steps 0, 1, 3 and 10) for an analytical isotropic metric map.*

a Reynolds 5,000. A characteristic configuration of the *fish tail* is sought with an instationarity due to shocks-wakes interactions. The mesh adaptations allow the shock regions, the boundary layers and the wake (Figure 21.15) to be captured. The density iso-contours are shown in Figure 21.16. Six adaptation steps were used to capture the physics. A type method (with a point insertion scheme using the edges of the current mesh, Chapter 7) was used for the global remeshing at each iteration step and a Navier-Stokes solution method was used.

To conclude, we give an example of adapted meshes in three dimensions. The shape of the domain is deliberately simple[13], a sphere with a unit radius centered at the origin. The isotropic metric map used here is analytic and also concerns the domain surface. Figure 21.17 shows the surface meshes at steps 0, 1 and 7 in the adaptation. Figure 21.18 shows the volume meshes (and some cuts by various planes) at the corresponding steps.

An isotropic mesh optimization procedure was used for the surface remeshing [Frey,Borouchaki-1998]. A Delaunay method was employed to adaptively remesh the volume [George,Borouchaki-1998].

Table 21.2 gives the main figures for the different iteration steps. The values $l_{\%}$ denote the percentage of edges whose lengths are compatible with the size map (*i.e.* such that $\sqrt{2}/2 \leq l \leq \sqrt{2}$), Q_T, $1-2$ and $2-3$ denote the worst quality in the mesh and the number of elements with a quality between 1 and 2 and between 2 and 3, respectively.

[13]Note that the visualization of map in three dimensions is delicate or even not really possible in the case of arbitrary complex geometries.

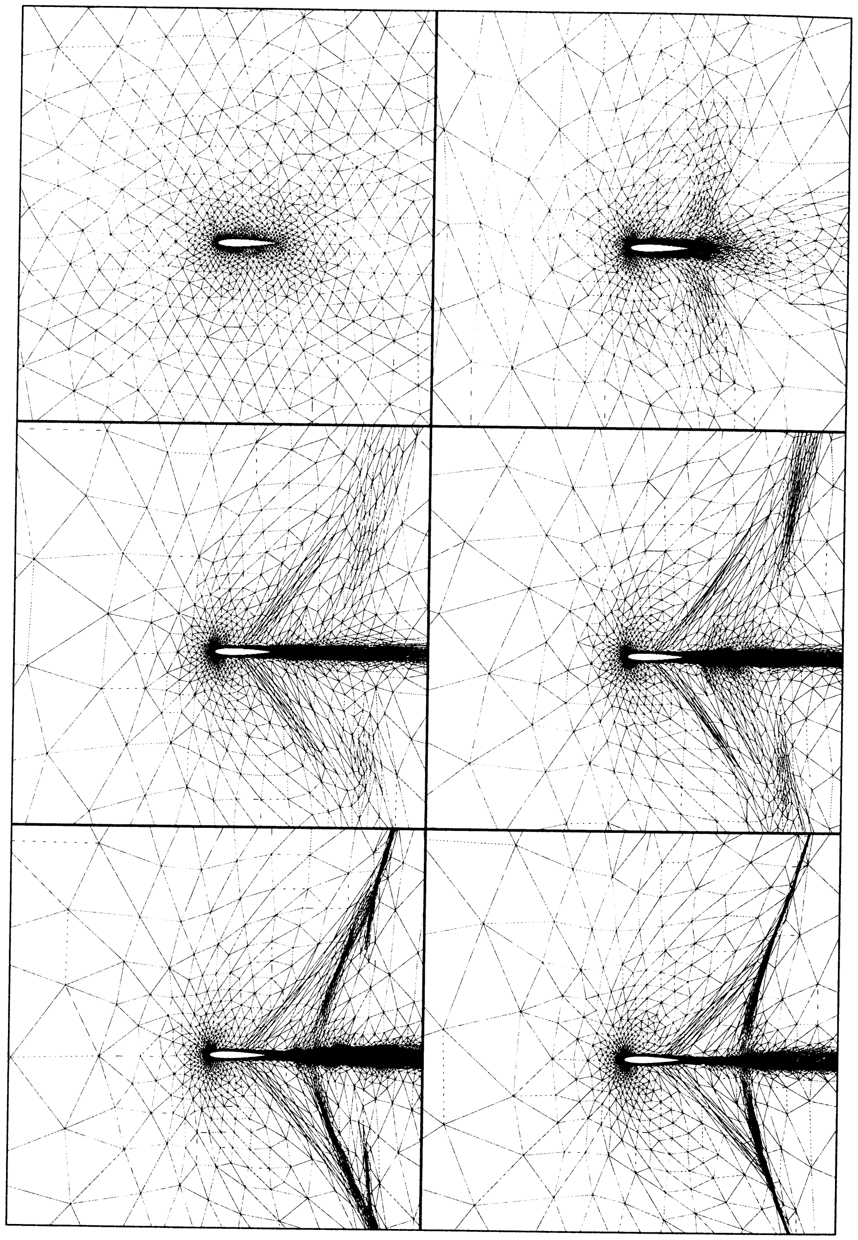

Figure 21.15: *Original mesh (top left-hand side) and adapted meshes using a (anisotropic) Delaunay type method around a wing profile (NACA012) at iteration steps 0, 1, 2, 3, 4 and 6 for a Navier-Stokes computation in CFD.*

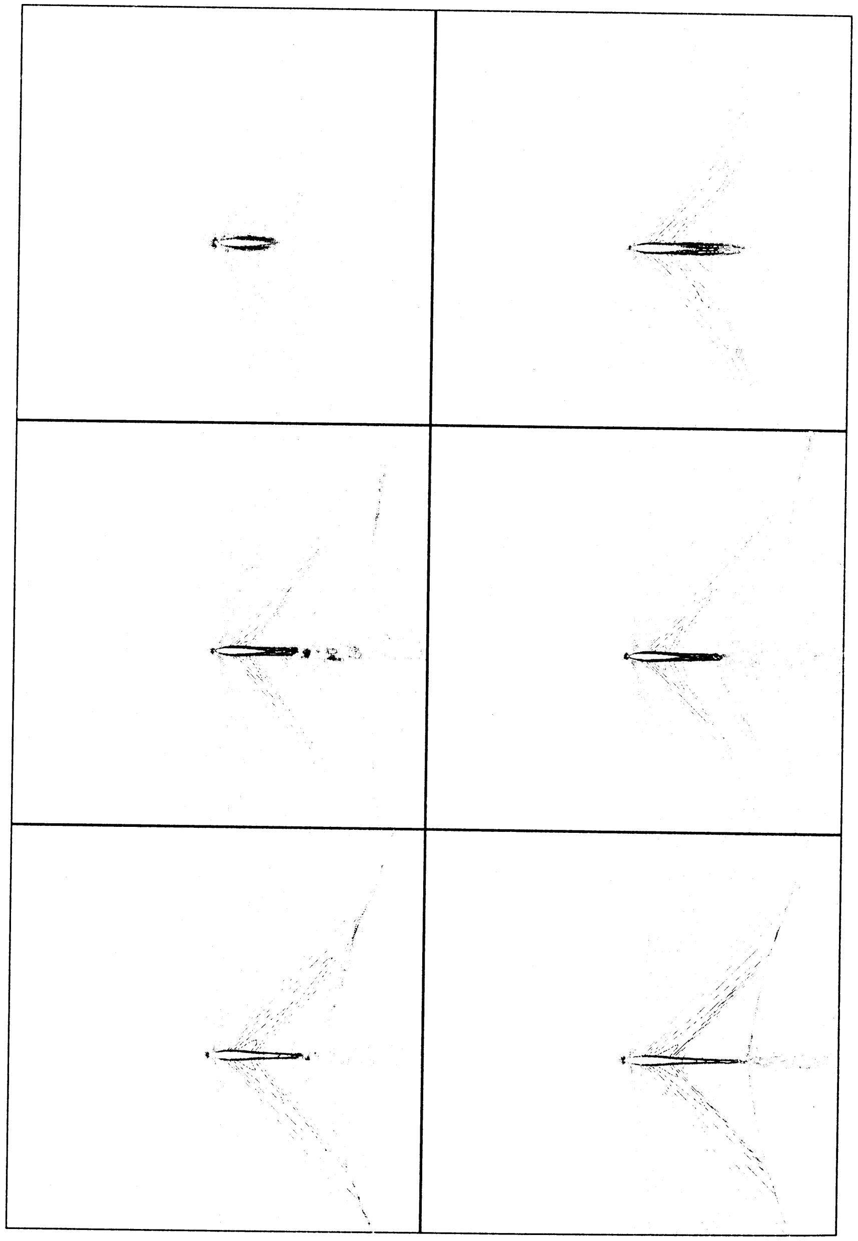

Figure 21.16: *Density iso-contours corresponding to the adapted meshes at iteration steps 0, 1, 2, 3, 4 and 6.*

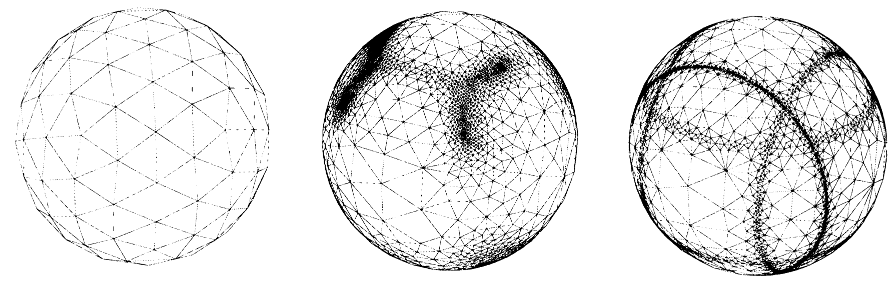

Figure 21.17: *Adaptation examples in three dimensions. Adapted surface meshes at iteration steps 0, 1 and 7.*

-	np	ne	τ	$l_{\%}$	Q_T	$1-2$	$2-3$
Initial mesh	277	1,200	0.515	7	1.8	100	-
Iteration 1	23,023	124,362	0.814	37	47.	81	11
Iteration 3	115,215	647,119	0.944	78	12.	78	20
Iteration 7	253,068	1,416,617	0.961	86	8.	74	24

Table 21.2: Statistics about the different iteration steps.

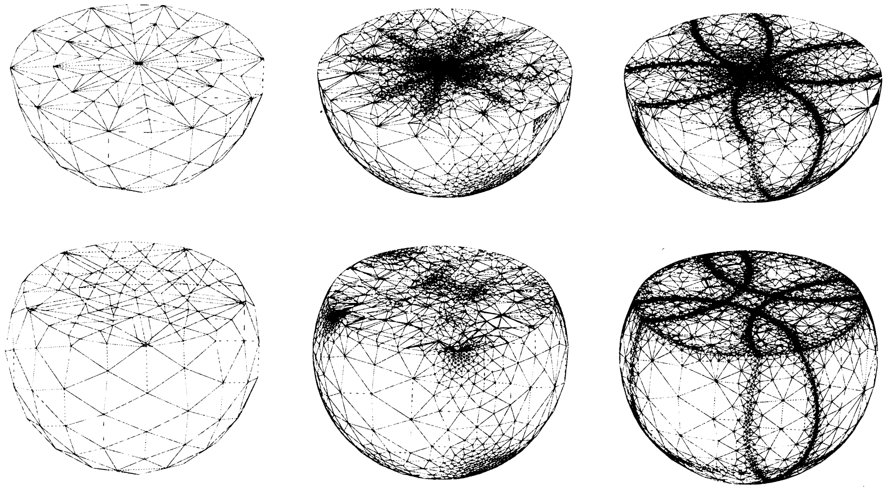

Figure 21.18: *Adaptation examples in three dimensions. Adapted volume meshes at iteration steps 0, 1 and 7, cuts by the plane $z = 0$ (top) and by the plane $z = 0.5$ (bottom).*