Part III, Chapter 8

Meshes

In Part III, composed of Chapters 8 to 17, we introduce the notion of meshes, show how to generate a finite element on each cell composing the mesh, and estimate the interpolation error in each mesh cell. We also derive important discrete inverse and functional inequalities in each mesh cell. Moreover, we discuss in some detail finite elements in $\boldsymbol{H}(\mathrm{div})$ and $\boldsymbol{H}(\mathrm{curl})$. In the present chapter, we study how to build a mesh of a bounded subset $D \subsetneq \mathbb{R}^d$, i.e., a finite collection of cells forming a partition of D. This is indeed the first important task to realize when one wants to approximate some PDEs posed in D. The viewpoint we adopt in this book is that each mesh cell is the image of a reference cell by some smooth diffeomorphism that we call geometric mapping. We show how to construct the geometric mapping and we present various important notions concerning meshes. We also discuss mesh-related data structures and mesh generators.

8.1 The geometric mapping

Let $\widehat{K}$ be a polyhedron in $\mathbb{R}^d$, called *reference cell*. We want to build a smooth diffeomorphism (i.e., an invertible mapping) $\boldsymbol{T}_K$ from $\widehat{K}$ to $K := \boldsymbol{T}_K(\widehat{K})$ using a set of *geometric nodes* $\{\boldsymbol{g}_i\}_{i\in\mathcal{N}_{\mathrm{geo}}}$ in K with $\mathcal{N}_{\mathrm{geo}} := \{1{:}n_{\mathrm{geo}}\}$ for some integer n_{geo}. In practice, these nodes are provided by a mesh generator. The key idea to build $\boldsymbol{T}_K$ is to use a Lagrange finite element in $\widehat{K}$, say $(\widehat{K}, \widehat{P}_{\mathrm{geo}}, \widehat{\Sigma}_{\mathrm{geo}})$, with reference Lagrange nodes $\{\widehat{\boldsymbol{g}}_i\}_{i\in\mathcal{N}_{\mathrm{geo}}}$ in $\widehat{K}$. This finite element is called *geometric finite element*. It is standard to assume that $\widehat{P}_{\mathrm{geo}}$ is a space of d-variate polynomials and that there is an integer $k_{\mathrm{geo}} \geq 1$ s.t.

$$\mathbb{P}_{k_{\mathrm{geo}},d} \subset \widehat{P}_{\mathrm{geo}} \subset C^\infty(\widehat{K}). \tag{8.1}$$

Notice that $n_{\mathrm{geo}} \geq d+1$ since $k_{\mathrm{geo}} \geq 1$. Let $\{\widehat{\psi}_i\}_{i\in\mathcal{N}_{\mathrm{geo}}}$ be the shape functions of the geometric finite element.

Definition 8.1 (Geometric mapping). *The* geometric mapping $\boldsymbol{T}_K : \widehat{K} \to K$ *is defined by*

$$\boldsymbol{T}_K(\widehat{\boldsymbol{x}}) := \sum_{i \in \mathcal{N}_{\mathrm{geo}}} \widehat{\psi}_i(\widehat{\boldsymbol{x}}) \boldsymbol{g}_i, \qquad \forall \widehat{\boldsymbol{x}} \in \widehat{K}. \tag{8.2}$$

Since $\widehat{\psi}_i(\widehat{\boldsymbol{g}}_j) = \delta_{ij}$ for all $i, j \in \mathcal{N}_{\mathrm{geo}}$, we have $\boldsymbol{T}_K(\widehat{\boldsymbol{g}}_j) = \boldsymbol{g}_j$. Notice that this construction implies that $\boldsymbol{T}_K$ is of class C^∞. We henceforth assume that $\boldsymbol{T}_K$ is a C^∞ diffeomorphism. Some care has to be taken when choosing the geometric nodes $\{\boldsymbol{g}_i\}_{i \in \mathcal{N}_{\mathrm{geo}}}$ to ensure that $\boldsymbol{T}_K$ is indeed bijective when $\boldsymbol{T}_K$ is not affine. Some counterexamples are shown in Figures 8.1 and 8.2.

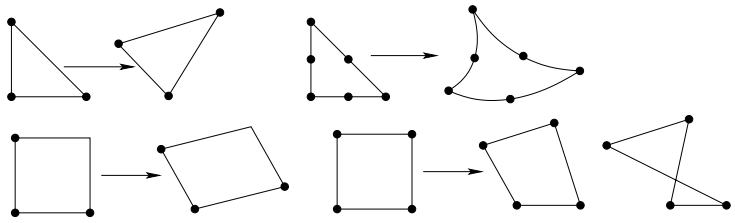


Fig. 8.1 $\mathbb{P}_1$-based generation of a triangle (top left), $\mathbb{P}_2$-based generation of a curved triangle (top right), $\mathbb{P}_1$-based generation of a parallelogram (bottom left), $\mathbb{Q}_1$-based generation of two quadrangles, the second one with a nonbijective mapping (bottom right).

We adopt the usual convention that consists of identifying vectors in $\mathbb{R}^d$ with column vectors. This allows us to identify $\boldsymbol{T}_K$ with the column vector with entries $(\boldsymbol{T}_K)_i$ for all $i \in \{1{:}d\}$ and the Jacobian of $\boldsymbol{T}_K$ with the matrix with entries

$$(\mathbb{J}_K)_{ij} := \partial_j(\boldsymbol{T}_K)_i, \qquad \forall i, j \in \{1{:}d\}, \tag{8.3}$$

where i is the row index and j the column index. The field $\mathbb{J}_K$ is $\mathbb{R}^{d \times d}$-valued and it is constant over $\widehat{K}$ if $\boldsymbol{T}_K$ is affine. Notice that the sign of $\det(\mathbb{J}_K)$ is necessarily constant over $\widehat{K}$ since we assumed that $\det(\mathbb{J}_K)(\widehat{\boldsymbol{x}}) \neq 0$ for all $\widehat{\boldsymbol{x}} \in \widehat{K}$ (this is indeed a necessary condition for $\boldsymbol{T}_K$ to be bijective). Contrary to what is done sometimes in the literature, we do not require that $\det(\mathbb{J}_K)$ has any particular sign.

Example 8.2 (Simplex generation). Let $\widehat{K}$ be the unit simplex in $\mathbb{R}^d$ with barycentric coordinates $\{\widehat{\lambda}_i\}_{i \in \{0:d\}}$ ($\widehat{\lambda}_0(\widehat{\boldsymbol{x}}) := 1 - \sum_{i \in \{1:d\}} \widehat{x}_i$ and $\widehat{\lambda}_i(\widehat{\boldsymbol{x}}) := \widehat{x}_i$ for all $i \in \{1{:}d\}$). Let K be a simplex in $\mathbb{R}^d$. Taking $\widehat{P}_{\mathrm{geo}} := \mathbb{P}_{1,d}$ and the $n_{\mathrm{geo}} := (d+1)$ vertices of K as geometric nodes, the geometric mapping $\boldsymbol{T}_K : \widehat{K} \to K$ is s.t. $\boldsymbol{T}_K(\widehat{\boldsymbol{x}}) := \sum_{i \in \{0:d\}} \widehat{\lambda}_i(\widehat{\boldsymbol{x}}) \boldsymbol{z}_i$ for all $\widehat{\boldsymbol{x}} \in \widehat{K}$. In dimension two, taking $\widehat{P}_{\mathrm{geo}} := \mathbb{P}_{2,2}$, i.e., $n_{\mathrm{geo}} := 6$, we can prescribe six geometric nodes in K and build a triangle with curved faces. See Figure 8.1 (top row) for illustrations. When using high-order elements, some care must

be taken to ensure that the geometric mapping $\boldsymbol{T}_K$ is indeed invertible. Figure 8.2 presents two examples where the mapping $\boldsymbol{T}_K$ is not invertible. For the one shown on the left, the enumerations chosen for the geometric nodes of $\widehat{K}$ and K_1 are not compatible. The example shown on the right is slightly more subtle since the singularity comes from the fact that the shape functions of the $\mathbb{P}_{2,2}$ Lagrange finite element can take negative values and that some geometric nodes of K_2 are too close. □

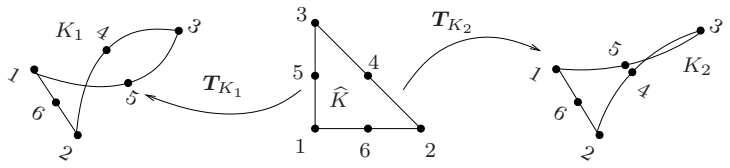


Fig. 8.2 Left: incompatible enumeration of the geometric nodes. Right: compatible enumeration, but some geometric nodes are too close.

Example 8.3 (Quadrangle generation). Let $\widehat{K} := (0,1)^2$ be the unit square in $\mathbb{R}^2$. Let us set $\widehat{\boldsymbol{z}}_0 := (0,0)$, $\widehat{\boldsymbol{z}}_1 := (1,0)$, $\widehat{\boldsymbol{z}}_2 := (0,1)$, and $\widehat{\boldsymbol{z}}_3 := (1,1)$. Taking $\widehat{P}_{\mathrm{geo}} := \mathbb{P}_{1,2}$, so that $n_{\mathrm{geo}} = 3$, we can prescribe three geometric nodes in K to build a smooth diffeomorphism. Let $\boldsymbol{z}_0$ be one vertex of K and let $\boldsymbol{z}_1, \boldsymbol{z}_2$ be the other two vertices of K sharing an edge with $\boldsymbol{z}_0$. Let $\boldsymbol{z}_3$ be the fourth vertex of K. Upon setting $\boldsymbol{T}_K(\widehat{\boldsymbol{x}}) := (1-\widehat{x}_1-\widehat{x}_2)\boldsymbol{z}_0+\widehat{x}_1\boldsymbol{z}_1+\widehat{x}_2\boldsymbol{z}_2$, we observe that K is a parallelogram. In particular, $\boldsymbol{z}_3 = \boldsymbol{T}_K(\widehat{\boldsymbol{z}}_3) = -\boldsymbol{z}_0+\boldsymbol{z}_1+\boldsymbol{z}_2$, i.e., $\boldsymbol{z}_0+\boldsymbol{z}_3 = \boldsymbol{z}_1+\boldsymbol{z}_2$. To generate a more general quadrangle, we can take $\widehat{P}_{\mathrm{geo}} := \mathbb{Q}_{1,2}$, so that $n_{\mathrm{geo}} = 4$, and use the four vertices of K as geometric nodes. In this case, $\boldsymbol{T}_K(\widehat{\boldsymbol{x}}) = (1-\widehat{x}_1)(1-\widehat{x}_2)\boldsymbol{z}_0+\widehat{x}_1(1-\widehat{x}_2)\boldsymbol{z}_1+(1-\widehat{x}_1)\widehat{x}_2\boldsymbol{z}_2+\widehat{x}_1\widehat{x}_2\boldsymbol{z}_3$. The mapping $\boldsymbol{T}_K$ is a smooth diffeomorphism whenever the nodes of K are properly enumerated. See the bottom row of Figure 8.1 for illustrations. In the rightmost example, $\boldsymbol{T}_K$ is not invertible because the nodes are not properly enumerated. □

8.2 Main definitions related to meshes

Definition 8.4 (Mesh). *Let D be a Lipschitz domain in $\mathbb{R}^d$. We say that $\mathcal{T}_h$ is a* mesh *of D if $\mathcal{T}_h$ is a finite collection of closed subsets of D, called* mesh cells *(or mesh elements), such that* (i) *the interiors of the mesh cells are all nonempty Lipschitz domains in $\mathbb{R}^d$ that are mutually disjoint and* (ii) *all the mesh cells cover $\overline{D}$ exactly, i.e.,*

$$\overline{D} = \bigcup_{K\in\mathcal{T}_h} K. \tag{8.4}$$

The subscript h refers to a level of refinement. It is common in the literature to set $h := \max_{K\in\mathcal{T}_h} h_K$ with $h_K := \mathrm{diam}(K) := \max_{\boldsymbol{x}_1,\boldsymbol{x}_2\in K} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_{\ell^2}$, where $\|\cdot\|_{\ell^2}$ is the Euclidean norm in $\mathbb{R}^d$, and to call h the meshsize.

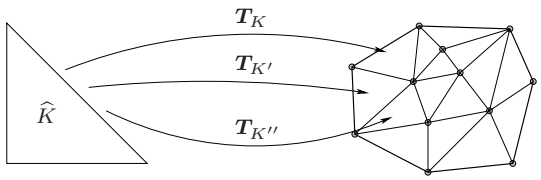


Fig. 8.3 Reference cell $\widehat{K}$ (left), mesh (right). The three arrows indicate the action of the geometric mapping for the three mesh cells K, K', K''.

The mesh cells have often a simple shape. For simplicity, we assume in this book that all the mesh cells have been generated from a fixed reference polyhedron $\widehat{K} \in \mathbb{R}^d$ (see §8.1) so that there is a smooth diffeomorphism $\boldsymbol{T}_K : \widehat{K} \to K$ for all $K \in \mathcal{T}_h$. Figure 8.3 presents an illustration using $\mathbb{P}_1$ geometric mappings to generate triangular cells. More generally, it is possible to consider a finite set of reference polyhedra to generate the mesh cells. One can for instance build meshes mixing triangles and quadrangles in dimension two, etc.

Remark 8.5 (Approximation of D). It happens sometimes that generating meshes that partition D exactly is too complicated, or that it is only possible to construct meshes of approximations of D. For instance, this situation arises when the boundary of D is curved; see §13.1 for examples. Unless specified otherwise, meshes are assumed to partition D exactly. □

Definition 8.6 (Simplicial/affine mesh). *The mesh $\mathcal{T}_h$ is said to be* simplicial *when the reference cell $\widehat{K}$ is a simplex, and the mesh $\mathcal{T}_h$ is said to be* affine *when all the geometric mappings $\{\boldsymbol{T}_K\}_{K\in\mathcal{T}_h}$ are affine.*

In this book, we often consider simplicial affine meshes, and we speak of triangulations when $d = 2$. An example is shown in Figure 8.4.

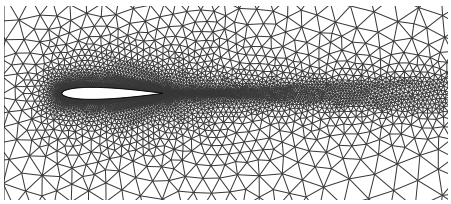


Fig. 8.4 Part of a triangulation around a two-dimensional NACA0012 airfoil profile.

Definition 8.7 (Faces, edges, and vertices of a cell). *Let $K \in \mathcal{T}_h$ be a cell. Assuming $d = 3$, the faces, edges, and vertices of K are defined to be the images by $\boldsymbol{T}_K$ of the faces, edges, and vertices of the reference polyhedron $\widehat{K}$, and these geometric entities are collected in the sets $\mathcal{F}_K$, $\mathcal{E}_K$, and $\mathcal{V}_K$, respectively. The same definition is valid in dimension $d = 2$ with the exception that the notions of edge and face coincide. The same definition is valid in dimension $d = 1$, with the exception that the notions of vertex, edge, and face coincide. We assume in the entire book that we have either $F \subset \partial D$ or $\mathrm{int}(F) \subset D$ for all $K \in \mathcal{T}_h$ and all $F \in \mathcal{F}_K$.*

Remark 8.8 (Geometric nodes). The notion of geometric nodes introduced in §8.1 and the notion of vertices are different. In general, the vertices of a cell form a subset of its geometric nodes. These two sets coincide if the geometric element is a $\mathbb{P}_{1,d}$ or $\mathbb{Q}_{1,d}$ Lagrange element. □

Definition 8.9 (Mesh faces, edges, and vertices). *Let $\mathcal{T}_h$ be a mesh. Assume $d = 3$. We say that a closed two-dimensional manifold $F \subset \overline{D}$ is a mesh face if there is a mesh cell $K \in \mathcal{T}_h$ s.t. F is a face of K, i.e., $F \in \mathcal{F}_K$. Similarly, a closed one-dimensional manifold $E \subset \overline{D}$ is a mesh edge if there is a mesh cell $K \in \mathcal{T}_h$ s.t. $E \in \mathcal{E}_K$, and a point $\boldsymbol{z} \in \overline{D}$ is a mesh vertex if there is a mesh cell $K \in \mathcal{T}_h$ s.t. $\boldsymbol{z} \in \mathcal{V}_K$.*

Another important notion is that of interfaces and boundary faces.

Definition 8.10 (Interfaces, boundary faces). *A subset $F \subset \overline{D}$ is an* interface *if F has positive $(d-1)$-dimensional measure and there are two distinct mesh cells $K_l, K_r \in \mathcal{T}_h$ such that $F := \partial K_l \cap \partial K_r$ and F is a subset of a face of K_l and of a face of K_r. A subset $F \subset \overline{D}$ is a* boundary face *if F has positive $(d-1)$-dimensional measure and if there is a mesh cell $K_l \in \mathcal{T}_h$ such that $F := \partial K_l \cap \partial D$ and F is a face of K_l. All the interfaces are collected in the set $\mathcal{F}_h^\circ$, all the boundary faces are collected in the set $\mathcal{F}_h^\partial$, and we define*

$$\mathcal{F}_h := \mathcal{F}_h^\circ \cup \mathcal{F}_h^\partial. \tag{8.5}$$

The subscripts $\{l, r\}$ in the definition $F := \partial K_l \cap \partial K_r$ refer to the left cell and to the right cell. The notion of left and right cell will be unambiguously defined later by orienting all the interfaces. Distinguishing the left from the right cell will be important when defining jumps across interfaces (see Definition 18.2). In addition, we also have $F = K_l \cap K_r$ since the mesh cells have mutually disjoint interiors by assumption. Furthermore, we observe that a boundary face is always a mesh face, but an interface is not necessarily a mesh face since the notion of interface depends on the way adjacent mesh cells come into contact. An illustration is presented in Figure 8.5. For the mesh shown in the left panel, we have $\mathcal{F}_h = \bigcup_{K\in\mathcal{T}_h} \mathcal{F}_K$. For that shown in the central panel, we have $\mathcal{F}_h \subset \bigcup_{K\in\mathcal{T}_h} \mathcal{F}_K$ but $\bigcup_{K\in\mathcal{T}_h} \mathcal{F}_K \not\subset \mathcal{F}_h$. For that shown in the right panel, we have $\mathcal{F}_h \not\subset \bigcup_{K\in\mathcal{T}_h} \mathcal{F}_K$ and $\bigcup_{K\in\mathcal{T}_h} \mathcal{F}_K \not\subset \mathcal{F}_h$.

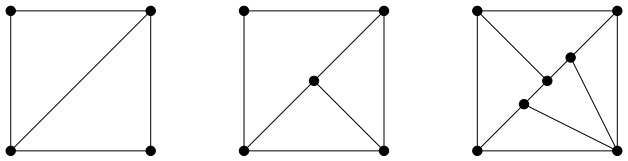


Fig. 8.5 Three examples of a triangulation of a square. Left panel: the mesh is composed of 2 cells and there is one interface. Central panel: the mesh is composed of 3 cells and there are 3 interfaces. Right panel: the mesh is composed of 5 cells and there are 7 interfaces. The three meshes contain 4 boundary faces.

The meshes shown in Figure 8.3, in Figure 8.4, and in the left panel of Figure 8.5 fall into the important class of matching meshes. Matching meshes play a central role in this book since they facilitate the construction of discrete spaces composed of piecewise smooth functions having an integrable gradient, curl or divergence (see Chapter 19 and onwards).

Definition 8.11 (Matching mesh). *A mesh $\mathcal{T}_h$ is said to be* matching *if for all cells $K, K' \in \mathcal{T}_h$ s.t. $K \cap K'$ is a manifold of dimension $(d-1)$, then $K \cap K'$ is an entire face of K and an entire face of K'.*

Proposition 8.12 (Mesh faces). *Let $\mathcal{T}_h$ be a matching mesh. Then,*

$$\mathcal{F}_h = \bigcup_{K \in \mathcal{T}_h} \mathcal{F}_K. \tag{8.6}$$

Proof. Let $F \in \mathcal{F}_h$. If $F \in \mathcal{F}_h^\partial$, we infer from Definition 8.10 that $F \in \mathcal{F}_{K_l}$, whence $F \in \bigcup_{K \in \mathcal{T}_h} \mathcal{F}_K$. If $F \in \mathcal{F}_h^\circ$, we have $F := \partial K_l \cap \partial K_r = K_l \cap K_r$, and we infer from Definition 8.11 that $F \in \mathcal{F}_{K_l} \cap \mathcal{F}_{K_r}$, whence $F \in \bigcup_{K \in \mathcal{T}_h} \mathcal{F}_K$. We have thus shown that $\mathcal{F}_h \subset \bigcup_{K \in \mathcal{T}_h} \mathcal{F}_K$. Conversely, let $K \in \mathcal{T}_h$ and $F \in \mathcal{F}_K$. If $F \subset \partial D$, we infer that $F \in \mathcal{F}_h^\partial$. Otherwise, our assumption on the faces of a mesh cell in Definition 8.7 implies that $\mathrm{int}(F) \subset D$, and since the mesh cells form a partition of D, we infer that there is a mesh cell $K' \neq K$ s.t. $K \cap K' \subset F$ and $K \cap K'$ is a manifold of dimension $(d-1)$. Since the mesh is matching, $K \cap K'$ is a full face of both K and K' so that $F = K \cap K'$, which proves that $F \in \mathcal{F}_h^\circ$. We have thus shown that $\bigcup_{K \in \mathcal{T}_h} \mathcal{F}_K \subset \mathcal{F}_h$, and this completes the proof. □

One can verify that Definition 8.11 implies that if $K \cap K' \neq \emptyset$ and $K \neq K'$, then the set $K \cap K'$ is a face, an edge (if $d = 3$), or a vertex that is common to K and K'. For matching meshes we denote the collection of the mesh edges (if $d = 3$) and the collection of the mesh vertices as follows:

$$\mathcal{E}_h := \bigcup_{K \in \mathcal{T}_h} \mathcal{E}_K, \qquad \mathcal{V}_h := \bigcup_{K \in \mathcal{T}_h} \mathcal{V}_K. \tag{8.7}$$

Remark 8.13 (Euler relations). Let $\mathcal{T}_h$ be a matching mesh of a polyhedron D in $\mathbb{R}^d$. If $d = 2$, let I be the degree of multiple-connectedness of

D (i.e., the number of holes in D). Let N_c, N_e, N_v, N_e^∂, N_v^∂ be the number of mesh cells, edges, vertices, boundary edges, and boundary vertices, respectively. Then we have

$$N_\mathrm{c} - N_\mathrm{e} + N_\mathrm{v} = 1 - I, \qquad N_\mathrm{v}^\partial - N_\mathrm{e}^\partial = 0. \tag{8.8}$$

If $d = 3$, let additionally J be the number of connected components of the boundary of D, and let N_f, N_f^∂ be the number of mesh faces and boundary faces, respectively. Then we have

$$N_\mathrm{c} - N_\mathrm{f} + N_\mathrm{e} - N_\mathrm{v} = -1 + I - J, \qquad N_\mathrm{f}^\partial - N_\mathrm{e}^\partial + N_\mathrm{v}^\partial = 2(J - I). \qquad \square$$

8.3 Data structure

A mesh is a data structure produced by a mesh generator. This data structure consists of a cloud of points, called *geometric nodes*, that are numbered and connected. There are many ways to construct this data structure. Let us give an example. We start by enumerating the geometric nodes $\{\boldsymbol{g}_1, \dots, \boldsymbol{g}_{N_\mathrm{geo}}\}$ where N_geo is the number of geometric nodes. This enumeration is said to be global. The geometric nodes are defined by their coordinates in $\mathbb{R}^d$. These quantities are stored in a two-dimensional array of size $d \times N_\mathrm{geo}$, which we denote by

$$\texttt{coord}(1{:}d, 1{:}N_\mathrm{geo}), \tag{8.9}$$

and we say that `coord` is the *coordinate array* of the mesh. For all $k \in \{1{:}d\}$ and all $n \in \{1{:}N_\mathrm{geo}\}$, $\texttt{coord}(k, n)$ is the k-th coordinate of $\boldsymbol{g}_n$.

The geometric nodes are organized into mesh cells by means of a *connectivity array*, in such a way that every mesh cell is assigned n_geo geometric nodes. Let us enumerate the mesh cells as $\{K_1, \dots, K_{N_\mathrm{c}}\}$ where N_c is the number of mesh cells. The geometric nodes associated with any mesh cell can be recovered from a two-dimensional array of size $N_\mathrm{c} \times n_\mathrm{geo}$, which we denote by

$$\texttt{j\_geo}(1{:}N_\mathrm{c}, 1{:}n_\mathrm{geo}). \tag{8.10}$$

For all $m \in \{1{:}N_\mathrm{c}\}$ and all $n \in \mathcal{N}_\mathrm{geo}$ (recall that $\mathcal{N}_\mathrm{geo} := \{1{:}n_\mathrm{geo}\}$), the integer $\texttt{j\_geo}(m, n)$ is the global index of the n-th node in the m-th cell. The second index in the array `j_geo` provides the *local enumeration* of the geometric nodes for each mesh cell. Using the connectivity array and the coordinate array, it is possible to rewrite the geometric mapping $\boldsymbol{T}_K$ from Definition 8.1 as follows:

$$(\boldsymbol{T}_{K_m}(\widehat{\boldsymbol{x}}))_i = \sum_{n \in \mathcal{N}_\mathrm{geo}} \widehat{\psi}_n(\widehat{\boldsymbol{x}})\, \texttt{coord}(i, \texttt{j\_geo}(m, n)), \tag{8.11}$$

for all $\widehat{\boldsymbol{x}} \in \widehat{K}$, all $m \in \{1{:}N_\mathrm{c}\}$, and all $i \in \{1{:}d\}$.

Example 8.14 (Enumeration in a simplex). Figure 8.6 shows an example of local and global enumerations. Here, the geometric reference element is the two-dimensional $\mathbb{P}_1$ Lagrange element, i.e., $n_{\mathrm{geo}} = 3$. We consider three mesh cells with global indices 56, 213, and 315. The values of the connectivity array are $\texttt{j\_geo}(315,1) = 13$, $\texttt{j\_geo}(315,2) = 37$, $\texttt{j\_geo}(315,3) = 250$, $\texttt{j\_geo}(56,1) = 13$, $\texttt{j\_geo}(56,2) = 37$, $\texttt{j\_geo}(56,3) = 53$, etc. We have adopted the convention that for any m, the value of $\texttt{j\_geo}(m,n)$ increases with n. This choice will be instrumental in Chapter 10 when orienting the mesh. Note that the sign of $\det(\mathbb{J}_K)$ is different in the cells 315 and 56. □

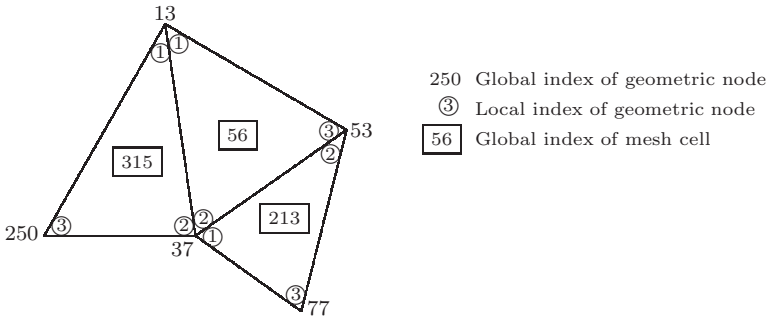


Fig. 8.6 Example of local and global enumerations of geometric nodes for three triangular mesh cells.

In many situations, it is useful to have two-dimensional arrays providing the global indices of the faces, edges, and vertices of any mesh cell. The reason is that finite element matrices are assembled by means of a loop over the mesh cells (see §29.2.3), and that these arrays are instrumental to identify degrees of freedom attached to the mesh faces, edges, and vertices. Let us focus on matching meshes and let us enumerate the mesh faces, edges, and vertices in $\mathcal{F}_h$, $\mathcal{E}_h$, and $\mathcal{V}_h$ from 1 to N_{f}, N_{e}, and N_{v}, respectively, i.e.,

$$\mathcal{F}_h = \{F_j\}_{j\in\{1:N_{\mathrm{f}}\}}, \qquad \mathcal{E}_h = \{E_j\}_{j\in\{1:N_{\mathrm{e}}\}}, \qquad \mathcal{V}_h = \{\boldsymbol{z}_j\}_{j\in\{1:N_{\mathrm{v}}\}}.$$

Let n_{cf}, n_{ce}, and n_{cv} be, respectively, the number of faces, edges, and vertices of a mesh cell. For instance, $n_{\mathrm{cf}} = 4$, $n_{\mathrm{ce}} = 6$, and $n_{\mathrm{cv}} = 4$ for a tetrahedron. We introduce the following two-dimensional arrays:

$$\texttt{j\_cf}(1{:}N_{\mathrm{c}}, 1{:}n_{\mathrm{cf}}), \qquad \texttt{j\_ce}(1{:}N_{\mathrm{c}}, 1{:}n_{\mathrm{ce}}), \qquad \texttt{j\_cv}(1{:}N_{\mathrm{c}}, 1{:}n_{\mathrm{cv}}). \tag{8.12}$$

For all $m \in \{1{:}N_{\mathrm{c}}\}$ and all $n \in \{1{:}n_{\mathrm{cf}}\}$, the integer $\texttt{j\_cf}(m,n)$ is the global index of the n-th face in the m-th cell, and similarly for $\texttt{j\_ce}$ and $\texttt{j\_cv}$. In other words, we have

$$\boldsymbol{T}_{K_m}(\widehat{F}_n) = F_{\texttt{j\_cf}(m,n)}, \quad \boldsymbol{T}_{K_m}(\widehat{E}_n) = E_{\texttt{j\_ce}(m,n)}, \quad \boldsymbol{T}_{K_m}(\widehat{\boldsymbol{z}}_n) = \boldsymbol{z}_{\texttt{j\_cv}(m,n)}.$$

Notice that the arrays `j_cv` and `j_geo` are different in general, just like the vertices and the geometric nodes may be different objects.

Remark 8.15 (Alternative data structure). Another choice is to consider the two-dimensional arrays `j_cf`$(1{:}N_{\rm c}, 1{:}n_{\rm cf})$ (as above) together with the two-dimensional arrays `j_fe`$(1{:}N_{\rm f}, 1{:}n_{\rm fe})$ (providing the global indices of the edges of a given mesh face, where $n_{\rm fe}$ is the number of edges of a face, assuming that this number is face-independent), and `j_ev`$(1{:}N_{\rm e}, 1{:}2)$ (providing the global indices of the two vertices of a mesh edge). The information stored in the array `j_ce` (resp., `j_cv`) can then be recovered from the arrays `j_cf` and `j_fe` (resp., `j_cf`, `j_fe`, and `j_ev`). The reader must be aware that all these compositions involve memory accesses that may be time consuming. □

8.4 Mesh generation

Mesh generation is a basic ingredient of finite element methods. Generating a mesh is often a time-consuming task, especially for complex three-dimensional configurations. Mesh generators involve two types of tasks: (1) representing geometrically the boundary of the domain by using suitable mappings parameterizing paths or surfaces; (2) meshing the lines, surfaces, and volumes that have been identified in the first task. This section briefly describes how to organize the above two tasks. The material is meant to provide some basic understanding of the process.

8.4.1 Two-dimensional case

Let us consider a two-dimensional domain D and let us think about how D can be geometrically represented.

1. D is entirely defined by its one-dimensional boundary, ∂D.
2. The boundary ∂D can be decomposed into its connected components.
3. Each connected component can be partitioned into a union of paths.
4. Each path can be assigned two extremities (possibly by cutting the paths that are closed). These points are referred to as the vertices of ∂D.
5. Each path can be mapped to the interval $[0, 1]$.

As an illustration, consider the domain shown in Figure 8.7. Its boundary is composed of two connected components. The external component is the union of the three paths PQ, QR, and RP. The internal boundary is transformed into a path that is homeomorphic to a segment by cutting it at S. In conclusion, the boundary of D is decomposed into the union of four paths: $\partial D_1 := PQ$, $\partial D_2 := QR$, $\partial D_3 := RP$, and $\partial D_4 := SS$.

A general algorithm for a two-dimensional mesh generator is obtained by reading in reverse order the above list:

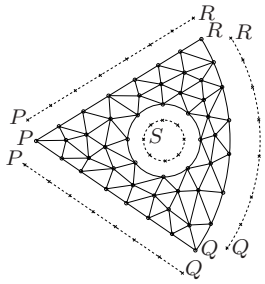


Fig. 8.7 Meshing a two-dimensional domain.

1. Locate the vertices of ∂D and partition $\partial D = \bigcup_{n\in\{1:\,N_{\mathrm{p}}^{\partial}\}} \partial D_n$ so that each elementary path ∂D_n is limited by two vertices (possibly identical). Here, $N_{\mathrm{p}}^{\partial}$ denotes the total number of elementary paths.
2. Connect the two vertices of ∂D_n for all $n \in \{1{:}\,N_{\mathrm{p}}^{\partial}\}$ by a parameterized path $\gamma_n : [0,1] \to \partial D_n$.
3. Letting $\bigcup_{i\in\{1:\,I_n\}}[x_{n,i-1}, x_{n,i}]$ be a partition of $[0,1]$ into I_n small segments, the *boundary mesh* on ∂D is $\bigcup_{n\in\{1:\,N_{\mathrm{p}}^{\partial}\}} \bigcup_{i\in\{1:\,I_n\}} \gamma_n([x_{n,i-1}, x_{n,i}])$.
4. Finally, mesh the interior of D by extending the boundary mesh. This last step usually involves an advancing front method where mesh vertices are progressively inserted inside the domain and connected to the other vertices to form new triangles (see Figure 8.8); see, e.g., Rebay [166] and the references therein.

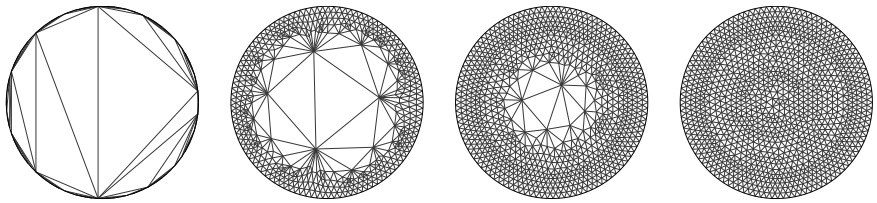


Fig. 8.8 Triangulation of a circle by an advancing front method. Various stages of the mesh generation process are illustrated.

8.4.2 Three-dimensional case

The above algorithm extends to dimension three. As in dimension two, the algorithm is deduced from the geometric description of three-dimensional domains. Let D be a three-dimensional domain.

1. D is entirely defined by its two-dimensional boundary, ∂D.
2. The boundary ∂D can be decomposed into its connected components.
3. Each connected component can be decomposed into a union of orientable surfaces with edges, say $\partial D = \bigcup_{n\in\{1:\,N_{\mathrm{s}}^{\partial}\}} \partial D_n$ ($N_{\mathrm{s}}^{\partial}$ is the total number

of these surfaces). For instance, a sphere can be decomposed into two hemispheres. The orientation of the connected components of ∂D says on which side of ∂D the interior of D is.

4. Each orientable surface ∂D_n can be mapped to a two-dimensional domain $\partial D_n^{\mathrm{2D}} \subset \mathbb{R}^2$ by a mapping $\gamma_n : \partial D_n^{\mathrm{2D}} \to \partial D_n$.
5. Each two-dimensional domain $\partial D_n^{\mathrm{2D}}$ for all $n \in \{1{:}N_{\mathrm{s}}^{\partial}\}$ can be described by means of the algorithm from §8.4.1.

An illustration is presented in Figure 8.9. The domain is a cone. Since the boundary of the cone is connected but has no edges, it is decomposed into two simpler surfaces by separating the base and the lateral surface. The base is homeomorphic to a disk, $\partial D_1^{\mathrm{2D}}$. The lateral surface is further transformed by cutting it along the segment PQ. The surface thus created is homeomorphic to a triangle, $\partial D_2^{\mathrm{2D}}$. When meshing the two sides of the triangle associated with the segment PQ, one must make sure that the two one-dimensional meshes coincide.

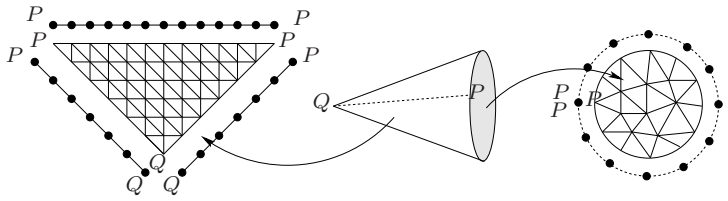


Fig. 8.9 Geometric representation of a three-dimensional domain.

An algorithm to mesh a three-dimensional domain is obtained by reading the above list from bottom to top:

1. Construct a mesh $\mathcal{T}_{h,n}^{\mathrm{2D}}$ of each two-dimensional domain $\partial D_n^{\mathrm{2D}}$ for all $n \in \{1{:}N_{\mathrm{s}}^{\partial}\}$ by applying the algorithm from §8.4.1.
2. A mesh for ∂D_n is defined to be $\mathcal{T}_{h,n}^{\partial} := \gamma_n(\mathcal{T}_{h,n}^{\mathrm{2D}})$ for all $n \in \{1{:}N_{\mathrm{s}}^{\partial}\}$.
3. The union $\bigcup_{n\in\{1:N_{\mathrm{s}}^{\partial}\}} \mathcal{T}_{h,n}^{\partial}$ is the boundary mesh.
4. Finally, mesh the interior of D by extending the boundary mesh.

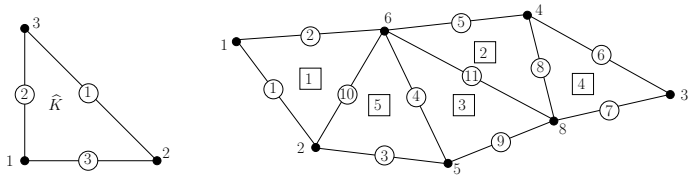
Remark 8.16 (Extruded meshes). Some applications use either cylinders or domains that are homeomorphic to cylinders. A possible strategy to mesh the interior of domains of this type consists of meshing first its right section, which can have any two-dimensional shape, then extruding the mesh of the right section along the generatrix. Depending on the elements chosen to mesh the right section, the volume mesh is typically composed of prisms of triangular or quadrangular base. These prisms can be further decomposed into tetrahedra if needed. □

Exercises

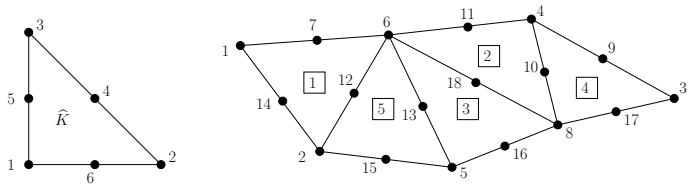
Exercise 8.1 (Curved triangle). Consider the $\mathbb{P}_2$ transformation of a triangle shown in the upper right panel of Figure 8.1. Consider a geometric node of K that is the image of the midpoint of an edge of $\widehat{K}$. Show that the tangent vector to the curved boundary at this node is collinear to the vector formed by the two vertices of the corresponding curved edge. (*Hint*: use the properties of the Lagrange $\mathbb{P}_2$ shape functions.)

Exercise 8.2 (Euler relations). Let $\mathcal{T}_h$ be a matching mesh in $\mathbb{R}^2$ composed of polygons all having ν vertices. (i) Show that $2N_\mathrm{e} - N_\mathrm{e}^\partial = \nu N_\mathrm{c}$. (ii) Combine this result with the Euler relations to show that $N_\mathrm{c} \sim \frac{2}{\nu-2} N_\mathrm{v}$ and $N_\mathrm{e} \sim \frac{\nu}{\nu-2} N_\mathrm{v}$ for fine enough meshes where $N_\mathrm{v}^\partial = N_\mathrm{e}^\partial \ll \min(N_\mathrm{v}, N_\mathrm{e}, N_\mathrm{c})$.

Exercise 8.3 (Connectivity arrays `j_cv`, `j_ce`). Write admissible connectivity arrays `j_cv` and `j_ce` for the following mesh where the face enumeration is identified with large circles and the cell enumeration with squares.



Exercise 8.4 (Connectivity array `j_geo`). Define a connectivity array `j_geo` for the following mesh such that the determinant of the Jacobian matrix of $\boldsymbol{T}_K$ is positive for all the cells.



Exercise 8.5 (Geometric mapping). Let $\boldsymbol{z}_1 := (0,0)$, $\boldsymbol{z}_2 := (1,0)$, $\boldsymbol{z}_3 := (0,1)$, $\boldsymbol{z}_4 := (\frac{1}{3}, \frac{1}{3})$. Consider the triangles $K_1 := \mathrm{conv}(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_4)$, $K_2 := \mathrm{conv}(\boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4)$, and $K_3 := \mathrm{conv}(\boldsymbol{z}_3, \boldsymbol{z}_1, \boldsymbol{z}_4)$. (i) Construct the affine geometric mappings $\boldsymbol{T}_{K_2} : K_1 \to K_2$ and $\boldsymbol{T}_{K_3} : K_1 \to K_3$ s.t. $\boldsymbol{T}_{K_2}(\boldsymbol{z}_1) = \boldsymbol{z}_2$, $\boldsymbol{T}_{K_2}(\boldsymbol{z}_4) = \boldsymbol{z}_4$, and $\boldsymbol{T}_{K_3}(\boldsymbol{z}_1) = \boldsymbol{z}_3$, $\boldsymbol{T}_{K_3}(\boldsymbol{z}_4) = \boldsymbol{z}_4$. (*Hint*: $\boldsymbol{T}_{K_2}$ is of the form $\boldsymbol{T}_{K_2}(\boldsymbol{x}) = \boldsymbol{z}_2 + \mathbb{J}_{K_2}(\boldsymbol{x} - \boldsymbol{z}_1)$.) (ii) Compute $\det(\mathbb{J}_{K_2})\mathbb{J}_{K_2}^{-1}$ and $\det(\mathbb{J}_{K_3})\mathbb{J}_{K_3}^{-1}$. *Note*: the transformation $\boldsymbol{v} \mapsto \det(\mathbb{J}_K)\mathbb{J}_K^{-1}\boldsymbol{v} \circ \boldsymbol{T}_K$ is called contravariant Piola transformation; see (9.9c).