sub-meshes so as to have the sub-domains defined. Various techniques have been proposed for this purpose (cf. [Simon-1991], [Farhat,Lesoinne-1993] for instance).

The main drawback of such a method is its memory requirement. In fact, it is necessary to store the initial mesh and, at least, one of the sub-meshes. Moreover, all the problems related to any partitioning methods must be addressed (load balancing, interface smoothness, etc.).

***A priori* partitioning.** The purpose of this approach is to construct an *a priori* partition of the domain, from the data of a coarse mesh of it or directly from a discretization of the domain boundaries. Once this partition is available, the resulting sub-domains are meshed in parallel thus taking advantage of the parallel capabilities of the computers right from the meshing stage.

The main difficulties in this approach are related to the load balancing aspect (that must be deduced from the coarse mesh or the domain boundary) and to the proper management of the domain interfaces. The coarse mesh may be an empty mesh (without internal vertices), for instance resulting from a Delaunay method. The interface between two sub-domains is constructed either from the data of the coarse mesh or else from the data of the domain boundary discretization.

Chapter 4

Algebraic, P.D.E. and multiblock methods

Introduction

This chapter describes some algebraic methods, some methods based on the solution of P.D.E.s as well as multiblock-type methods. An algebraic method is designed to carry out the mesh construction of domains having an analogy with a simple shaped logical domain (such as a square or a quadrangle, a triangle, etc.) while a P.D.E.-type method is designed to handle domains that can be mapped onto a square (a cuboid in three dimensions) using different kinds of analogies. These methods are therefore limited regarding the shape of the domains they can successfully deal with. A multiblock type method is one possible solution to carry out arbitrarily shaped domains. First, the domains are decomposed into simply shaped regions where the previous methods can be used. Then, the mesh of the entire domain is obtained as the union of the local meshes corresponding to the above regions.

The first section discusses various algebraic methods based on a mapping function which is defined *a priori*. The second section briefly considers P.D.E. style methods where the mesh is obtained by solving an adequate system of differential equations. The third section shows how to define a multiblock method using one of the above methods as a local meshing process.

4.1 Algebraic methods

Any algebraic mesh generation method consists of constructing a mesh on a (real) domain using a given function that is explicitly defined. Main references about algebraic methods, mostly for quad or hex geometries, include [Gordon,Hall-1973], [Cook-1974] and, for transfinite interpolation style methods suitable for simple

shapes, a recent synthesis by [Perronnet-1998]. The given function is used to map an easily defined mesh in a logical domain $\hat{\Omega}$ having a simple shape (unit square, unit triangle, etc.) onto the real domain Ω (Figure 4.1). The so-created logical mesh is in essence a structured mesh. In general, the mapping function consists of polynomials defined in such a way as to ensure certain properties.

In this section, we assume that a suitable domain boundary discretization is supplied as input data and we show how to construct a mesh covering the domain thus defined. In two dimensions, the boundary discretization consists of a series of segments (a polygonal line) enjoying some specific properties. In three dimensions, this discretization is a surface mesh which also conforms to a peculiar type (see below).

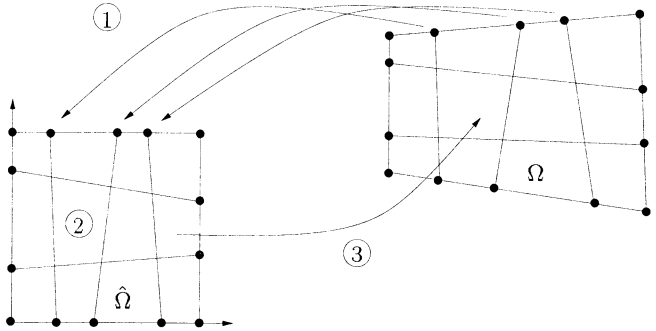

Figure 4.1: *General principle of any algebraic method in the case where the domain Ω is seen as a quad. Three steps are involved, the data mapping on the boundary, the mesh construction into the reference square and the mapping of this mesh onto the actual domain (steps denoted by 1, 2 and 3 in the figure).*

4.1.1 Trivial mapping functions

It seems natural to use as the mapping function the shape function corresponding to the shape of the domain (the mapping function is then the shape function of the finite element with the corresponding geometry). Thus, in the case where the domain "looks like" a triangle, the mapping function can be defined by the formula :

$$F(\xi,\eta) = (1-\xi-\eta)\,a_1 + \xi\,a_2 + \eta\,a_3\,, \tag{4.1}$$

where a_i is the *corner* with index i (see hereafter) of the real domain. Obviously such a function only matches the corners of the domain. In other words, given a mesh on the logical triangle, the resulting mesh does not match the given boundary discretization (unless the latter is composed of straight lines). As indicated, the above function is the shape function of the classical P^1 triangle (Chapter 20).

To improve the accuracy of the approximation of the domain, a more complicated shape function can be used, *i.e.,* by reference to a more sophisticated finite

element. For instance, the mapping function :

$$\begin{aligned} F(\xi,\eta) = \; & (1-\xi)(1-\eta)(1-2\xi-2\eta)\,a_1 + \xi(1-\eta)(2\xi-2\eta-1)\,a_2 \\ & - \xi\eta(3-2\xi-2\eta)\,a_3 + (1-\xi)\eta(-2\xi+2\eta-1)\,a_4 \\ & + 4\xi(1-\xi)(1-\eta)\,a_5 + 4\xi\eta(1-\eta)\,a_6 + 4\xi(1-\eta)\eta\,a_7 \\ & + 4(1-\xi)\eta(1-\eta)\,a_8\,, \end{aligned} \tag{4.2}$$

insures the respect of a boundary composed by arcs of parabola for a domain (Figure 4.2) similar to a quad (a_i are the corners while a_{i+4} are the edge midpoints, for $i = 1, 4$).

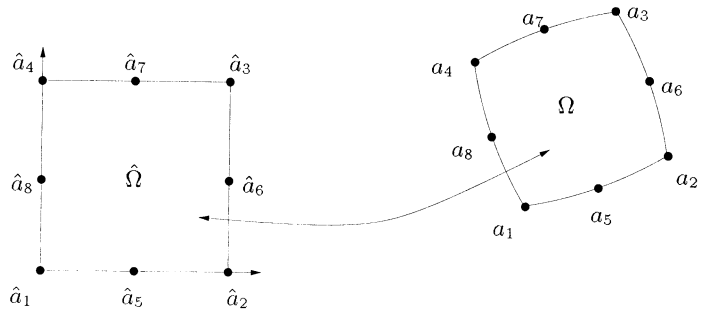

Figure 4.2: *Quadrilateral domain whose boundary is approximated by a number of arcs of parabola.*

Since the above functions, as well as the similar functions that can be easily found for the other shape analogies, do not guarantee sufficient properties, other types of functions must be developed.

4.1.2 Quadrilateral analogy

In this case, the given discretization of the real domain can be considered as a set of four logical sides, each of which consists of a series of segments. Thus, an analogy with a quadrilateral is exhibited. The endpoints of the sides are the so-called *corners*, $a_{i_{i=1,4}}$, defined counterclockwise. Similarly, the domain is assumed to be on the "left-hand side" of the boundary which is also defined counterclockwise. The first side, defined from a_1 to a_2, consists of a series of n_1 segments. The second side, from a_2 to a_3 includes n_2 segments. For the sake of simplicity, the third side (a_4, a_3) is formed by n_1 segments while the fourth side (a_1, a_4) has n_2 segments. This means that the number of segments of the side discretization is the same for two (logically) opposite sides. In other words, two opposite sides must enjoy some similarity (for instance in terms of length) so that only two *subdivision parameters* (n_1 and n_2) can be used.

Let $\phi_i(.)$ be the discretization of side i. In fact, $\phi_i(.)$ is defined as a series of straight segments joining two consecutive points of the given boundary discretization.

Logical mesh on the unit square boundary. Let a_j^i be the j^{th} point serving to define the discretization of the real side i with a_0^i the first corner of side i and a_n^i the other endpoint this side (where n stands for n_1 or n_2), then one can construct a discretization of side i of the unit square in such a way as to conform to the discretization of the real corresponding side, in terms of relative distances from point to point.

Let (omitting index i associated with the side under treatment) $l_j = l_{a_j,a_{j+1}}$ be the distance between a_j and a_{j+1} for $j = 0, 1, ..., n-1$, then

$$l_{side} = \sum_{j=0}^{n-1} l_j \,,$$

with this notation, the j^{th} point on the logical side, say $\hat{a}_j$, is constructed on the relevant side by means of a formula like

$$\hat{a}_j = \frac{\sum_{k=0}^{j-1} l_k}{l_{side}} \,.$$

This process is then repeated for the four sides, resulting in the desired points $\hat{a}_j^i \,, i = 1, 4$.

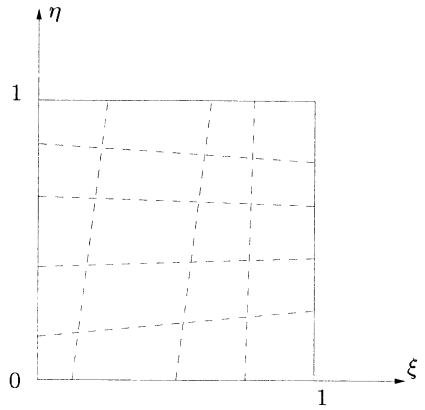

Figure 4.3: *Logical mesh on the unit square. The doted lines show the two families of lines serving as a support for the construction.*

Logical mesh on the unit square. The above $\hat{a}_j^i$ are now used to define a simple mesh on the unit square. We define the lines joining two opposite points. Then considering a line joining side 1 and side 3 together with a line joining line 2 and side 4, an intersection point can be found. Actually, the intersection points result from the intersection of two families of lines. Applying this process for all the lines results in a valid mesh in the unit square. Both the connectivities (*i.e.*, the element vertex connectivities and numbering) and the vertex positions of this logical mesh are trivially obtained.

Mapping onto the real domain. The aim is now to map the above logical mesh onto the real domain. To this end, a suitable mapping function, F, is needed. The question is how to construct such a function. Indeed, several choices are possible leading to different mesh generation methods. To help the choice, one has to define what properties must be ensured. In this respect, we introduced three categories of properties which concern some suitable invariances and some degree of regularity.

- Pr_1 : The image of a logical corner is a real corner. This is the well-known *corner identity*. For instance, $F(0,0) = a_1$.
- Pr_2 : The image of a logical side matches the corresponding real side. Thus, $F(\xi, 0) \equiv \phi_1(\xi, 0) = \phi_1(\xi)$ (and similar expressions for the other sides).
- Pr_3 : The image of a regular logical mesh is a regular real mesh. In fact, this means that a regular mesh is obtained for a real domain whose boundary discretization is uniform (for the two possible pairs of opposite sides). This leads to an *invariant property* of F when applied to the logical mesh itself. Then, $F(\xi, \eta) = (\xi, \eta)$, point of coordinates ξ and η is mapped onto itself.

We initially restrict ourselves to simple polynomials (in fact, forms of polynomials of first order for each of the variables). Then one possible method corresponds to :

$$F(\xi, \eta) = \frac{\sum_{i=1}^{4} \alpha_i \, \phi_i(\xi, \eta)}{\sum_{i=1}^{4} \alpha_i} \,, \tag{4.3}$$

where ξ and η live in $[0, 1]$. The α_i's are functions of ξ and η defined as :

$$\alpha_1 = (1-\eta)\frac{1-\xi}{(1-\xi+\eta)}\frac{\xi}{(\xi+\eta)} \,, \qquad \alpha_2 = \xi\frac{\eta}{(1-\xi+\eta)}\frac{1-\eta}{(2-\xi-\eta)} \,,$$

$$\alpha_3 = \eta\frac{1-\xi}{(2-\xi-\eta)}\frac{\xi}{(1+\xi-\eta)} \,, \qquad \alpha_4 = (1-\xi)\frac{\eta}{(\xi+\eta)}\frac{1-\eta}{(1+\xi-\eta)} \,.$$

Remark 4.1 *Note that the sum*[1] *of the α_i's is 1. Thus in the previous formula, the denominator can be removed.*

For this function, Pr_1 and Pr_2 are satisfied while Pr_3 does not hold. Thus, another type of function may be sought. For instance, the function

$$F(\xi, \eta) = \frac{\sum_{i=1}^{4} \beta_i \, \phi_i(\xi, \eta)}{\sum_{i=1}^{4} \alpha_i} \,, \tag{4.4}$$

[1] A system like Mapple can be used to verify this property.

where now :

$$\beta_1 = (\alpha_1 + \alpha_3)(1-\eta), \qquad \beta_2 = (\alpha_2 + \alpha_4)\,\xi,$$
$$\beta_3 = (\alpha_1 + \alpha_3)\,\eta, \qquad \beta_4 = (\alpha_2 + \alpha_4)(1-\xi).$$

implies that Pr_1, Pr_2 along with Pr_3 hold (as can be easily verified).

Remark 4.2 *Obviously, the sum of the β_i's is also equal to one.*

As indicated, with this method, Pr_1, Pr_2 along with Pr_3 hold.

Proof. First, it is obvious to see that Pr_2 holds. For instance, for $\eta = 0$, $\alpha_2 = \alpha_4 = 0$, then $\beta_2 = \beta_4 = 0$ and $\beta_3 = 0$. Thus, $F(\xi, 0) = \beta_1\,\phi_1(\xi, \eta)$. As $\beta_1 = \alpha_1 + \alpha_3$, then $\beta_1 = \sum \alpha_i - \alpha_2 - \alpha_4 = 1$ that implies that $F(\xi, 0) = \phi_1(\xi, 0) = \phi_1(\xi)$.

Now, obviously Pr_1 holds. For instance, if $\xi = 0$, the above relation leads to $F(0, 0) = \phi_1(0) = a_1$.

Property Pr_3 leads to checking whether the image of point (ξ, η) is this point when the real domain is the logical domain. So, we have to compute $\sum_{i=1}^{4} \beta_i\,\phi_i(\xi, \eta)$. In this particular case we have $\phi_1(\xi) = \begin{pmatrix} \xi \\ 0 \end{pmatrix}$, $\phi_2(\eta) = \begin{pmatrix} 1 \\ \eta \end{pmatrix}$, $\phi_3(\xi) = \begin{pmatrix} \xi \\ 1 \end{pmatrix}$ and $\phi_4(\eta) = \begin{pmatrix} 0 \\ \eta \end{pmatrix}$, then :

$$\sum_{i=1}^{4} \beta_i\,\phi_i(\xi, \eta) = \beta_1 \begin{pmatrix} \xi \\ 0 \end{pmatrix} + \beta_2 \begin{pmatrix} 1 \\ \eta \end{pmatrix} + \beta_3 \begin{pmatrix} \xi \\ 1 \end{pmatrix} + \beta_4 \begin{pmatrix} 0 \\ \eta \end{pmatrix}.$$

The first component of this expression is $\beta_1\,\xi + \beta_2 + \beta_3\,\xi$ whose value is :

$$\xi\,(1-\eta)\,(\alpha_1+\alpha_3) + \xi\,(\alpha_2+\alpha_4) + \xi\,(\alpha_1+\alpha_3)\,\eta = \xi\,(\alpha_1+\alpha_3) + \xi\,(\alpha_2+\alpha_4) = \sum_{i=1}^{4} \alpha_i\,\xi,$$

that is ξ. Similarly, the second component is η, thus :

$$\sum_{i=1}^{4} \beta_i\,\phi_i(\xi, \eta) = \begin{pmatrix} \xi \\ \eta \end{pmatrix}.$$

and the proof is completed. □

Another popular method is the *transfinite interpolation* which is defined in a slightly different manner. In fact, the corresponding function is not only a combination of the ϕ_i's but involves such a combination coupled with a correction term based on the corners. Note that a tensor product is used to define the main part of the expression and the correction term is then added to meet the desired properties :

$$\begin{aligned} F(\xi, \eta) \;=\; & (1-\eta)\,\phi_1(\xi) + \xi\,\phi_2(\eta) + \eta\,\phi_3(\xi) + (1-\xi)\,\phi_4(\eta) \\ & - ((1-\xi)(1-\eta)\,a_1 + \xi(1-\eta)\,a_2 + \xi\eta\,a_3 + (1-\xi)\,\eta a_4). \end{aligned} \tag{4.5}$$

Exercise 4.1 *Show that function F satisfies the three above properties (Hint : follow the scheme of the above proof).*

The mesh of the real domain is then easily obtained. Its connectivity is that of the logical mesh. The coordinates of its vertices are known by applying F to the above $\hat{a}_j^i$'s.

The question is to decide which function F is the best (4.3, 4.4 or 4.5). Clearly the function of Relation 4.3 is probably not so good. To select one of the two others, we have to look at the limitations of the method.

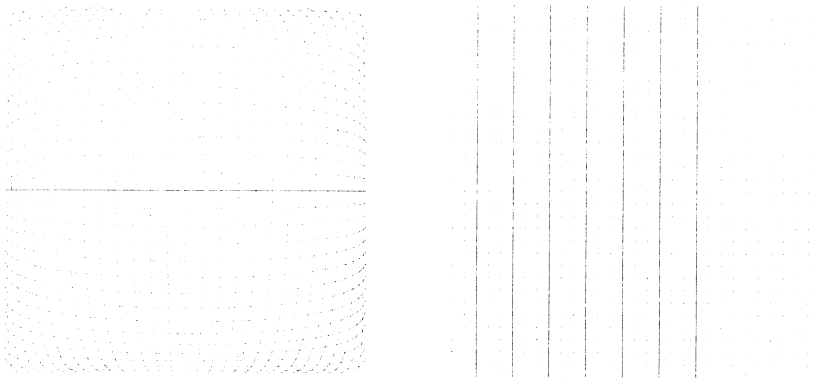

Figure 4.4: *Image of a uniform unit square when using the function of Relation (4.3), left-hand side, and that of Relation (4.4) or (4.5), right-hand side.*

Limitations. Unfortunately, the above method is unlikely to be suitable when handling complex (four sides) geometries. Actually, if the real domain is convex, the resulting mesh is valid while for some non convex domains, two complications may arise. First, the image of a point inside the logical mesh may fall outside the real domain and, on the other hand, the image of a valid quadrilateral in the logical mesh may be a quadrilateral with a null or negative surface area, thus resulting in overlapping elements and an invalid mesh.

In other words, such a method is suitable only under some restrictive conditions about the shape of the real domain (in addition to its quadrilateral analogy).

Nevertheless, we would like to give some examples. In Figure 4.4, we consider how the three above functions act with regard to Pr_3. Then, as previously mentioned, Relation (4.3) is unlikely to be suitable. In Figure 4.5, we show the mesh obtained using methods (4.4) and (4.5) for a non convex domain. Both meshes are wrong but it seems that method (4.4) is less sensitive to the non convex geometry. Nevertheless, other non convex geometries seem to indicate that the method of Relation (4.5) is more robust in most cases.

One can also observe that it is not possible to enforce some orthogonality properties in the element edges (in contrast to the P.D.E. methods briefly presented below).

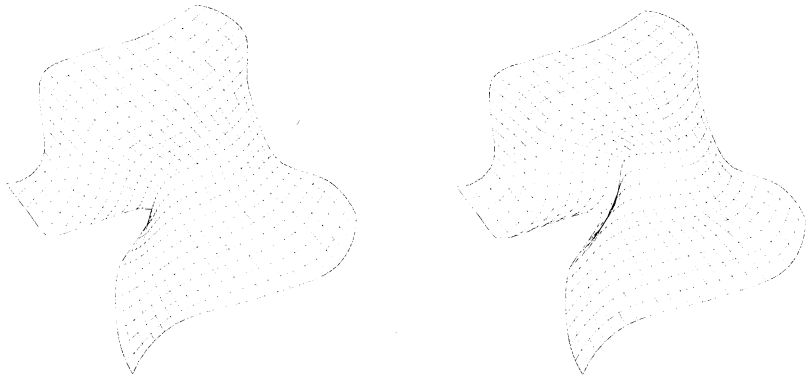

Figure 4.5: *Non convex domain meshed using the function of Relation (4.4), left-hand side, and that of Relation (4.5), right-hand side.*

A final comment about this mesh generation method concerns its rigidity. Indeed, as for all structured meshes, the type of connectivities from point to point results in a certain level of rigidity which means that it is not easy to deal with a problem where some flexibility (in terms of sizes, variations from region to region, etc.) is required.

Variations. The two parameters n_1 and n_2 allow for some flexibility in the method. The restriction leading to only two parameters being considered can be, to some extent, over-passed. Indeed, it is possible to define four parameters (one for each side discretization) and to define a more sophisticated method based on the same approach. Such a construction results in a mesh consisting of quadrilateral elements as well as some triangles to ensure a transition between the layers of elements.

Triangular meshing for a quadrilateral domain. In essence, this algebraic method completes a mesh whose elements are quadrilaterals (except in the case where four parameters are defined). To meet a triangular mesh, the quads must be split into two triangles (to maintain the same number of vertices). Such an operation is trivial at the time a criterion is selected to decide which diagonal is used to split a given element. In this respect, a given direction for the diagonals, the diagonal lengths, the element qualities, etc., can be chosen to govern the process. One should note that, in general, it is advisable to pay special attention to the elements that have one of the corners as a vertex. In this case, considering the diagonal that includes such a corner as endpoint is often a nice solution (see Chapter 18 about the notion of an over-constrained mesh).

Computational aspects. Regarding the computer implementation of the method (whatever the function F may be), several remarks can be made.

- First, one should note that the memory resources that are required can be easily known (or estimated) using the two (four) parameters defining the boundary discretization.

- The key-idea to complete the logical mesh is the intersection of two families of lines, thus no difficulties are expected as such intersections are well defined (in contrast to the case of a triangular analogy, as shown below).

- Applying Relation (4.i) ($i = 3, 5$) is easy at the time the terms $\phi(.)$'s are computed. Actually $\phi(.)$ is known in a discrete manner. For instance, $\phi_1(\eta, 0)$ is the polygonal line written as $(a_1^1, a_2^1)\,(a_2^1, a_3^1) \ldots (a_{n_1-1}^1, a_{n_1}^1)$. This implies using an interpolation scheme. Thus, the computational process can be as follows :

  - Given $\hat{M}$ a vertex in the logical mesh with coordinates ξ and η, we pick the interval $(\hat{a}_j^1, \hat{a}_{j+1}^1)$ within which falls ξ as well as the interval $(\hat{a}_k^3, \hat{a}_{k+1}^3)$ (in terms of the third logical side). Similarly we find the two intervals (sides 2 and 4) corresponding to η.
  - We find the relationships between ξ and $\hat{a}_j^1$ and $\hat{a}_{j+1}^1$ and we map the same ratio between a_j^1 and a_{j+1}^1 (say the discrete form of ϕ_1). Similarly we find the three other relationships for the three other sides.
  - Then, the desired function is used by replacing the terms $\phi_j(.,.)$ by the above interpolations.

- Null or negative surface area elements must be checked explicitly. To this end, associating four triangles with a given quad (*i.e.,* by using the two diagonals, each of them enabling to define two triangles) proves to be an efficient way to detect the cases where the method fails (see also Chapter 18).

As a consequence, implementing such a method, in its range of application, results in a rather inexpensive mesh generation method and, actually, is quite easy.

4.1.3 Triangular analogy

An analogy with a triangular shape is exhibited in the case where the given discretization of the real domain can be considered as a set of three logical sides (consisting of a series of segments). In addition, we assume that each side includes n segments (thus, only one parameter is defined). Then the algebraic mesh generation method follows the same principle as the previous one.

The boundary discretization is mapped onto the unit logical triangle. The logical corners $\hat{a}_1$, $\hat{a}_2$ and $\hat{a}_3$ are the points $(0,0)$, $(1,0)$ and $(0,1)$ and the three logical sides are defined as $\hat{a}_1 \longrightarrow \hat{a}_2$ for the first, $\hat{a}_2 \longrightarrow \hat{a}_3$ for the second and $\hat{a}_1 \longrightarrow \hat{a}_3$ for the third. Three families of lines are constructed. Points $\hat{a}_j^i$, $i = 1, 3$ are obtained by intersecting these lines. One should observe that now the desired intersections are not well defined in the sense that the three relevant lines do not define a point as a solution but a small region. Nevertheless, a point, for instance

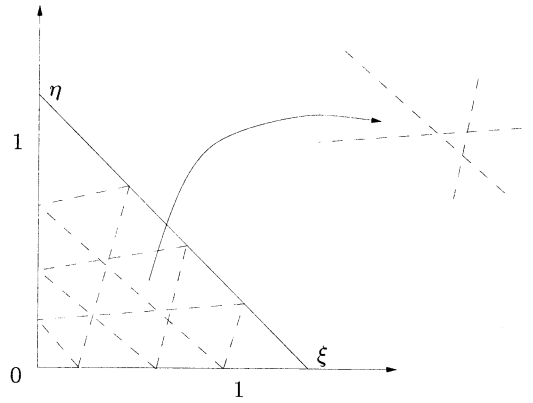

Figure 4.6: *Logical mesh on the unit triangle. In dotted lines are the three families of lines serving as a support for the construction. A close-up shows the region defined by the intersection of three lines that serves to find a unique intersection point.*

the centroid[2] of such a region, can be defined which allows construction we are seeking. In this way the logical mesh is obtained.

To map the above mesh onto the real domain, we use the same idea as for the above quad method. In fact, several categories of functions can be used. For instance, one may use a function like :

$$F(\xi,\eta) = \frac{\sum_{i=1}^{3} \alpha_i \, \phi_i(\xi,\eta)}{\sum_{i=1}^{3} \alpha_i}, \tag{4.6}$$

where the α_i's are the functions of ξ and η defined as :

$$\alpha_1 = (1+\eta)\frac{1-\xi-\eta}{(1-\xi)}\frac{\xi}{(\xi+\eta)},$$

$$\alpha_2 = (2-\xi-\eta)\frac{\xi}{(1-\eta)}\frac{\eta}{(1-\xi)},$$

$$\alpha_3 = (1+\xi)\frac{\eta}{(\xi+\eta)}\frac{1-\xi-\eta}{(1-\eta)}.$$

For this function, Pr_1 and Pr_2 are satisfied while Pr_3 does not hold.

Remark 4.3 *Note that the sum of the α_i's is 1.*

A form similar to Relation (4.5) is the following :

$$\begin{aligned} F(\xi,\eta) &= (1-\xi-\eta)\,(\phi_1(\xi)+\phi_3(\eta)) + \xi\,(\phi_1(\xi+\eta)+\phi_2(\eta) \\ &+\eta\,(\phi_3(\xi+\eta)+\phi_2(1-\xi)) \;-((1-\xi-\eta)\,\phi_1(0)+\xi\,\phi_2(0)+\eta\,\phi_3(0)), \end{aligned} \tag{4.7}$$

[2] In principle, the point minimizing the distances to the three lines is the best possible solution.

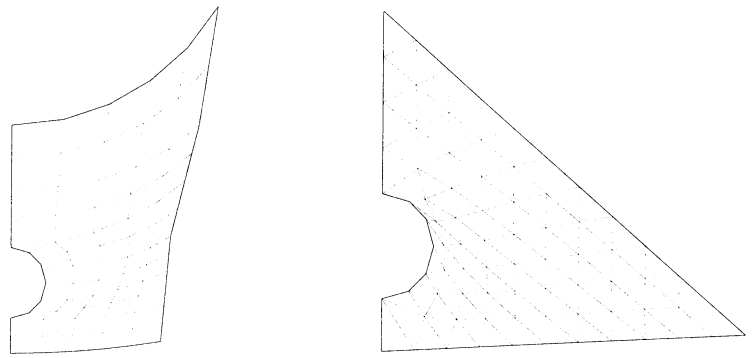

Figure 4.7: *Transfinite interpolation for a "quadrilateral" domain, left-hand side and same type of interpolation for a "triangular" domain, right-hand side.*

while a form involving rational polynomials is :

$$\begin{aligned} F(\xi,\eta) &= \tfrac{1-\xi-\eta}{1-\xi}\,\phi_1(\xi) + \tfrac{\xi}{1-\eta}\,\phi_2(\eta) + \tfrac{\eta}{\xi+\eta}\,\phi_3(\xi+\eta) \\ &-(\tfrac{\eta}{\xi+\eta}(1-\xi-\eta)\,a_1 + \tfrac{1-\xi-\eta}{1-\xi}\xi\,a_2 + \tfrac{\xi}{1-\eta}\eta\,a_3)\,. \end{aligned} \tag{4.8}$$

The previous functions satisfy the three desired properties. Limits and variations of the present method are of the same nature as for the above case. Computational issues do not lead to major difficulties.

4.1.4 Application examples

In this section, two simple application examples are provided where the domains in question are not strictly convex. Nevertheless, the resulting meshes are valid. The last example concerns a case where the (transfinite interpolation) method fails. Indeed, due to the geometry of the domain, some overlapping elements are constructed (*i.e.*, negative elements exist). To some extent, it is possible to modify such an invalid mesh so as to obtain a suitable solution. For instance, in this test example, by means of successive iterations of point relocations, the mesh can be corrected.

4.1.5 Surface meshing

The two above types of methods also provide a way to mesh a surface following the desired analogy which is defined by a discretization of its four (three) boundary edges. One should observe that the resulting surface is only controlled by its boundary discretization. Thus, while rather easy to implement, such a method may result in a poor approximation of the real surface or even may result in undesirable twists or folds (see Chapters 13 and 15 where the transfinite interpolation is used for surface definition).

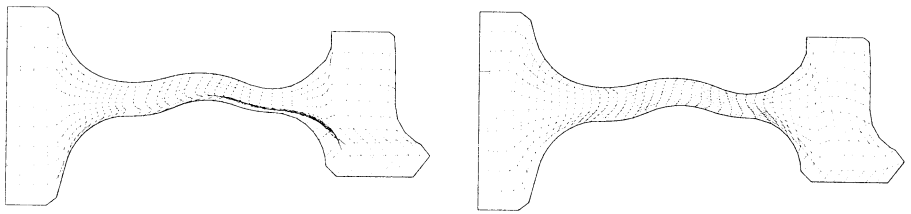

Figure 4.8: *Domain inducing degeneracies (overlapping elements are present), left-hand side, and mesh resulting from local corrections, right-hand side.*

4.1.6 Hexahedral analogy

Methods similar to those previously described can be defined so as to handle three-dimensional domains that can be considered as analogous to a simple logical domain. In this case, the real domain is assumed to be similar to a cuboid.

Actually, eight corners can be identified as well as six faces which are similar to quadrilateral faces. To follow a mesh generation method similar to those in two dimensions, the discretization of the faces is assumed to be of the structured type. For a quad face, a typical discretization is a grid defined by two subdivision parameters. For a triangular face (see hereafter the pentahedral or the tet analogies), only one subdivision parameter is supposed. Thus, provided $\phi_i(.,.,.)$, $(i = 1, 6)$, the adequate discretization of these faces, the logical mesh of the unit cube can be mapped on the real domain using the following function :

$$F(\xi,\eta,\zeta) = \frac{\sum_{i=1}^{6} \alpha_i \, \phi_i(\xi,\eta,\zeta)}{\sum_{i=1}^{6} \alpha_i}, \tag{4.9}$$

where the α_i's are the following functions of ξ, η, ζ :

$$\alpha_i = \overline{x}_{i-1} \frac{x_i}{(x_i + x_{i-1})} \frac{\overline{x}_i}{(\overline{x}_i + x_{i-1})} \frac{x_{i+1}}{(x_{i+1} + x_{i-1})} \frac{\overline{x}_{i+1}}{(\overline{x}_{i+1} + x_{i-1})},$$

and

$$\alpha_{i+3} = x_{i-1} \frac{x_i}{(x_i + \overline{x}_{i-1})} \frac{\overline{x}_i}{(\overline{x}_i + \overline{x}_{i-1})} \frac{x_{i+1}}{(x_{i+1} + \overline{x}_{i-1})} \frac{\overline{x}_{i+1}}{(\overline{x}_{i+1} + \overline{x}_{i-1})},$$

where $i = 1, 3$ with $x_{j+1} = \xi, \eta$ or ζ, $j = 0, 2$ ($i + j$ *modulo* 3) and $\overline{x}_j = 1 - x_j$. As before, this function satisfies the "corner identities" and matches the boundary.

Proof. First, one can verify the following properties (held for $i = 1, 3$ and $j = 1, 6$). First, the α_i's are such that :

- $x_{i-1} = 0 \longrightarrow \alpha_i = 1$ and $\alpha_j = 0$ for $j \neq i$,
- $x_{i-1} = 1 \longrightarrow \alpha_{i+3} = 1$ and $\alpha_j = 0$ for $j \neq i$.

Thus, we have :

- $F(0,0,0) = a_1$ where a_1 is the first corner and we have similar relations for the other corners (*i.e.*, property Pr_1 holds).
- $F(\xi,\eta,0) = \phi_1(\xi,\eta,0)$ where ϕ_1 stands for the discretization of the real face identified to the first face of the logical cube (*i.e.*, the face $\zeta = 0$), ... Then, property Pr_2 holds.
- $\alpha_i(\overline{x}_{i-1}) = \alpha_{i+3}(x_{i-1})$, $\alpha_i(\overline{x}_i) = \alpha_i(x_i)$ and $\alpha_i(\overline{x}_{i+1}) = \alpha_i(x_{i+1})$, then Relation (4.9) is symmetric in some sense, faces ϕ_i and ϕ_{i+3} act in a symmetric way when $\overline{x}_{i-1}$ replaces x_{i-1}. Indeed, we return to the discussion about Relation (4.4). While some symmetric features exist, property Pr_3 is not satisfied. □

As for the quad analogy, a method can be easily derived so as to ensure Pr_3. The idea is the same, we take as a function the following :

$$F(\xi,\eta,\zeta) = \frac{\sum_{i=1}^{6} \beta_i \, \phi_i(\xi,\eta,\zeta)}{\sum_{i=1}^{6} \alpha_i}, \tag{4.10}$$

with (for $i = 1, 3$) :

$$\beta_i = (\alpha_i + \alpha_{i+3})\, \overline{x}_{i-1}$$

$$\beta_{i+3} = (\alpha_i + \alpha_{i+3})\, x_{i-1}\,.$$

The transfinite interpolation for the hex analogy is as follows :

$$\begin{aligned} F(\xi,\eta,\zeta) \;=\; \frac{1}{2} \;& \big[(1-\zeta)\,\phi_1(\xi,\eta) + (1-\eta)\,\phi_2(\xi,\zeta) + (1-\xi)\,\phi_3(\eta,\zeta) \\ & + \zeta\,\phi_4(\xi,\eta) + \eta\,\phi_5(\xi,\zeta) + \xi\,\phi_6(\eta,\zeta) \\ & - \big((1-\xi)(1-\eta)(1-\zeta)\,a_1 + \xi(1-\eta)(1-\zeta)\,a_2 \\ & + \xi\eta(1-\zeta)\,a_3 + (1-\xi)\eta\zeta\,a_4 + (1-\xi)(1-\eta)\zeta\,a_5 \\ & + \xi(1-\eta)\zeta\,a_6 + \xi\eta\zeta\,a_7 + (1-\xi)\eta\zeta\,a_8\big)\big]\,. \end{aligned} \tag{4.11}$$

Then, Pr_1, Pr_2 as well as Pr_3 are satisfied.

Proof. First, it is obvious that Pr_1 holds. Regarding Pr_2, the proof needs to have the ϕ_i's defined by a transfinite interpolation. To show Pr_3, we use the same method as before. *I.e.*, we prove that the image of point (ξ,η,ζ) is invariant due to the particular form of the ϕ_i's. □

4.1.7 Pentahedral analogy

In this case, one can follow the same method with a mapping function like :

$$F(\xi,\eta,\zeta) = \frac{\sum_{i=1}^{5} \alpha_i \, \phi_i(\xi,\eta,\zeta)}{\sum_{i=1}^{5} \alpha_i} \tag{4.12}$$

where the α_i are defined by :

$$\alpha_1 = (1-\zeta)\frac{1-\xi-\eta}{(1-\xi-\eta+\zeta)}\frac{\xi}{(\xi+\zeta)}\frac{\eta}{(\eta+\zeta)},$$

$$\alpha_2 = (1-\xi)\frac{\zeta}{(\xi+\zeta)}\frac{1-\zeta}{(1-\zeta+\xi)}\frac{1-\xi-\eta}{(1-\eta)}\frac{\eta}{(\xi+\eta)},$$

$$\alpha_3 = (1-\eta)\frac{\zeta}{(\eta+\zeta)}\frac{1-\zeta}{(1+\eta-\zeta)}\frac{\xi}{(\xi+\eta)}\frac{1-\xi-\eta}{(1-\xi)},$$

$$\alpha_4 = \zeta\frac{\xi}{(1+\xi-\zeta)}\frac{\eta}{(1+\eta-\zeta)}\frac{1-\xi-\eta}{(2-\xi-\eta-\zeta)}$$

$$\alpha_5 = (\xi+\eta)\frac{\zeta}{(\zeta+1-\xi-\eta)}\frac{1-\zeta}{(2-\xi-\eta-\zeta)}\frac{\xi}{(1-\eta)}\frac{\eta}{(1-\xi)}.$$

One should note that $\alpha_1(1-\zeta) = \alpha_4(\zeta)$ and conversely, which means that faces one and four play a special role (in fact, they correspond to the logical faces $\zeta = 0$ and $\zeta = 1$). With this definition, Pr_1 and Pr_2 hold while Pr_3 is not achieved. Note that defining a function ensuring Pr_3 is a tedious task.

4.1.8 Tetrahedral analogy

For a tetrahedral analogy, one could retain the function :

$$F(\xi,\eta,\zeta) = \frac{\sum_{i=1}^{4} \alpha_i \, \phi_i(\xi,\eta,\zeta)}{\sum_{i=1}^{4} \alpha_i} \tag{4.13}$$

where the α_i are defined by :

$$\alpha_i = (1-x_{i-1})\frac{x_i}{(x_i+x_{i-1})}\frac{x_{i+1}}{(x_{i+1}+x_{i-1})}\frac{x_{i+2}}{(x_{i+2}+x_{i-1})}$$

for $i = 1,4$ with $x_{j+i} = \xi, \eta, \zeta$ or $1-\xi-\eta-\zeta$ for $j = 0,3$ ($i+j$ *modulo* 4). In fact, we meet the barycentric coordinates (that can be used in this case, as we are considering a simplex). In other words, we have :

$$\alpha_1 = (1-\zeta)\frac{\xi}{(\xi+\zeta)}\frac{\eta}{(\eta+\zeta)}\frac{1-\xi-\eta-\zeta}{(1-\xi-\eta)},$$

$$\alpha_2 = (1-\xi)\frac{\eta}{(\eta+\xi)}\frac{1-\xi-\eta-\zeta}{(1-\eta-\zeta)}\frac{\zeta}{(\zeta+\xi)},$$

$$\alpha_3 = (1-\eta)\frac{1-\xi-\eta-\zeta}{(1-\xi+\zeta)}\frac{\zeta}{(\zeta+\eta)}\frac{\xi}{(\xi+\eta)},$$

$$\alpha_4 = (\xi+\eta+\zeta)\frac{\zeta}{(1-\xi-\eta)}\frac{\xi}{(1-\eta-\zeta)}\frac{\eta}{(1-\xi-\zeta)}.$$

Thus, Pr_1 and Pr_2 hold while Pr_3 is not satisfied. Note that it is not possible to partition a tetrahedron into uniform sub-tetrahedra (unlike the same problem in two dimensions, for a triangle). Nevertheless, it could be of interest to have the corresponding uniform distribution of points. But, exhibiting a function ensuring Pr_3 is, *a priori*, not obvious.

4.1.9 Other algebraic methods

The above algebraic methods only infer point coordinates (in this sense, they are of a Lagrangian type). Thus, no way is provided to control directional features such as orthogonality (for instance, at the boundary level). Different algebraic methods can be investigated to permit such a control. For instance, a Hermitian type method can be defined involving not only point coordinates but also some derivatives. We will see in Chapter 13 the Coons patches which are based on a method that can also be used in this mesh generation context.

4.1.10 An alternative to algebraic methods

We consider in detail the case of a three-dimensional method. Apart for the hexahedral analogy, the two other shape analogies (tet and pentahedral) can be dealt with by the previously described algebraic method. Nevertheless, while simple in principle, these methods are not so easy to implement (and probably do not guarantee property Pr_3). Thus, a different mesh generation principle can be advocated.

To start the meshing process, we follow the first three steps of a classical algebraic method, *i.e.*,

- we construct a discretization on the boundary of the logical domain in accordance with the given discretization of the boundary of the real domain,
- we complete the mesh of the logical domain following the above boundary definition,
- we map the logical mesh onto the real domain by means of the classical P^1 interpolation scheme corresponding to the shape analogy.

The resulting mesh is then modified using a deformation technique. Let M be a point of the current mesh, we compute a new location for M as follows

$$M = M + \frac{1}{\alpha} \sum_{a_k \in \mathcal{B}} \omega_k(M) \, \alpha_k \, def(a_k) \tag{4.14}$$

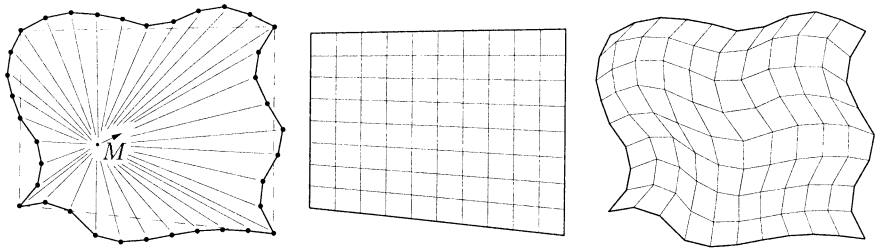

Figure 4.9: *Deformation governed by the boundary points, left-hand side. Straight mesh (middle) and resulting mesh, right-hand side.*

where the a_k's are the members of $\mathcal{B}$ the boundary of the domain (*i.e.,* the a_k's are the boundary vertices of the real domain) and the quantities involved in the formula are :

- $def(a_k)$, the distance between the (real) point a_k and the image of the corresponding $\hat{a}_k$,
- α_k is a weight associated with a_k. Actually, one can compute the average length of the edges sharing a_k,
- α, a normalization factor, which is defined as

$$\alpha = \sum_{a_k \in \mathcal{B}} \omega_k(M)\, \alpha_k \,, \tag{4.15}$$

- $\omega_k(M)$, a coefficient associated with a_k, is defined as $d_k^{-\beta}(M)$ where β is a value of attraction (for instance 4) and $d_k(M)$ is the distance between the image of $\hat{a}_k$ and M.

Now, note that $def(a_k)$ is zero for the corners[3].

Computational issues. While almost quadratic, the above algorithm has proved to be robust enough to carry out some non trivial geometries (nevertheless, a "too non convex" domain will be quite difficult to handle with success).

The theoretical analysis of this method is quite easy. Actually, each point creation needs the analysis of the members of $\mathcal{B}$. Thus, for instance for a hexahedral domain whose subdivision parameters are equal ($n_1 = n_2 = n_3 = n$), the cardinality of $\mathcal{B}$ is of the order of $(n+2)^2$, *i.e.,* something like n^2. While the number of points is something like n^3. Then, the number of operations is in n^5 resulting in a complexity in $n^{\frac{5}{3}}$, say $\mathcal{O}(n^{\frac{5}{3}})$, n being now the number of vertices. As a consequence, the method is in essence almost quadratic.

In Table 4.1, np is the number of vertices, ne is the number of elements, t is the CPU time in seconds while v_p indicates the number of points dealt with within one second and v_e the number of elements created in the same time. One

[3]Due to the use of the interpolation function which preserves the corner identity.

-	np	ne	t	v_p	v_e
cas 1	2,024	9,261	1.10	1,840	8,419
case 2	5,984	29,791	6.56	912	4,541
case 3	2,601	4,096	1.24	2,097	3,303
case 4	5,566	9,261	4.51	1,234	2,053
case 5	16,896	29,791	31.27	540	952
case 6	2,744	2,197	1.68	1,633	1,307
case 7	17,576	15,625	44.08	398	354

Table 4.1: Statistics related to some selected examples (the first two lines concern a tet analogy while the following three concern a pentahedral analogy and the last two concern a hexahedral analogy).

could observe the ratio $\left(\frac{np_2}{np_1}\right)^{\frac{5}{3}}$ between t_2 and t_1 where t_i $(i = 1, 2)$ is the time required for creating np_i points. While the behavior is the same for the three types, the global efficiency is related to the number of members in $\mathcal{B}$ compared with the number of points (thus the constant from case to case is rather different). Indeed, this constant is related to the number of face vertices of the logical element as compared with the number of vertices of the real mesh.

Extensions. One should note that the above meshing method by means of deformations is also a way to update a given mesh whose boundary discretization moves from one step to another of a given iterative process.

Remark 4.4 *The same method applies in two dimensions. In this case, it is very easy to define. Moreover, its complexity remains reasonable as it is something like* $n^{\frac{3}{2}}$, *say* $\mathcal{O}(n^{\frac{3}{2}})$.

4.2 P.D.E.-based methods

P.D.E. mesh generation methods represent an elegant alternative to algebraic methods and may be used when the domain (Ω with boundary denoted by Γ hereafter) can be identified by a quad (in two dimensions), or a cuboïd (in three dimensions). The major reference for P.D.E. type methods is [Thompson *et al.* 1985]. Contrary to any algebraic methods, a transformation from the domain to this quad (cuboid), the logical domain, is sought. A *generation system* is associated with such a transformation, which allows us to compute the required mesh.

4.2.1 Basic ideas

In what follows, variables $x, y,$ (resp. x, y, z) describe the domain (Figure 4.10) while the logical region is described using variables $\xi, \eta,$ (ξ, η, ζ). The problem

now becomes one of finding the functions

$$x = x(\xi, \eta) \quad \text{and} \quad y = y(\xi, \eta),$$

or

$$x = x(\xi, \eta, \zeta), \quad y = y(\xi, \eta, \zeta), \quad \text{and} \quad z = z(\xi, \eta, \zeta),$$

according to the spatial dimension, assuming that the transformation maps the logical region one-to-one onto the domain and that the boundaries are preserved.

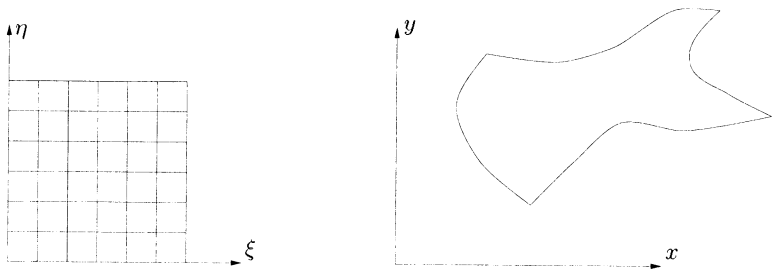

Figure 4.10: *Logical domain (a unit square), left-hand side, and real domain, right-hand side.*

The one-to-one property is ensured by requiring that the Jacobian of the transformation is non-zero. The transformation (for example in two dimensions) is defined by the matrix :

$$[\mathcal{M}] = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}$$

where x_ξ stands for $\frac{\partial x}{\partial \xi}$, x_η stands for $\frac{\partial x}{\partial \eta}$, and so on. The Jacobian J is $x_\xi y_\eta - x_\eta y_\xi$. As it is assumed to be non-zero, the inverse of the transformation exists and variables ξ, η can be expressed in terms of x, y as follows :

$$\xi = \xi(x, y) \quad \text{and} \quad \eta = \eta(x, y)$$

The two ways of expressing the variables are mathematically equivalent and lead to two possibilities for solving the problem. If variables ξ, η are expressed in terms of x, y, the logical mesh can be transformed into a mesh on the domain, and the physical problem is solved in the domain as in the classical way. On the other hand, if variables x, y are expressed in terms of ξ, η, either the physical problem can be written in terms of these variables and then solved in the logical region, or we return to the above classical solution.

As a simple example of P.D.E. methods, we briefly consider the following generation system based on the regularizing properties of the Laplacian operator Δ. We consider the two following systems :

$$\begin{cases} \xi_{xx} + \xi_{yy} & = & 0 \text{ in } \Omega, \\ \text{Boundary conditions} & \text{on} & \partial\Omega \end{cases} \tag{4.16}$$

and

$$\begin{cases} \eta_{xx} + \eta_{yy} & = & 0 \text{ in } \Omega, \\ \text{Boundary conditions} & \text{on} & \partial\Omega \end{cases} \tag{4.17}$$

which are then inverted in order to find $x(\xi, \eta)$ and $y(\xi, \eta)$, thus we obtain as the generation system :

$$\begin{cases} g_{11} x_{\xi\xi} + g_{22} x_{\eta\eta} + 2 g_{12} x_{\xi\eta} & = & 0. \\ g_{11} y_{\xi\xi} + g_{22} y_{\eta\eta} + 2 g_{12} y_{\xi\eta} & = & 0. \end{cases} \tag{4.18}$$

with

$$g_{ij} = \sum_{m=1}^{2} A_{mi} A_{mj}$$

where $A_{mi} = (-1)^{i+m}(\text{Cofactor}_{m,i} \text{ of } [\mathcal{M}])$ and $[\mathcal{M}]$ is the above matrix.

This results in a system expressed in the logical space (where a mesh exists). It is a non-linear coupled system which can be solved using relaxation techniques or, more generally, iterative methods after an initialization by a solution in which the real boundary conditions are prescribed.

Variants. Variants of the previous generation systems can be experimented with to obtain special properties. For example, adding a non-zero right-hand side and considering :

$$\begin{cases} \xi_{xx} + \xi_{yy} & = & P \\ \text{Boundary conditions} \end{cases} \tag{4.19}$$

along with

$$\begin{cases} \eta_{xx} + \eta_{yy} & = & Q \\ \text{Boundary conditions} \end{cases} \tag{4.20}$$

enables us to control the distribution of points inside the domain. In this situation, the inverse system is :

$$\begin{cases} g_{11} x_{\xi\xi} + g_{22} x_{\eta\eta} + 2 g_{12} x_{\xi\eta} + J^2(P x_\xi + Q x_\eta) & = & 0, \\ g_{11} y_{\xi\xi} + g_{22} y_{\eta\eta} + 2 g_{12} y_{\xi\eta} + J^2(P y_\xi + Q y_\eta) & = & 0. \end{cases} \tag{4.21}$$

using the same notation and J, the Jacobian, being defined by $J = det([\mathcal{M}])$.

The right-hand sides P and Q interact as follows :

For $P > 0$, the points are attracted to the "right", $P < 0$ induces the inverse effect.

For $Q > 0$ the points are attracted to the "top", $Q < 0$ leads to the inverse effect.

Close to the boundary, P and Q induces an inclination of lines $\xi = constant$ (or $\eta = constant$).

P (Q) can also be used to concentrate lines $\xi = constant$ or $\eta = constant$ towards a given line, or to attract them towards a given point. To achieve this, the right-hand side can be defined as follows :

$$P(\xi,\eta) = -\sum_{i=1}^{n} a_i sign(\xi-\xi_i) e^{-c_i|\xi-\xi_i|} - \sum_{i=1}^{m} b_i sign(\xi-\xi_i) e^{-d_i\left[(\xi-\xi_i)^2+(\eta-\eta_i)^2\right]^{\frac{1}{2}}}$$

where n and m denote the number of lines in ξ and η of the grid. Such a control function induces the following :

for $a_i > 0$, $i = 1, n$, lines ξ are attracted to line ξ_i,

for $b_i > 0$, $i = 1, m$, lines ξ are attracted to point (ξ_i, η_i).

These effects are modulated by the amplitude of a_i (b_i) and by the distance from the attraction line (attraction point), modulated by coefficients c_i and d_i. For $a_i < 0$ ($b_i < 0$), the attraction is transformed into a repulsion. When $a_i = 0$ ($b_i = 0$), no particular action is connected to line ξ_i (or point (ξ_i, η_i)).

A right-hand side Q of the same form produces analogous effects with respect to η by interchanging the roles of ξ and η.

The major difficulty for automating this class of mesh generation systems is how to choose the control functions (P, Q, etc.) and the parameters they involve. However, these methods can be extended to three dimensions and, for a complete discussion, the reader is referred to [Thompson-1982a], where other forms of right-hand sides P and Q producing other properties are discussed (for example, the concentration of lines ξ or η towards an arbitrary line and not only towards a particular one ($\xi = constant$ or $\eta = constant$) or towards a given point to increase the mesh density near this point). In the above mentioned reference, and some others (for instance, [Knupp,Steinberg-1993]), other types of generation systems, including parabolic and hyperbolic operators, are discussed and numerous examples are provided.

How to define the quadrilateral (cuboid) analogy. When using a generation method of the present class, it is convenient to find the best analogy from the domain to a logical shape (quadrilateral or cuboïd). Such an analogy is often obtained either by partitioning the domain into several simpler domains, or by identifying the domain with the required shape, using several methods.

For example, in two dimensions, there are several major classes of decompositions of the domain under consideration from which different kinds of grids will result in order to capture the physics of the problem as well as possible. In this respect, a domain can be discretized following an O-type, C-type or H-type analysis (Figure 4.11). To obtain such an analogy, artificial cuts must be introduced. Such analogies extend to a greater or lesser degree to three dimensions.

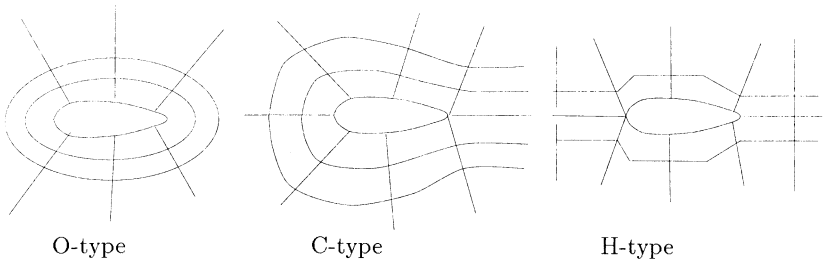

Figure 4.11: *O-type, C-type and H-type decompositions.*

4.2.2 Surface meshing and complex shapes

P.D.E. based methods can also be employed to generate surface meshes or domains having more complex geometries as local mesh generation processes (as will be discussed below in the multiblock method).

4.3 Multiblock method

As algebraic methods and P.D.E. methods are unlikely to be suitable for complex geometries, other methods must be used to deal with such geometries. Multiblock type methods are an initial answer to this problem. The underlying idea of such methods is to take advantage of a local meshing process such as an algebraic or a P.D.E. meshing algorithm and to over pass its limitations by applying it in its successful range of applications.

4.3.1 Basic ideas

Provided with a local meshing process (of the algebraic or P.D.E. type) the aim is to split the geometry in terms of regions where the local meshing process applies. Thus, in two dimensions, a domain is decomposed in terms of convex (or not too deformed) triangles or quadrilaterals (when an algebraic method is used) or in terms of quadrilaterals only (when a P.D.E. method is involved). In three dimensions, the partitions that can be handled successfully are made of tetrahedral, pentahedral or hexahedral regions (when algebraic methods are considered) or hexahedral regions only (when P.D.E. methods are used).

Thus the key-point is to obtain such a suitable partition. Two kinds of partitions may be considered. The first type considers a partition to be *conformal* in itself while the second does not require such a property (Figure 4.12). See Chapter 3 where three categories of multiblock methods are introduced. In what follows, we consider a composite type method which is more demanding, in terms of continuity, at the block interfaces. Note that patch or overlapping type methods are less demanding in this respect but, for some aspects, can be based on what is described below.

Conformal partitions lead to a very simple method as the union of the meshes of two distinct members of the partition automatically results in a conformal mesh.

Otherwise some care must be taken to partition the domain (unless if a non conformal mesh is desired, as it is for some solution methods). For instance, in the example in Figure 4.12 (right), one must enforce the continuity between the lower line of the upper region with the upper lines of the two lower blocks.

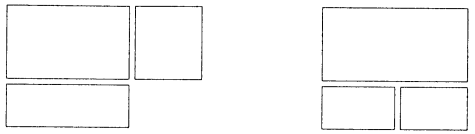

Figure 4.12: *Conformal and non-conformal decomposition of a two-dimensional domain. Three quad regions are defined which form a conformal partition (left-hand side) and a non-conformal partition (right-hand side).*

Obtaining a partition is a tedious task and generally it is the user who must undertake the task, which means that automating such a process is not so easy (see Chapter 9).

Some constraints must be considered to construct the partition of the whole domain, especially when a conformal partition is expected. A member of the partition is called a block[4] or a super-element, and some consistency is needed from block to block (or super-element to super-element).

Thus, the problem is one of finding an adequate partition into blocks or super-elements such that the interfaces between the blocks are consistently defined.

4.3.2 Partitioning the domain

This task is done by the user and, in this sense a multiblock method can be regarded as a *semi-automatic method*. The aim of this task is to define the different blocks necessary to define the domain in such a way that each block is *a priori* suitable for the local meshing process which will be applied to it. At the same time, some consistency must be ensured from block to block. Moreover, while following these constraints, an accurate approximation of the geometry must be obtained together with a control of the nature of the expected mesh (in terms of the number of elements, element sizes, etc.). To this end, both the way in which the partition is defined and the choice of the different subdivision parameters must be properly carried out.

To illustrate this multiple aspect, we consider a problem in two dimensions and we consider that two kinds of local meshing processes are available, one capable of carrying out triangular regions, the other suitable for quad regions (see Figure 4.13 where one can see how the subdivision parameters of the various blocks are related to one another). There remain two subdivision parameters, one for the triangle analogy and another one for the quad regions. The problem is now to choose these parameters appropriately in order to define a suitable partition.

[4] and as several blocks are created, the method is called a multiblock method.

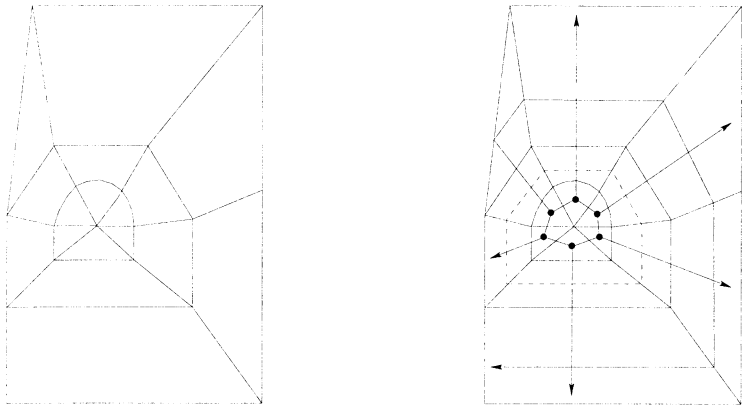

Figure 4.13: *Definition of the blocks defining the partition (left-hand side). Relationships between the subdivision parameters so as to enforce the continuity (right-hand side).*

Geometric aspect. This step is motivated by two goals. First, the geometry of the domain must be well approached by the blocks and, on the other hand, the shape of the blocks must be as close as possible to a convex region (to insure, in addition, a successful application of the local meshing process). With regard to these two aspects, a certain number of blocks must be constructed. A block is described by its corners, its edges (and its faces, in three dimensions). While several blocks may be useful, care must be taken of the interfaces from block to block. In two dimensions, this leads to defining the block edges in such a way as to insure a nice continuity. In three dimensions, faces from block to block must be carefully defined.

A pertinent choice of the number of blocks, together with that of the possible subdivision parameters, enables us to obtain a good approximation of the geometry. In the regions with high curvature, several blocks or a fine discretization (*i.e.*, a large enough subdivision parameter) is one solution to suit the geometry.

On completion to this step, several blocks are available whose corners, edges and faces are known.

Conformal and consistency requirements. The previous coarse partition can, in some cases, be refined to insure both the conformal aspect of the partition and the consistency between the items which are logically connected. Indeed, a subdivision parameter is associated with the block edges but some edges are connected (in the same block and from block to block), thus the number of possible subdivision parameters can be reduced. See again Figure 4.13 where we have only two possible different subdivision parameters.

4.3.3 Computational issues

Given the previous analysis, the blocks are now well defined. This means that we have defined the necessary corners, edges and faces which are the constitutive entities of the blocks. Now, we have to consider all these entities in a global way. Thus, a possible synthetic scheme of a multiblock method is as follows :

Step 1 : Corner definition. This stage involves a global numbering of the corners of the different blocks so as not to have a local numbering (when one considers only a given block) but a global corner numbering that could be used subsequently.

Step 2 : Edge definition and meshing. The edges are defined by their two endpoints (which are corners as previously introduced) and a subdivision parameter (n). Then, according to the meshing capabilities, each edge is split into $n+1$ segments (meaning that n intermediate points are created along it). In terms of point locations, we face a curve meshing problem (see Chapter 14). In terms of point numbering, we have, as above, to define a global numbering of the thus created vertices.

Step 3 : Face definition and meshing. The faces are defined by their edges. Following the type of the face (triangle or quad) and according to the different subdivision parameters, the face is meshed using for instance an algebraic method or a similar surface mesh generation method which completes a structured mesh. The resulting mesh is then of a nature that will enable us to continue the process (*i.e.*, the surface meshes are of the structured type). As above, a global numbering of the created vertices must be done.

Step 4 : Block definition and meshing. The blocks are defined by their faces and then, the local meshing process is applied. The numbering of the internal vertices can be then made sequentially starting from the first available index, *i.e.*, the last number of the last face vertices plus 1.

Step 5 : Global mesh construction. Actually, this step is automatically completed since a global numbering of the vertices has been developed in the previous steps.

Thus the idea is to process all the corners, all the edges and all the faces before processing the mesh of the different blocks. Relationships between corners, edges, faces and parent blocks are to be properly defined. In peculiar, a given edge (face) shared by several blocks must be defined in a *global* and *unique* way but, from a block point of view, can appear in various ways. Thus, some flags are needed to insure consistency between the global definition (which is unique) and the different local definitions (which can vary based on the block considered).

Global definition of the corners. The corners are introduced to match the previous requirements (geometry, consistency, etc.). An index (for instance, starting from index 1 for the first corner of the first block) is associated with each corner.

Global definition of the edges and faces. Provided the corners (let np_c be the number of corners), the edges must be defined in a global way. One solution is, when considering an edge, say ab where a and b are the two endpoint indices, to define the edge as ab if $a < b$ and as ba otherwise.

Then, when visiting an element like abc, we meet edge ab whereas when looking at element dba, we meet edge ba. In terms of edge definition, edge ab as well as edge ba must be uniquely defined. Note that the previous convention allows this to be done.

The global definition of the faces follows a similar rule. Let us consider a quad face whose endpoints are a, b, c and d. These indices are sorted from the smallest to the largest and this new series of indices is the global definition of the face. In fact, let a be the smallest index among the four face indices, we consider the index of point b (which is next to a) and that of point d (which precedes a), then

- if $d < b$, then use ad as a basis for global numbering,
- otherwise, use ab (see below).

Definition of blocks in terms of corners, edges and faces. The point is to identify the corners, the edges and the faces of the various blocks so as to return to the global definition of these entities.

Let us consider a hexahedral block involving three subdivision parameters, n_i ($i = 1, 3$). The block definition involves 8 corners, 12 edges and 6 quad faces. The issue is to find the proper correspondence between these entities and the block and to know the indices of the points that are involved (the corners, the edge vertices and the face vertices) as well as the indices of the vertices which will be created inside the block. A simple way to access all these vertex indices is to associate a *numbering matrix* with the block. To this end, we introduce a matrix with three indices which conforms to the following :

$$\mathcal{M}(0 : n_1 + 1, 0 : n_2 + 1, 0 : n_3 + 1)\,,$$

where, for instance, $\mathcal{M}(i, j, 0)$, for $i = 0, n_1$ and $j = 0, n_2$ corresponds to the "bottom" face of the block. More precisely, $\mathcal{M}(0, 0, 0)$, $\mathcal{M}(n_1, 0, 0)$, $\mathcal{M}(0, n_2, 0)$ and $\mathcal{M}(n_1, n_2, 0)$ are the four corners of the face and $\mathcal{M}(i, 0, 0)$, $\mathcal{M}(n_1, j, 0)$, $\mathcal{M}(i, n_2, 0)$ and $\mathcal{M}(0, j, 0)$ where i and j vary corresponding to the four edges of the face. Thus the proper definition of the block involves filling its numbering matrix for the entities already known while the part of the matrix not yet known will be defined when the block is meshed.

Global numbering of the edge vertices. The given edges are first dealt with. Let $free = np_c + 1$. Then, for the first edge we apply an algorithm as below. Next, the $free$ value being completed, we turn to the next edge until all the edges have been visited :

Algorithm 4.1 *Global numbering of the edge vertices.*

Procedure GlobalNumbering
DO FOR $i = 1, n$ (n being the number of desired points
along the current edge, after the subdivision parameter
$free = free + 1$,
$v_i = free$, *i.e.,* vertex i of the edge is labeled
with index $free$,
END DO FOR $i = 1, n$

Then, for edge ab, if $a < b$, the vertex indices are :

$$a, free, free + 1, ..., free + n, b\,,$$

while if $a > b$, these indices are :

$$a, free + n, free + n - 1, ..., free, b\,.$$

The adequate sequence of indices is put on the various numbering matrices which correspond to the various blocks sharing this edge. Depending on the block, *i.e.,* depending on the location of a and b in the block under examination, the sequence $free, free+1, ..., free+n$ or $free+n, free+n-1, ..., free$ is merged in the matrix at the relevant place. Actually, the matrix indices of interest are those of the line (of indices) "joining a and b".

Global numbering of the face vertices. Then we consider the face vertices. First, the vertices located along the face edges are already labeled (see above). It remains to find a global label for the vertices interior to the face prior to filling the corresponding numbering matrices. The idea is to define (for the sake of simplicity, we consider a quad analogy) two directions of numbering. For instance, if $free$ is the first available label, the first line of the matrix could be :

$$free, free + 1, free + 2, ..., free + n$$

and, the second line could be :

$$free + n + 1, free + n + 2, ..., free + n + n\,.$$

Thus, according to the global definition of a face, several systems of numbering can be found. Let a_i be the four corners of the face. We pick the smallest index, say a_1, and we examine the indices of the corners before and after a_1 (see above), then

- if $a_2 < a_4$, the base of numbering is a_1a_2 meaning that a (sequential) variation from a_1 to a_2 is used,
- if $a_4 < a_2$, the base of numbering is a_1a_4 meaning that a (sequential) variation from a_1 to a_4 is used,

a_4	-	-	-	a_3
-	7	8	9	-
-	4	5	6	-
-	1	2	3	-
a_1	-	-	-	a_2

a_4	-	-	-	a_3
-	3	6	9	-
-	2	5	8	-
-	1	4	7	-
a_1	-	-	-	a_2

Table 4.2: The two "global" indice systems when a_1 is the smallest index among the four a_i's and, left, when $a_2 < a_4$ or, right, $a_4 < a_2$ (in this table, for the sake of simplicity, we assume $free = 1$ (which is obviously not possible, as in practice a shift must be made) and $n = 2$ for both pairs of edges).

thus resulting, a_1 being identified, in two possible situations. Then, depending on the case, eight possible numbering systems can be found.

Now, the global indices of the face vertices are stored onto the numbering matrices at the proper places depending on what the a_i's are.

Global numbering of the internal vertices. Once again, let $free$ be the last index used when labeling the face vertices, then the internal vertices are sequentially numbered. Three directions are used (in terms of index variation), an i-direction, a j-direction and a k-direction. Given the corners of the block, the i-direction follows edge a_1, a_2, the j-direction follows edge a_1, a_4 and the third follows edge a_1, a_8 (for a hex block).

Element vertex enumeration. At this stage, a global numbering system is available for all the vertices. In fact, a vertex is also known through its logical position in a given block. This is done using the matrix $\mathcal{M}$ associated with the block (we assume the same hex example as above). Then, the enumeration of the vertices of the different elements of the resulting mesh is easy to obtain. For instance, given i, j and k, the vertices of the corresponding (final) element are the values contained in :

$$\mathcal{M}(i,j,k), \mathcal{M}(i+1,j,k), \mathcal{M}(i+1,j+1,k), \mathcal{M}(i,j+1,k) \quad \text{and}$$

$$\mathcal{M}(i,j,k+1), \mathcal{M}(i+1,j,k+1), \mathcal{M}(i+1,j+1,k+1), \mathcal{M}(i,j+1,k+1)$$

for the bottom (resp. top) face of the element.

Note that while this enumeration is trivial for a hex or a pentahedral analogy, it requires some care for a tet analogy. This is due to the fact that, on the one hand, the final mesh is not formed of similar layers of elements and, on the other hand, that a region bounded by two triangles leads to the construction of one, two or three tet (in contrast to the other cases where two faces on two consecutive layers define only one element).

Thus the tet case is more tedious. Actually, given a face (say the bottom face of the block), we can classify the triangles covering this face into four categories.

- those resulting in three tets when considering a layer (say, in terms of index k, when going from k to $k + 1$). Here, we must fill a small pentahedron,

- those also resulting in three tets (although they are inverted as compared with the previous ones),
- those resulting in two tets (those sharing a point with the "blended" block face), where the region to be meshed reduces to a prism (a pentahedron from which a tet is removed),
- those resulting in one tet (those sharing an edge with the "blended" block face or those located at a corner at level $n-1$). In this case, only a small tet must be meshed.

Exercise 4.2 *Find the four patterns encountered when dealing with a tet. Find the vertex indices (prior to applying the numbering matrix) of the final mesh based on these cases.*

Limitations. In principle, a multiblock method has the same range of applications than the local meshing processes that are used. Nevertheless, as the partition of the domain is made so as to over pass these limits, a multiblock method can successfully handle any arbitrarily shaped domain. In fact, the user is responsible for the success of the method by constructing a partition in suitable blocks.

Remark 4.5 *Applied in the surface case, say the blocks consist of triangles and quads in $\mathbb{R}^3$, the multiblock method provides a way to mesh a surface. Note that, in this case, the geometry of the surface is only related to the edges of the partition (see also Chapters 13 and 15).*

4.3.4 Application examples

The example in Figures 4.14 and 4.15 is a two-dimensional application of a multiblock method. The coarse partition consists of 10 quad regions and 8 triangular regions. There are 18 corners and 33 edges. Actually, only one subdivision parameter is used (for consistency reasons), leading to rather different element density. One should note that different block partitions can be used in this case resulting in different global meshes.

Figure 4.15 depicts a simple three dimensional application example. The figure on the left shows the part of the domain effectively considered. It consists of seven blocks. The figure on the right displays the mesh of the whole domain obtained from the previous mesh after several symmetries and (sub)mesh merging operations (see Chapter 17 about such mesh manipulation operators).

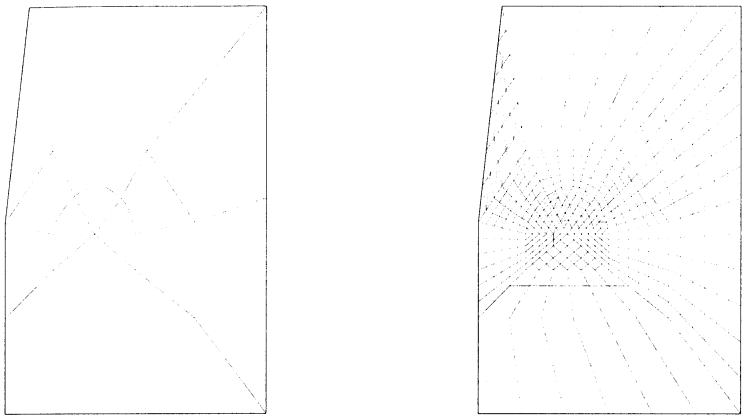

Figure 4.14: *Input data for the multiblock method, left-hand side, resulting mesh, right-hand side (two dimensional example).*

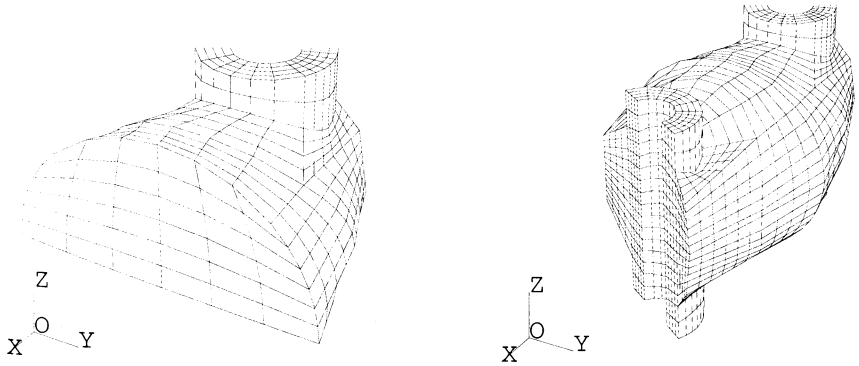

Figure 4.15: *Detail of the part effectively dealt with and resulting mesh after post-processing (three dimensional example).*