MathematicS

MathS In A.

In Action

G. Balarac, F. Basile, P. Bénard, F. Bordeu, J.-B. Chapelier, L. Cirrottola, G. Caumon, C. Dapogny, P. Frey, A. Froehly, G. Ghigliotti, R. Laraufie, G. Lartigue, C. Legentil, R. Mercier, V. Moureau, C. Nardoni, S. Pertant & M. Zakari

Tetrahedral remeshing in the context of large-scale numerical simulation and high performance computing

Volume 11 (2022), p. 129-164.

https://doi.org/10.5802/msia.22

MathematicS In Action *est membre du*
Centre Mersenne pour l'édition scientifique ouverte
http://www.centre-mersenne.org/
e-ISSN : 2102-5754

MathematicS In Action
Vol. 11, 129-164 (2022)

Tetrahedral remeshing in the context of large-scale numerical simulation and high performance computing

G. Balarac [*], F. Basile [**], P. Bénard [***], F. Bordeu [†], J.-B. Chapelier [††], L. Cirrottola [†††], G. Caumon [‡], C. Dapogny [‡‡], P. Frey [‡‡‡], A. Froehly [§], G. Ghigliotti [§§], R. Laraufie [§§§], G. Lartigue [¶], C. Legentil [¶¶], R. Mercier [¶¶¶], V. Moureau [♭], C. Nardoni [♭♭], S. Pertant [♭♭♭] & M. Zakari [♯]

[*] Univ Grenoble Alpes, CNRS, Grenoble INP, LEGI UMR 5519, 38000 Grenoble, France
[**] ONERA - Université Paris-Saclay, 92322 Châtillon, France; Airbus Operations SAS, 31060 Toulouse, France
[***] CORIA, Normandie Univ, UNIROUEN, INSA Rouen, CNRS, 76000 Rouen, France
[†] SAFRAN Tech, Digital Sciences & Technologies, Rue des Jeunes Bois, 78114 Magny-Les-Hameaux, France
[††] ONERA - Université Paris-Saclay, 92322 Châtillon, France
[†††] INRIA, Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, 33405 Talence cedex, France
[‡] Université de Lorraine, CNRS, GeoRessources, 54000 Nancy, France
[‡‡] Univ Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
[‡‡‡] Sorbonne Université, CNRS, Laboratoire J.L. Lions, UMR 7598, 75005 Paris, France
[§] INRIA Service d'Expérimentation et de Développement, 33405 Talence cedex, France
[§§] Univ Grenoble Alpes, CNRS, Grenoble INP, LEGI UMR 5519, F-38000 Grenoble, France
[§§§] Airbus Operations SAS, 31060 Toulouse, France
[¶] CORIA, Normandie Univ, UNIROUEN, INSA Rouen, CNRS, 76000 Rouen, France
[¶¶] Université de Lorraine, CNRS, GeoRessources, 54000 Nancy, France
[¶¶¶] SAFRAN Tech, Digital Sciences & Technologies, Rue des Jeunes Bois, 78114 Magny-Les-Hameaux, France
[♭] CORIA, Normandie Univ, UNIROUEN, INSA Rouen, CNRS, 76000 Rouen, France
[♭♭] Institut de Recherche Technologique SystemX, 2 Boulevard Thomas Gobert 91120 Palaiseau, France
[♭♭♭] Univ Grenoble Alpes, CNRS, Grenoble INP, LEGI UMR 5519, F-38000 Grenoble, France
[♯] Université de Lorraine, CNRS, GeoRessources, 54000 Nancy, France.

This article is dedicated to the memory of our dear colleague Cécile Dobrzynski

Abstract

The purpose of this article is to discuss several modern aspects of remeshing, which is the task of modifying an ill-shaped tetrahedral mesh with bad size elements so that it features an appropriate density of high-quality elements. After a brief sketch of classical stakes about meshes and local mesh operations, we notably expose (i) how the local size of the elements of a mesh can be adapted to a user-defined prescription (guided, e.g., by an error estimate attached to a numerical simulation), (ii) how a mesh can be deformed to efficiently track the motion of the underlying domain, (iii) how to construct a mesh of an implicitly-defined domain, and (iv) how remeshing procedures can be conducted in a parallel fashion when large-scale applications are targeted. These ideas are illustrated with several applications involving high-performance computing. In particular, we show how mesh adaptation and parallel remeshing strategies make it possible to achieve a high accuracy in large-scale simulations of complex flows, and how the aforementioned methods for meshing implicitly defined surfaces allow to represent faithfully intricate geophysical interfaces, and to account for the dramatic evolutions of shapes featured by shape optimization processes.

Keywords: remeshing, implicit domain meshing, level-set discretization, topology optimization, mesh adaptation, h-adaptation, error estimator, metric, "lagrangian" mesh deformation, distributed memory parallel remeshing, hybrid RANS/LES, LES, geophysical inverse problem.
2020 *Mathematics Subject Classification:* 65X50, 65Y05, 90C06.

1. Introduction

Since the early days of scientific computing, simplicial meshes (that is, meshes composed of triangles in 2d, or tetrahedra in 3d) have been raising a constantly renewed interest as a prominent means to represent and process complex domains. For instance, they have been used extensively to account for a shape in the perspective of its visualization; more recently, `STL` mesh files have become one of the most widespread formats under which shapes are supplied to 3d printing machines. Furthermore, and closer to the scope of the present article, the most popular numerical simulation frameworks for physical phenomena (namely, the finite element method, or the finite volume method) crucially rely on a meshed discretization of the computational domain. We refer for instance to [21] for a general presentation of several stakes and applications of meshing.

In line with this omnipresence of meshes in the numerical treatment of shapes, the issues of mesh generation and mesh processing have received a tremendous amount of attention in the literature, resulting in various efficient algorithms and software packages, such as `Meshlab` [32] `Gmsh` [59], `CGAL` [97], `Tetgen` [93], to name a few free and open-source instances.

Despite these numerous investigations and achievements, meshing is still a thriving field for academic and industrial research: some old and major challenges remain (in particular, it is still unfortunately difficult to create a valid surface triangulation of the boundary of a complex 3d domain, then to fill the volume with a valid tetrahedral mesh) while modern applications suggest new and promising directions for investigations.

Many such applications fit in with the context of *remeshing*, where one aims to modify an existing, valid, albeit ill-shaped and badly sampled mesh, so that it better comply with given requirements, such as a good element quality, an adapted element density, etc. The purpose of this article is to discuss and illustrate several such aspects of tetrahedral remeshing. These, as well as the proposed means to address them, are suggested by the knowledge of the authors, without ambition of exhaustivity.

(1) The primary target of remeshing is to adapt the local size of a mesh (i.e. its element density) to the geometric features of the represented domain, or to an error estimate associated to a numerical simulation of interest.

(2) When a physical simulation implies an evolution of the underlying domain (as it is often the case for instance in computational fluid dynamics, or in shape optimization), suitable combinations between Lagrangian strategies and remeshing techniques make it possible to efficiently account for this motion.

(3) The recent development of Eulerian interface-capturing methods (such as the celebrated level set method) has made it quite popular to represent a shape Ω *implicitly*, i.e. as the negative subdomain of a scalar function $\phi : D \to \mathbb{R}$, defined on (a fixed mesh of) a given computational domain D. In this context, one may need an exact mesh of Ω, which calls for a methodology for constructing a mesh of such an implicit domain.

(4) When the size of the computational mesh is so large that it can barely be stored in memory, it is crucial that the remeshing process be carried out in a parallel fashion.

Understandably enough, these issues find particularly relevant applications in the recent context where the sustained increase in computational power and the advent of high-performance computing herald high resolution, very accurate numerical simulations of complex physical phenomena. The presentation of this article is guided by this perspective of large-scale simulations. In particular, it mainly takes place in the three-dimensional context, which is on every aspect more involved than its two-dimensional counterpart, to which we shall occasionally refer for pedagogical purposes, or as a preliminary step towards a future three-dimensional implementation. We discuss the use of the presented remeshing features in the context of applications involving

high performance computing; all these features are implemented in the open-source software `mmg` [3], which is used consistently in this exposition. A thorough technical description of this environment can be found in the article [37], see also [36, 43].

This article is organized as follows. In the next Section 2, we briefly review some basic material about meshes and remeshing, before describing a little more precisely some modern applications in different contexts of use. We then present several applications of these methods, and variants of them dedicated to the context of high-performance computing: in Section 3, an error estimate based adaptive remeshing strategy is implemented in the context of unsteady fluid mechanics simulations. The next Section 4 arises in the context of shape and topology optimization, where an implicit domain meshing methodology is carried out to track the motion of the optimized domain, while allowing for an explicit, meshed description of the latter throughout the optimization iterations. The application of this idea of meshing implicitly-defined interfaces is then applied in the field of geoscience in Section 5, to the construction of an explicit and accurate meshed representation of geophysical interfaces. Finally, in Section 6, we exemplify how a parallel implementation of mesh adaptation techniques makes it possible to track efficiently the motion of complex fluid interfaces, such as a turbulent flame front or a liquid droplet, with a very high resolution.

2. A few aspects of tetrahedral remeshing

In this section, we present the remeshing features at stake in this article. After a short summary of basic concepts about meshes in Section 2.1 and a general presentation of remeshing in Section 2.2, we describe in Section 2.3 the issue of mesh adaptation. In the next Section 2.4, the problem of mesh displacement is discussed; in Section 2.5 we broach the idea of isosurface discretization before finally dealing with parallel remeshing in Section 2.6.

2.1. General facts about meshes and notation

Let us start by introducing briefly the needed background material about tetrahedral meshes in this article; for these issues, we refer to e.g. [20, 27, 55, 68, 98] for further details.

Formally, let $\Omega \subset \mathbb{R}^3$ be a bounded domain; a *mesh* of Ω is a collection $\mathcal{T} = \{T_i\}_{i=1,\dots,N_T}$ of open tetrahedra, accounting for a covering of Ω, in the sense that

$$\overline{\Omega} = \bigcup_{i=1}^{N_T} \overline{T_i}.$$

In addition, it is assumed that

- $\mathcal{T}$ is *valid*: the open simplices T_i are mutually disjoint: $T_i \cap T_j = \emptyset$, $i \neq j$,
- $\mathcal{T}$ is *conforming*: each intersection $\overline{T_i} \cap \overline{T_j}$, $i \neq j$, is reduced to either a vertex, an edge, or a face of the mesh.

A mesh $\mathcal{T}$ additionally bears information about a surface triangulation $\mathcal{S}$, that is, a collection $\{S_j\}_{j=1,\dots,N_S}$ of triangles $S_j \subset \mathbb{R}^3$ accounting for one or several pieces of surface. In the most simple instances, $\mathcal{S}$ is made of the external faces of the tetrahedra $T \in \mathcal{T}$, as a (approximate) representation of the boundary of Ω. However, in some applications, Ω comprises several subdomains, whose tetrahedra are identified with different labels. In such cases, $\mathcal{S}$ also contains the triangles of the corresponding inner boundaries, see Figure 2.5 (right) below for an example.

Usually, the creation of a mesh $\mathcal{T}$ of Ω starts from the datum of a surface triangulation $\mathcal{S}$ of the boundary $\partial\Omega$ (which is often supplied by a CAD software). Thence, several strategies are available to fill the volume of Ω with tetrahedra conforming to this triangulation, the perhaps most efficient ones being based on the constrained Delaunay algorithm, see for instance [27, 55]

for detailed presentations of the latter. Let us mention that, in spite of having been extensively addressed for decades, mesh generation is still a delicate issue when very intricate shapes are considered.

Beyond the aforementioned mild requirements, it is often desirable to evaluate more closely "how well" a mesh $\mathcal{T}$ lends itself to accurate numerical simulations. This feature can be appraised in at least two independent ways, which are illustrated on Figure 2.1.

- Most numerical methods experience trouble when the mesh $\mathcal{T}$ contains very flat, nearly degenerate elements. For instance, such configurations are well-known to increase dramatically the condition number of finite element systems based on this mesh, thus slowing down iterative matrix solvers, see e.g. the books [31, 49] about this classical issue. Several quantities are used in the literature to discriminate "well-shaped elements" T_i (i.e. those that are close from being regular tetrahedra), from "ill-shaped" ones (i.e. that are nearly degenerate); for instance, the following measure of the quality of a tetrahedron T is quite popular in the literature:

$$\mathcal{Q}(T) = \alpha \frac{\mathrm{Vol}(T)}{\left(\sum_{i=1}^{6} |e_i|^2\right)^{\frac{3}{2}}},$$

  where e_i, $i = 1, \ldots, 6$ are the edges of T, $\mathrm{Vol}(T)$ is its volume, and α is a normalization factor. This quantity $\mathcal{Q}(T)$ equals 1 when T is regular, and it is close to 0 when T is nearly degenerate.

- Another crucial feature in the evaluation of the quality of $\mathcal{T}$ is related to an issue which we have overlooked so far. Often, the domain Ω of interest is smooth, and has a curved boundary $\partial\Omega$, while the faces of the surface triangulation $\mathcal{S}$, intended as a discrete approximation of the latter, are planar. Hence, it should be required from $\mathcal{S}$ that it be a close approximation of the smooth surface $\partial\Omega$ up to a certain tolerance, for instance in terms of the Hausdorff distance between $\mathcal{S}$ and $\partial\Omega$.

Summarizing, we shall expect from a mesh $\mathcal{T}$ of Ω that each tetrahedron T_i have quality $\mathcal{Q}(T_i)$ close to 1, and that the surface triangulation $\mathcal{S}$ be a "close" approximation of the continuous surface $\partial\Omega$.

2.2. **Basic stakes about remeshing**

As suggested by the name, remeshing assumes the datum of a tetrahedral mesh $\mathcal{T}$ of Ω which is valid and conforming, but may still be unsuitable for computation: as discussed in Section 2.1, $\mathcal{T}$ may suffer from poor element quality, or its element density may be inadequate, in the sense that curved regions of the boundary $\partial\Omega$ (or other, internal surfaces) are represented by too few elements. Remeshing aims to modify $\mathcal{T}$ into a high-quality mesh $\widetilde{\mathcal{T}}$ of Ω, whose element density is well-tailored to its geometric features.

This objective is usually achieved thanks to a series of local operations, which are applied iteratively; these are briefly described below, and illustrated on Figure 2.2 in the (simpler) two-dimensional context.

- *Edge split:* When an edge pq is "too long", a new vertex m is inserted in the mesh $\mathcal{T}$; pq is replaced by the two edges pm and mq and the connections of $\mathcal{T}$ are updated accordingly.

- *Edge collapse:* When an edge pq is "too short", its endpoints p and q are merged and the connections of the mesh $\mathcal{T}$ are updated accordingly.

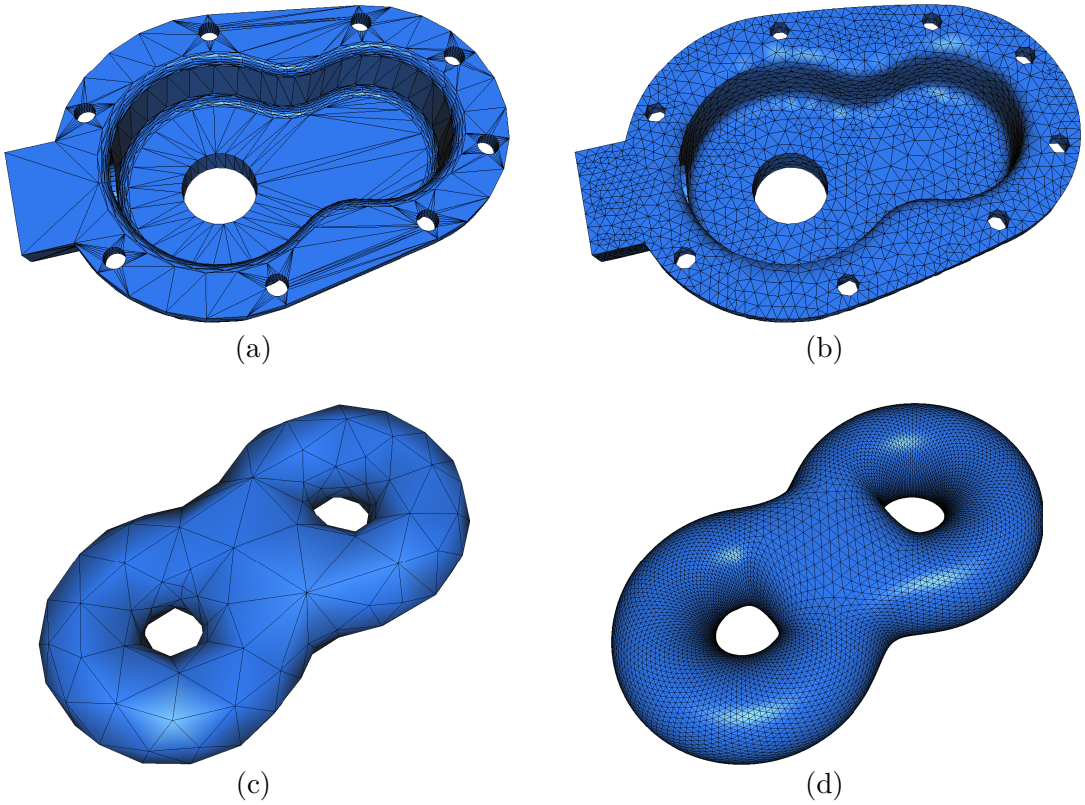

Figure 2.1. (a) Fine geometric approximation of a domain Ω_1, with a mesh containing bad quality elements; (b) good quality mesh of Ω_1; (c) good quality mesh of a domain Ω_2, accounting for a poor geometric approximation of Ω_2; (d) mesh of Ω_2 with a good geometric approximation property.

- *Edge swap:* An edge pq is removed from the mesh $\mathcal{T}$ and the "shell" of pq, consisting of all the tetrahedra sharing this edge, is adequately reconnected.

- *Vertex relocation:* A vertex p is moved slightly, while all its connections remain unaltered.

Each of these operators exists under two different versions, depending on whether it is applied to a surface configuration, involving tetrahedra bearing surface triangles $S_j \in \mathcal{S}$, or to an internal one. For instance, when an internal edge pq of $\mathcal{T}$ is split, the inserted point m is usually chosen as the midpoint of p and q; on the contrary, when $pq \in \mathcal{S}$ is a boundary edge, m is rather placed on the continuous surface $\partial\Omega$. In practice, since this "ideal" surface is unknown, the position of m is inferred from those of p and q, and other associated geometric quantities (such as the normal vectors at p and q).

In the above description, the criterion whereby an edge pq is deemed to be "too long" (or "too short") may depend on the situation. For instance, pq may be "too long" with respect to a user's prescription for the size of the elements of the mesh (see the next Section 2.3 about this practice) or, when $pq \in \mathcal{S}$ is a surface edge, it may be considered to be "too long" with respect to the curvature of $\partial\Omega$, as its size entails a too coarse geometric approximation of $\partial\Omega$.

Last but not least, let us emphasize that the use of the above remeshing operators should be carefully monitored: several checks are in order so as to prevent the emergence of invalid configurations.

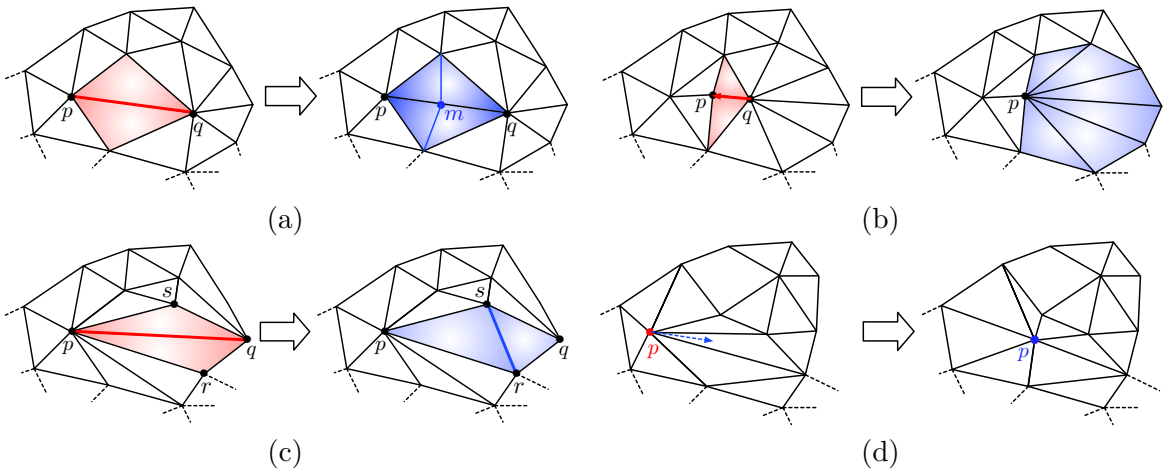

Figure 2.2. (a) Split of the "long" edge pq: a new point m is inserted in $\mathcal{T}$ and the triangles sharing this edge are subdivided accordingly; (b) collapse of point q onto p: the two red triangles disappear, and q is replaced by p in all the other triangles connected to q (in blue); (c) swap of the edge pq (in red): the connection pq is traded for the alternative configuration, featuring the edge rs (in blue); (d) relocation of the vertex p while maintaining all the connections in the mesh.

2.3. **Goal-oriented mesh adaptation**

As we have seen, the primary objective of remeshing $\mathcal{T}$ is to produce a mesh $\widetilde{\mathcal{T}}$ with fine element quality, which is a close approximation of the underlying domain Ω. In addition to these requirements, one usually demands that $\widetilde{\mathcal{T}}$ should comply with a user-defined local size prescription, which may be either isotropic or anisotropic:

- In the former case, the size prescription is encoded as a *size map* $h : \Omega \to \mathbb{R}$, which is often supplied at the vertices of $\mathcal{T}$ and interpolated from these data when necessary: $h(x)$ is the desired size for the edges near the point $x \in \Omega$.

- In the latter case, the local size is imposed under the form of a *metric tensor* $M : \Omega \to \mathbb{R}^{3\times 3}$: the eigenvalues of the symmetric, positive definite matrix $M(x)$ encode the desired size near $x \in \partial\Omega$, in the directions of the corresponding eigenvectors; see [99] about this Riemannian framework and [70] for an interesting continuous paradigm based on this idea.

This extra ingredient is incorporated into the general remeshing framework of Section 2.2 via the calculation of the length of an edge pq when deciding whether it is "too long" or "too short". For instance, the length $\ell_h(pq)$ of an edge pq with respect to an isotropic size map $h : \Omega \to \mathbb{R}$ equals

$$\ell_h(pq) := \int_0^1 \frac{|\gamma'(t)|}{h(\gamma(t))}\,\mathrm{d}t,$$

where $\gamma : [0,1] \to \mathbb{R}^3$ is an arbitrary parametrization of the segment pq such that $\gamma(0) = p$ and $\gamma(1) = q$. Hence, $\ell_h(pq) \gg 1$ (resp. $\ell_h(pq) \ll 1$) when pq is "too long" (resp. "too short") with respect to the desired local size.

Whether it is isotropic or anisotropic, the size prescription may stem from various considerations. Here are two examples:

- *Geometric error estimate:* The local size of the edges of $\mathcal{T}$ should guarantee that the geometric approximation of the smooth boundary $\partial\Omega$ of Ω by the surface triangulation

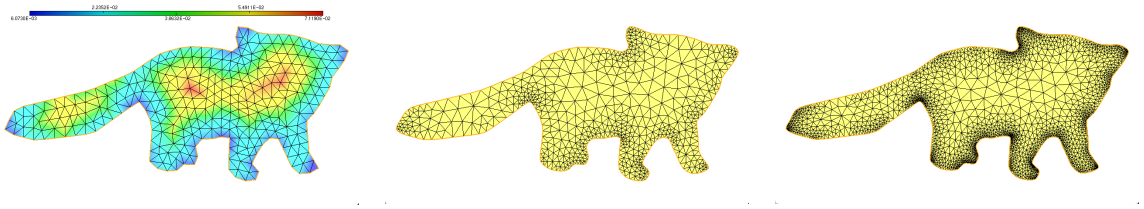

FIGURE 2.3. (Upper row) Adaptation of a mesh with respect to a geometric size prescription; (left) initial mesh with size map, calculated on the basis of the geometric approximation; (middle) adapted mesh with respect to this size prescription; (right) adapted mesh with respect to a finer geometric approximation requirement.

$\mathcal{S}$ is accurate enough, for instance, that the Hausdorff distance between $\partial\Omega$ and $\mathcal{S}$ is small; see Figure 2.3 for an example.

- *A priori or a posteriori error estimate:* In the perspective of reducing the computational effort of physical simulations, the local size should be adapted to a surrogate quantity for the error $\|u - u_h\|$ between the solution u to a partial differential equation posed on Ω and its finite element approximation u_h. This surrogate quantity may depend on u (a priori estimate) or u_h (a posteriori estimate); see for instance [49] for an introduction to this practice.

Eventually, let us observe that while isotropic remeshing is by now a very popular idea for tackling realistic and industrial applications, the use of an anisotropic size prescription seems more limited, since even a slight misalignment of the resulting, very stretched elements may jeopardize with the accuracy of the numerical computation, see e.g. [84, 92] about this issue.

2.4. **"Lagrangian" mesh deformation**

Many time-dependent physical phenomena imply an evolution of the simulation domain Ω; for instance, Ω may represent a volume filled by a fluid whose velocity is predicted by the resolution of the Stokes, or the Navier–Stokes equations.

In this spirit, various applications demand to realize the motion of a domain Ω^n, equipped with a mesh $\mathcal{T}^n$, according to a velocity field $V^n : \Omega^n \to \mathbb{R}^3$, and to obtain a mesh $\mathcal{T}^{n+1}$ of the next domain $\Omega^{n+1} := (\mathrm{Id} + V^n)(\Omega^n)$. Here and throughout this section, the superscript ^{n} refers to discrete time. As we have mentioned, in practice, V^n results from a physical simulation, conducted on the mesh $\mathcal{T}^n$; for the purpose of this section, we suppose that it is given, as a (time independent) vector field defined at the vertices of $\mathcal{T}^n$.

Realizing this motion is a highly challenging task, which has been extensively considered in the literature, see for instance [4, 11, 35, 44, 72]. The perhaps most intuitive strategy consists in intertwining steps where each vertex $p \in \mathcal{T}^n$ is pushed in the direction of the vector $V^n(p)$ as long as the resulting mesh is valid (which quickly deteriorates the quality of $\mathcal{T}^n$), with remeshing steps for improving the quality of the resulting mesh, thereby allowing to reiterate the process. One possible procedure based on this principle outlines as follows, see also Figure 2.4:

(1) Find the largest number $0 < t^* \leq 1$ such that moving each vertex $p \in \mathcal{T}^n$ to $p + t^* V^n(p)$ results in a valid mesh;

(2) Apply all or just one subset of the remeshing operations of Section 2.2 in order to improve the quality and the density of the resulting mesh;

(3) If $t^* < 1$, go back to (1) by replacing the vector field $V^n(x)$ with $(1 - t^*)V^n(x)$.

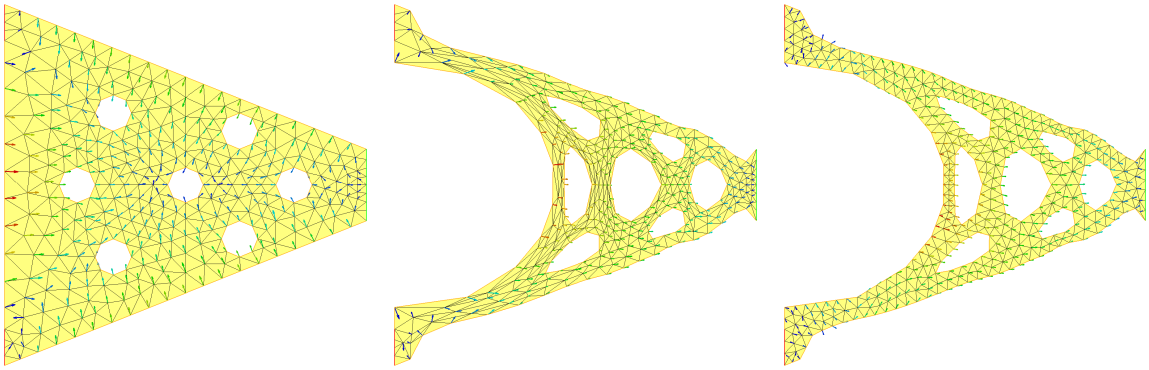

FIGURE 2.4. Evolution of a domain Ω (an elastic cantilever beam) via a velocity field V^n (supplied by the resolution of a shape optimization problem); the vertices of the mesh $\mathcal{T}^n$ are displaced according to V^n. (Left) initial shape and associated deformation field; (middle) at iteration 150, the mesh becomes very stretched; the resolution of the linear elasticity equation is very inaccurate, and the deformation cannot be applied lest that the mesh becomes invalid; (right) after quality-oriented remeshing, the deformation process is able to go on.

A number of additional heuristics may help this process, postponing the emergence of overlapping elements, i.e. allowing t^* to be as close to 1 as possible during the first step. One popular practice relies on the fact that only the values of the velocity field $V^n(p)$ at those vertices p in the surface part $\mathcal{S}^n$ of $\mathcal{T}^n$ have an influence on the geometry of Ω^{n+1}, so that the motion of the internal vertices of $\mathcal{T}^n$ can actually be chosen arbitrarily. Among such possibilities, extending the values of V^n on $\mathcal{S}^n$ to the inner nodes of $\mathcal{T}^n$ by solving a linearized elasticity system with Dirichlet data V^n on $\mathcal{S}^n$ tends to produce a motion with little compression, thus mitigating the trend of elements to degenerate.

"Reasonably large" displacements of Ω can be realized owing to such Lagrangian strategies. Yet, a number of situations are difficult to handle, especially when the considered motion implies topological changes (e.g. the merger of two holes). A more robust strategy, combining remeshing techniques with an implicit representation of the evolving domain, is presented in the next Section 2.5 and Section 4.

2.5. **Implicit domain remeshing**

In several applications, the domain Ω of interest is not readily defined by a meshed representation. Rather, a large computational box D is introduced, which is equipped with a mesh $\mathcal{T}$, and Ω is defined as the negative subdomain of a scalar "level set" function $\phi : D \to \mathbb{R}$ (in practice, defined at the vertices of $\mathcal{T}$), that is:

$$\forall\, x \in D, \quad \begin{cases} \phi(x) < 0 & \text{if } x \in \Omega, \\ \phi(x) = 0 & \text{if } x \in \partial\Omega, \\ \phi(x) > 0 & \text{if } x \in D \setminus \overline{\Omega}. \end{cases} \tag{2.1}$$

This type of representation is ubiquitous in e.g. geometric modeling or shape reconstruction. It has raised a tremendous enthusiasm within the numerical simulation community as the pivotal ingredient of the celebrated *level set method* [82], devoted to the description of the motion of a domain $\Omega(t) \subset D$ according to a velocity field $V(t, x)$. Indeed, introducing a level set function $\phi(t, \cdot)$ for $\Omega(t)$ (i.e. (2.1) holds at any time $t > 0$), this motion translates into the following

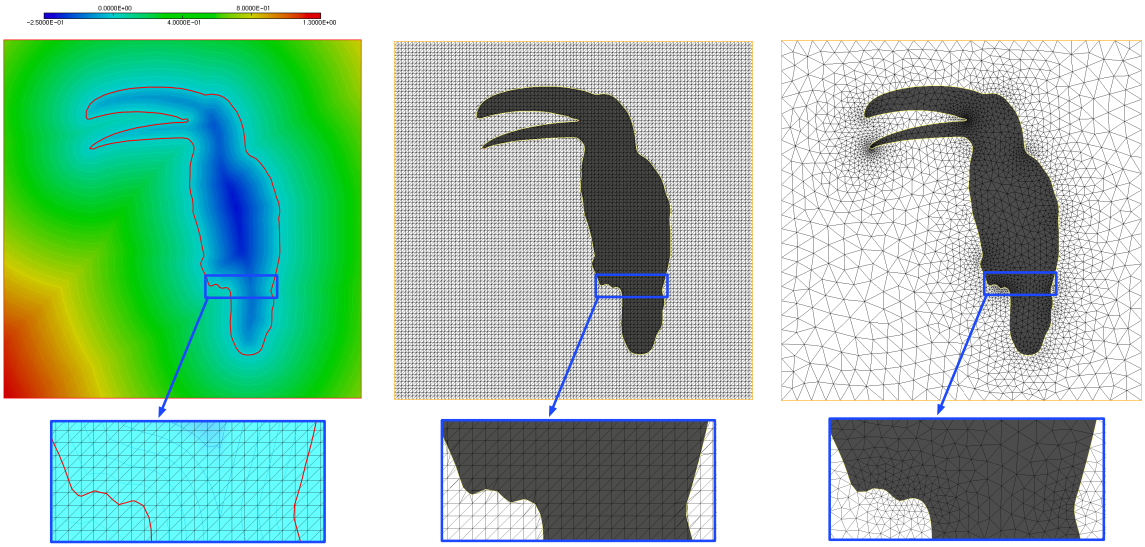

Figure 2.5. (Left) One "level set" function ϕ is defined at the vertices of the mesh $\mathcal{T}$ of the square-shaped computational domain D; its 0 level set (which is not explicitly discretized in $\mathcal{T}$) is depicted in red; (middle) explicit discretization of the 0 level set of ϕ into the mesh $\mathcal{T}$; the intermediate, low-quality mesh $\mathcal{T}_{\text{temp}}$ is obtained; (right) high-quality mesh $\widetilde{\mathcal{T}}$ obtained after remeshing $\mathcal{T}_{\text{temp}}$.

advection-like equation:

$$\frac{\partial \phi}{\partial t}(t,x) + V(t,x) \cdot \nabla \phi(t,x) = 0, \quad t > 0, \ \ x \in D. \tag{2.2}$$

Hence, a geometric evolution problem is reformulated as a partial differential equation posed on a fixed domain D, equipped with the fixed mesh $\mathcal{T}$; see for instance [81, 91] for comprehensive introductions to the level set method.

Depending on the application, it may be of utmost importance to recover a mesh of Ω from the data of D and ϕ; this is the case when for instance Ω is intended as the domain of physical phenomenon, whose accurate numerical simulation is desired.

It is actually possible to modify the mesh $\mathcal{T}$ of D into a new (valid, conforming) "body-fitted" mesh $\widetilde{\mathcal{T}}$ of D where both Ω and its complement $D \setminus \overline{\Omega}$ are explicitly discretized. This may be achieved within two steps, as illustrated on Figure 2.5:

(1) Discretize the isosurface $\{x \in D, \ \phi(x) = 0\}$ into the mesh $\mathcal{T}$. This stage is simple and relies the *marching tetrahedra* algorithm [45], as a variant of the well-known marching cubes algorithm [69]: in a nutshell, each tetrahedron $T \in \mathcal{T}$ crossed by this isosurface is subdivided according to a pattern. This ends up with a valid, conforming mesh $\mathcal{T}_{\text{temp}}$ of D, where Ω is explicitly discretized, but which generally has very bad quality.

(2) Remesh $\mathcal{T}_{\text{temp}}$ into a fine quality mesh $\widetilde{\mathcal{T}}$ of D, where Ω is explicitly discretized.

2.6. **Parallel remeshing**

As the range of applications of physical simulations grows wider, and more and more realistic applications are targeted, it is natural that very large meshes need to be handled and processed. This raises the issue of modifying such a large mesh $\mathcal{T}$ in a parallel fashion.

In this direction, two quite different paradigms can be thought of. On the one hand, according to *shared memory* strategies, the whole mesh data are stored on one single node; they are shared

by the different *processes* or *threads*, and the remeshing operations are carried out in parallel. On the other hand, *distributed memory* strategies advocate to divide $\mathcal{T}$ into several regions with shared interfaces, which are independently remeshed on the different processors. Each of these two strategies faces new, specific issues with respect to sequential remeshing algorithms, such as the prevention of "parallel data races" (i.e. multiple processes trying to adapt the same mesh entities) in the implementation of shared memory algorithms, and the preservation of mesh conformity at the interface between regions treated on different processes in distributed memory strategies.

In practice, physical simulation solvers are often parallelized over distributed memory architectures. Hence, their combination with a sequential or shared memory implementation of remeshing would imply to gather the whole mesh on one process, call the remesher and last, redistribute the mesh onto each parallel process before calling the solver. The intense exchanges of memory entailed in doing so are a well-known performance bottleneck for remeshing-based numerical simulation strategies, see [84] for a discussion. Moreover, the considered mesh is sometimes so large that it cannot be stored in the memory of one single node, thus ruling out the very possibility to use a sequential, or a shared memory implementation. These major drawbacks plead in favor of distributed memory parallel strategies for remeshing.

Conducting the remeshing of $\mathcal{T}$ over a distributed memory architecture first requires to partition $\mathcal{T}$ into disjoint submeshes $\mathcal{T}_k$, $k = 1, \dots, K$: no tetrahedra $T \in \mathcal{T}$ belongs to two different $\mathcal{T}_k$. The surface triangles at the interface between two of these submeshes constitute a surface mesh $\mathcal{O}$, also referred to as *parallel interface*. The $\mathcal{T}_k$ are sent and treated on different processes, or *ranks*; this raises the following issues:

- The imbalance between the amounts of work conducted on each rank can hardly be predicted. Most often, the modifications involved in the remeshing process are non uniformly distributed over $\mathcal{T}$ (this is the case when, for instance, goal-oriented mesh adaptation is performed, see Section 2.3) and it is difficult to propose a priori an even repartition of the amount of operations across the $\mathcal{T}_k$. For instance, balancing the number of elements between ranks does not guarantee a balance of the work loads.

- Enforcing the conformity of the reunion of the submeshes $\mathcal{T}_k$ requires specific parallel data structures (e.g. a table maintaining the correspondence between the shared surface triangles $S \in \mathcal{O}$ and the supporting tetrahedra pertaining to different ranks), whose management is intricate. Notably, their update requires some parallel communication and synchronization between ranks when one tetrahedron T bearing an interface entity $S \in \mathcal{O}$ is modified, and a careful reconstruction of the parallel interface $\mathcal{O}$ is in order when the partition of $\mathcal{T}$ is modified (as is the case in the remeshing-repartitioning parallel strategies broached below).

These concerns open the way to two strategies for conducting remeshing in parallel over distributed memory architectures, which essentially differ by their treatment of the entities $S \in \mathcal{O}$ at the interface between two of the submeshes $\mathcal{T}_k$.

- On the one hand, each remeshing operator could be applied in parallel to the entities in $\mathcal{O}$, see e.g. [23, 30, 39, 80]. Doing so requires a tight communication between the various processes sharing the considered configuration, in order to check the validity of the realized operation and to update consistently the mesh connectivity;

- On the other hand, an iterative remeshing-repartitioning (or moving-interface) strategy could be used, which intertwines
  - A parallel remeshing of each domain $\mathcal{T}_k$ on the associated process, while keeping the entities in $\mathcal{O}$ unmodified;

– A new subdivision of the mesh $\mathcal{T}$, creating new submeshes $\mathcal{T}_k$ and interface triangulation $\mathcal{O}$;

see [14, 25, 42, 52] about this approach, and Figure 2.6 for a 2d illustration.
It is expected that the repetition of this procedure allows each region of the mesh to be modified. In this direction, the most crucial issues lie in the repartitioning operations. Indeed, the remeshing of each region $\mathcal{T}_k$ simply makes use of a sequential algorithm such as those presented in Section 2.2, but a fluid migration of the domain interfaces from one rank to the other is mandatory to ensure that no element in the mesh stay at the interface between submeshes, lest that it would stay unmodified. Moreover, this interface migration procedure must ensure a fair balance of the work load between ranks in terms of CPU cost, while keeping the amount of migrating data low, insofar as possible.

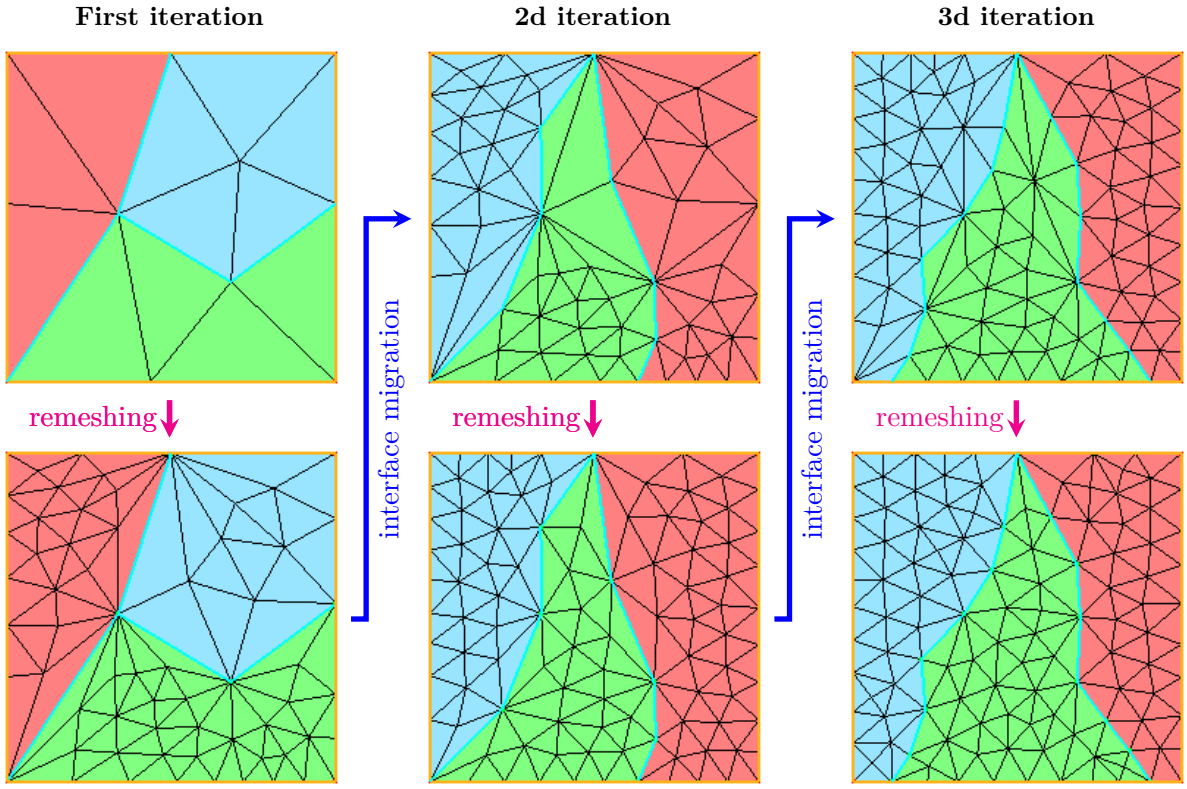

Figure 2.6. Illustration of an iterative remeshing-repartitioning parallel mesh adaptation strategy performed over 3 processors: at each iteration, the mesh is divided into 3 submeshes $\mathcal{T}_k$, $k = 1, 2, 3$, which are distributed over as many processors; each $\mathcal{T}_k$ is then remeshed, independently of the other two, while the entities at the rank interfaces (cyan edges) are preserved.

3. h-adaptive RANS and hybrid RANS/LES simulations of a nozzle with the Discontinuous Galerkin method using unstructured meshes and isotropic mesh adaptation

In this section, we present a mesh adaptation strategy devoted to the solution of the compressible steady Reynolds-Averaged Navier–Stokes (RANS) and the unsteady Zonal Detached

Eddy Simulation (ZDES) equations on hybrid prismatic/tetrahedral grids using Discontinuous Galerkin (DG) methods, in the context of realistic, industrial applications. The developed mesh adaptation algorithm is applied to RANS and hybrid RANS/LES simulations on the PPRIME nozzle configuration for a DG formulation featuring elements with polynomial degree $p = 1$.

3.1. **Context and motivation**

The work presented in this section is motivated by the search for a simple mesh adaptation algorithm to improve the computational efficiency of DG simulations for complex flow configurations relevant in an industrial context, where the necessity of keeping the simulation cost "reasonable" is of utmost importance.

Steady RANS simulations are well established, and extensively used for industrial purposes. Nevertheless, they may fail to capture the turbulence and noise generation mechanisms which are often required in the design process. On the other hand, scale-resolving simulations (relying for instance on the DNS (Direct Numerical Simulation), LES (Large Eddy Simulation), or hybrid RANS/LES methods [89]), are capable of capturing the unsteady features in transitional flows, gas turbine combustors and nozzles. In this work, we assess both the steady RANS [96] and the ZDES hybrid RANS/LES models [40], in order to reduce the computational cost of the DNS and wall-resolved LES turbulence modeling approaches, which require a very large (often intractable in practice) number of degrees of freedom in space and time for capturing the smaller structures developed in the boundary layer.

DG methods are particularly suited for turbulent flow simulations thanks to their good dispersion and dissipation properties. In addition, these methods provide a high order of accuracy on arbitrary unstructured meshes, are suitable for parallel computing thanks to their compact stencil and provide a natural framework for *hp*-adaptation; not only the size h of the elements of the mesh can be adapted but also their polynomial degree p [46, 66, 100].

The new generation compressible flow solver CODA is employed to compute the flow solution and the error estimators on an unstructured mesh. The CODA solver, designed for an efficient use on current and future parallel HPC systems, is developed in partnership by Airbus, ONERA and DLR [67] and targets academic and industrial aerodynamic problems. The object-oriented CODA framework permits the integration of two spatial discretizations: a second-order Finite Volume (FV) and a DG discretization with variable order, applied to the Euler, Navier–Stokes, RANS and hybrid RANS/LES equations.

3.2. **Mesh adaptation algorithm**

3.2.1. *Presentation of the mesh adaptation strategy*

The general strategy is to adapt the computational mesh with respect to a relatively affordable steady RANS simulation, which already captures some important features of the targeted unsteady simulation. This mesh is then used as the computational support of an accurate, albeit more expensive hybrid RANS/LES simulation.

The mesh adaptation process implemented in this work outlines as follows. A first steady RANS simulation is carried out on a very coarse initial mesh. An a posteriori error estimator aimed at controlling the accuracy of the solution in the elements of the mesh, is then computed; the latter is described in the next Section 3.2.2. It is used to define a new mesh size prescription guiding the refinement of the elements where the error is high, as we discuss in Section 3.2.4. Remeshing is then conducted on the basis of this datum. The previous flow solution is projected onto the obtained mesh, and the process is iterated: a new steady RANS simulation is performed, etc.

Five such RANS adaptation steps are carried out, and a hybrid RANS/LES simulation is eventually realized on the finest mesh adapted to the RANS equation.

Note that the present strategy, whereby the accurate and expensive hybrid RANS/LES analysis is performed only on the finest mesh, which is adapted with respect to the relatively cheap RANS simulation, is preferred over adapting the mesh from the beginning with respect to the hybrid RANS/LES simulation. Indeed, an initial very coarse mesh would prevent the turbulent structures of the flow from developing and yield numerical instabilities as well as a dramatic increase in computational time for the whole adaptation process.

Note that this idea of employing adapted meshes obtained from steady simulations at a low computational cost, as the starting point of an unsteady turbulent adaptation procedure is expected to be a robust and reasonably cheap means to achieve a highly accurate hybrid RANS/LES simulation. The evaluation of this methodology will be addressed in future research.

3.2.2. *Description of the error estimator for the steady RANS simulation*

Several indicators based on the discretization error have been developed in the DG simulation literature. These types of error indicators are convenient thanks to their efficiency, locality, simplicity and low computational cost [58, 71, 85, 87]. The error estimator employed in this work is made of two contributions, similarly to what has been proposed in [34, 13, 12]: the first one is based on the measure of the energy associated with the highest-order polynomial modes, the Small Scale Energy Density (SSED) [78], while the second one relies on the inter-element jumps (JUMP) [17] of the momentum. The resulting error indicator is local, inexpensive and flexible, in the sense that an error indicator based on the highest order modes of the solution is more reliable for a high polynomial degree p, while a jump-based error estimator is accurate for every value of p. The two aforementioned contributions are normalized by their respective maximum and minimum values over the tetrahedra $T \in \mathcal{T}$, so that the considered error estimator finally reads:

$$\epsilon_T = \frac{\epsilon_{\text{SSED},T} - \min_{T' \in \mathcal{T}}(\epsilon_{\text{SSED},T'})}{\max_{T' \in \mathcal{T}}(\epsilon_{\text{SSED},T'}) - \min_{T' \in \mathcal{T}}(\epsilon_{\text{SSED},T'})} + \frac{\epsilon_{\text{JUMP},T} - \min_{T' \in \mathcal{T}}(\epsilon_{\text{JUMP},T'})}{\max_{T' \in \mathcal{T}}(\epsilon_{\text{JUMP},T'}) - \min_{T' \in \mathcal{T}}(\epsilon_{\text{JUMP},T'})} \quad (3.1)$$

The quantity ϵ_T is naturally defined at the level of the tetrahedra $T \in \mathcal{T}$. In practice, most remeshing strategies rely on error estimators attached to the vertices of the computational mesh. A consistent value ϵ_x may be attached to a vertex x by using a volume-weighted average of the values ϵ_T at the elements sharing x as vertex. This indicator is the key ingredient in the definition of the imposed size h_x^*, as we describe next in Section 3.2.4.

3.2.3. *Description of the mesh adaptation loop*

As we have mentioned, five mesh adaptation steps are performed. At the end of the RANS simulation of step n, the element-based and vertex-based error estimators ϵ_T^n and ϵ_x^n are computed, and the next imposed size h_x^{n+1} is defined (see Section 3.2.4). The computational mesh $\mathcal{T}^n$ available at step n is eventually remeshed, resulting in the new mesh $\mathcal{T}^{n+1}$.

The meshes $\mathcal{T}^n$ are made of 2 regions: while most of the computational domain is filled with tetrahedra, another region, associated to the boundary layer of the physical phenomenon under scrutiny is composed of structured or pseudo-structured (i.e. prismatic) elements. At each step n, only the tetrahedral part of $\mathcal{T}^n$ is remeshed. Moreover the user may decide not to remesh some of the tetrahedra in the computational domain. Summarizing, the computational mesh $\mathcal{T}^n$ is split into two parts:

$$\mathcal{T}^n = \mathcal{T}^n_{\text{free}} \cup \mathcal{T}^n_{\text{fixed}} \quad (3.2)$$

where $\mathcal{T}^n_{\text{free}}$ is the tetrahedral zone subject to remeshing at step n, and $\mathcal{T}^n_{\text{fixed}}$ contains the constituent prisms of the boundary layer mesh, as well as the fixed tetrahedra.

3.2.4. *Size prescription for the steady RANS simulation*

The size prescription is based on the idea, already present in [17] and [88], that the error ϵ_T attached to each element $T \in \mathcal{T}$ converges to zero with the rate $p+1$, when no geometrical or physical discontinuities are observed.

At each adaptation step n, the imposed size h_x^{n+1} at the vertex $x \in \mathcal{T}$ is updated according to the rule:

$$h_x^{n+1} = h_x^n \left(\frac{\epsilon^{n+1,*}}{\epsilon_x^n} \right)^{\frac{1}{p+1}}, \tag{3.3}$$

where $\epsilon^{n+1,*}$ is a maximum value for the error, which is imposed globally to all the elements of the mesh, with the aim to provide a fixed increase in the number of elements of the mesh at each adaptation step [12, 17]. Adopting this strategy, the regions showing a larger value of the error estimator than the imposed target $\epsilon^{n+1,*}$ are refined by a factor depending on the ratio $\epsilon^{n+1,*}/\epsilon_x^n$ between this target and the actual value of the error, while regions characterized by an error lower than $\epsilon^{n+1,*}$ are left unchanged.

3.3. **Numerical example: simulation of the subsonic turbulent nozzle jet flow**

We appraise the above adaptive remeshing strategy in the physical context described in [22], for which experimental and numerical data are available in the literature, see also [57] and [79] for further investigations aimed at predicting turbulence generation and jet noise.

The main goal of this study is to simulate the turbulent isothermal subsonic jet issued from a nozzle with exit diameter $d_N = 0.05m$. The nozzle has an axial symmetry with respect to the first coordinate axis e_1 (where (e_1, e_2, e_3) stands for the canonical basis of $\mathbb{R}^3$ and we denote by (x_1, x_2, x_3) the coordinates of a point $x \in \mathbb{R}^3$ in this frame). The operating conditions are defined in terms of the total pressure ratio $P_t/P_\infty = 1.7$ and total temperature ratio $T_t/T_\infty = 1.15$, where the $_t$ and $_\infty$ subscripts refer to the stagnation and free-stream states, respectively. The jet is assumed to be isothermal ($T_{\text{jet}}/T_\infty = 1.0$), the jet Mach number is $M_{\text{jet}} = U_{\text{jet}}/c_{\text{jet}} = 0.9$, and the Reynolds number equals $\text{Re}_{d_N} = \rho_{\text{jet}} U_{\text{jet}} d_N/\mu_{\text{jet}} \simeq 1 \cdot 10^6$, where U_{jet} is the mean jet exit streamwise velocity, $d_N = 0.05m$ is the exit diameter of the nozzle, c_{jet} is the speed of sound, ρ_{jet} is the density and μ_{jet} is the dynamic viscosity.

As we have mentioned, five mesh adaptation steps based on steady RANS simulations are performed in the present study, and a final unsteady hybrid RANS/LES simulation is performed on the finest RANS adapted mesh.

The initial mesh is generated with the pre-processing software `ANSA` [2]: the geometry of the body and the far field boundaries are created and meshed with surface triangles; then, the prismatic boundary layer surrounding the surface of the body (the internal and external walls of the nozzle) is obtained thanks to a normal extrapolation of these surface triangles. The remaining volume of the computational domain is eventually filled with tetrahedra.

Not only the boundaries of the computational domain and the prismatic elements of the boundary layer mesh, but also the tetrahedral elements which are internal to the nozzle, are fixed throughout the computation, and are not subject to remeshing. Indeed, tetrahedra free to change size inside the nozzle but constrained by the fixed size of the surface could severely deteriorate the quality of the mesh.

The initial mesh, containing around 1.5 million degrees of freedom is represented on Figure 3.1, as well as the final mesh, resulting from the five-step adaptation process, containing around 10 millions of degrees of freedom. Remarkably, the mesh adaptation algorithm is capable of detecting the flow regions of interest, leading to a concentration of the elements around the potential core, and in the shear layer of the jet.

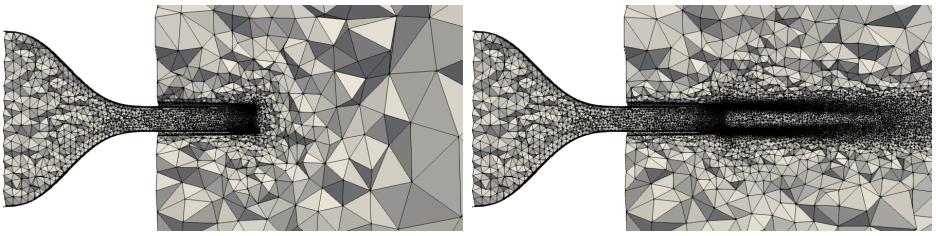

FIGURE 3.1. Illustration of the PPRIME nozzle configuration studied in Section 3.3; (left) zoom of the initial mesh; (right) computational mesh after 5 steady RANS adaptation steps in the jet zone.

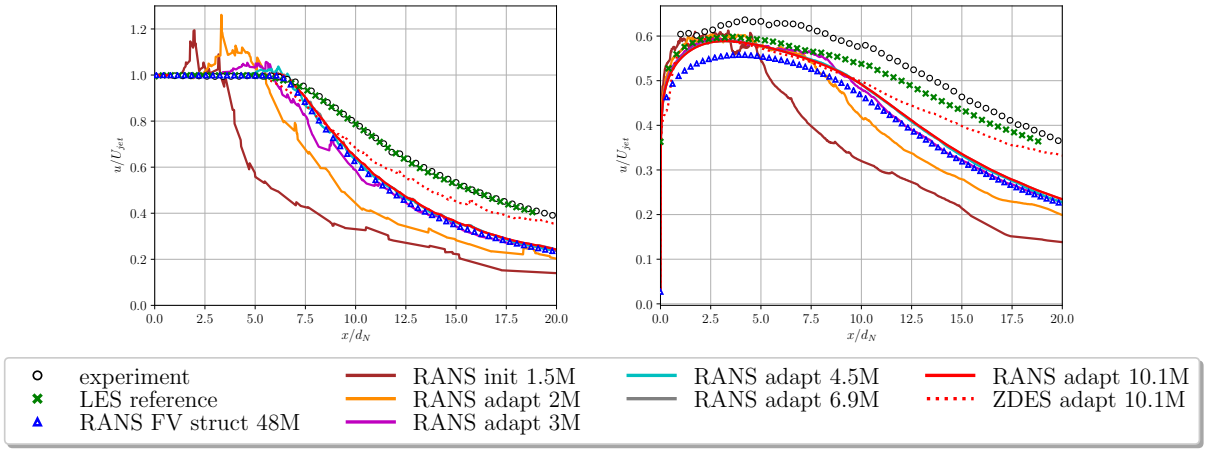

FIGURE 3.2. Velocity profiles (left) on the centerline and (right) on the lipline for the PPRIME nozzle example of Section 3.3.

The streamwise velocity profiles for four stations of the jet issued from the nozzle and the mean streamwise velocity profiles along the centerline are compared to the experimental and LES results obtained in [22]. The spurious numerical peaks in the velocity profiles, particularly visible for the coarsest simulations, originate from the discontinuous nature of the DG spatial discretization, for which in the proximity of interfaces of coarse elements, the polynomial approximating the solution can display sharp shapes. In order to assess our DG h-adaptive results obtained with a RANS model, we use for reference simulation a RANS second-order FV computation on a hexahedral structured mesh counting 48 millions of elements.

The meshes resulting from the third (4.5 millions of degrees of freedom), the fourth (6.9 millions of degrees of freedom) and the fifth adaptation steps (10.1 millions of degrees of freedom) are very similar, indicating that mesh convergence for the RANS case has been attained.

The results reported on Figure 3.2 reveal that the RANS computations, conducted with either the adaptive DG and structured FV methods, show a reasonable agreement with the LES results inside the potential core of the jet and in its vicinity, but that they tend to underestimate the centerline velocity of the jet for $x_1/d_N > 6$, leading to an earlier dissipation of the streamwise axial velocity. This underestimation is a typical behavior of RANS models applied to jet flows, see [1]. Nevertheless, the results of our adaptive simulation show a similar overall behavior as those of the FV reference computation, achieving closer results to the LES and experimental computations, with around 20% the number of degrees of freedom of the structured FV simulation (10.1 vs 48 millions). Moreover, the use of an adaptive process circumvents the difficulties posed by a classical, manual structured meshing process, especially when complex geometries are considered.

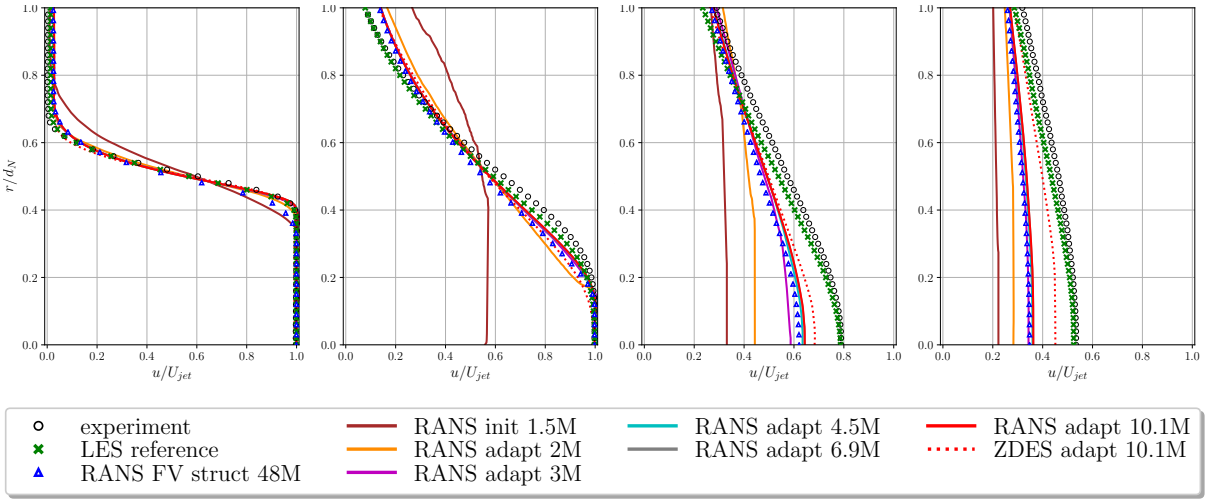

FIGURE 3.3. (From left to right) Velocity profiles on the axis, at $x_1/d_N = 1$, $x_1/d_N = 5$ and $x_1/d_N = 10$ in the PPRIME nozzle example of Section 3.3.

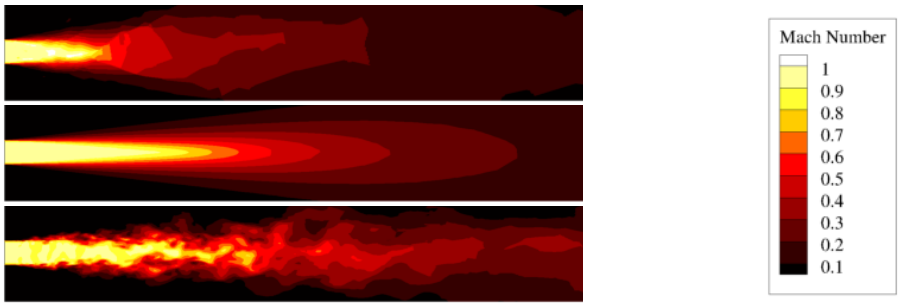

FIGURE 3.4. (Top) Contour of the reconstructed Mach field in the PPRIME nozzle example of Section 3.3 for the initial mesh, containing 1.5 million degrees of freedom; (middle) corresponding contour obtained with the final adapted mesh, containing 10.1 million degrees of freedom; (bottom) snapshot of the instantaneous Mach number of the 10.1 million degrees of freedom mesh using unsteady equations.

In Figure 3.3, we report radial velocity profile cuts at four locations in the mesh. At $x_1/d_N = 1$ the adaptive simulations match almost perfectly the LES and experimental results, and show a large increase in accuracy with respect to the fine FV structured simulation. At $x_1/d_N = 5$, our adaptive simulations are still very close to the LES and experimental results, but show a slight over-prediction of the spreading rate of the jet, while at $x_1/d_N = 10$ and $x_1/d_N = 15$ the radial velocity profiles of RANS simulations show significant discrepancies with respect to the experimental and LES reference results, but still perform slightly better than FV computations.

Eventually, we investigate the improvement of the jet flow prediction entailed by the use of a hybrid RANS/LES approach on RANS-adapted hybrid prismatic/tetrahedral meshes. The formulation of ZDES used in this work forces the whole interior part of the nozzle to act in RANS mode, while the DES equations are solved in the rest of the domain, see [95].

The benefits that the unsteady simulation can bring to the finest mesh solution are shown with red dotted lines in Figures 3.2 and 3.3. The velocity field of the unsteady simulation is averaged over a period of 150 times the characteristic time of the flow (d_N/U_{jet}). Despite being a relatively coarse mesh with respect to the standards of unsteady turbulent computations, the velocity profiles match more closely the experimental and LES references, thanks to the lower

degree of modeling of turbulence that hybrid RANS/LES and classic LES models introduce with respect to RANS. In Figure 3.4 we represent the steady Mach RANS solution on the initial mesh and on the finest mesh, and an instantaneous snapshot of the Mach field on the finest mesh performing an unsteady hybrid RANS/LES simulation. In the latter, a lower dissipation of the jet velocity is observed.

To conclude, the present example demonstrates the improvement allowed by the proposed unstructured metric-based RANS adaptation procedure on the flow field solution over simulations relying on classical FV methods, as well as the suitability of RANS-adapted meshes for unsteady computations. In the meantime, the resolution of the unsteady adaptive process once a good RANS solution has been obtained is considered essential in order to capture the unsteady features of the flow. The extension of the present work to unstructured *hp*-adaptation for unsteady turbulence models will be addressed in future research, taking advantage of the good dissipative and dispersive properties of high-order schemes, which are better suited for turbulent unsteady simulations.

4. An application of implicit domain meshing in shape optimization

This section illustrates how a coupling between the implicit domain remeshing feature introduced in Section 2.5 and the level set framework allows to account for dramatic mesh evolution, in the context of shape optimization.

4.1. A few generalities about shape and topology optimization

Shape and topology optimization is about finding the "best" shape with respect to mechanical, geometrical and manufacturing specifications. Over the last decades, this discipline has been raising an increasing enthusiasm within the academic and industrial communities, as the dramatic increase in the cost of raw materials calls for the optimization of the shape of mechanical parts since the early stages of design; we refer to e.g. [6, 7, 8, 16] for further motivations and comprehensive introductions to this field.

A typical shape optimization problem is of the form:

$$\min_{\Omega \subset D} J(\Omega) \text{ s.t. } C(\Omega) \leq 0, \tag{4.1}$$

where

- Ω is the optimized shape, which is sought within the fixed computational domain D;
- $J(\Omega)$ is a given performance criterion;
- $C(\Omega)$ stands for a (collection of) constraint functional.

Usually, the functionals $J(\Omega)$ and $C(\Omega)$ depend on the shape Ω via its mechanical behavior, that is, mathematically, via a *state* u_Ω, solution to a "physical" partial differential equation posed on Ω (for instance, the heat equation, the elasticity system, etc.).

The treatment of shape optimization problems of the form (4.1) by constrained optimization algorithms requires a notion of derivative with respect to the domain. Here, we rely on Hadamard's boundary variation method [9, 62, 76, 94]; when combined with the so-called adjoint technique from optimal control theory, it allows to calculate a "shape gradient", that is, a descent direction, for a function $F(\Omega)$ of the domain. The latter arises as a vector field $\theta : \mathbb{R}^3 \to \mathbb{R}^3$ such that, for small $t > 0$:

$$F((\mathrm{Id} + t\theta)(\Omega)) < F(\Omega);$$

intuitively, a "small" displacement of the boundary $\partial\Omega$ in the direction pointed by the vector field θ results in a new domain $(\mathrm{Id} + t\theta)(\Omega)$ with a slightly lower value of $F(\Omega)$.

4.2. **Body-fitted shape and topology optimization**

Usually, large modifications of the shape are expected in the course of the resolution of a shape and topology optimization problem of the form (4.1), making the use of Lagrangian strategies such as those described in Section 2.4 unrealistic in this context.

Multiple alternatives have been thought of in order to circumvent this problem, and notably the resort to the level set method outlined in Section 2.5, see [8] for the seminal contribution in shape optimization.

Recently, this framework has been successfully combined with remeshing techniques, resulting in a body-fitted shape optimization method, which features an exact mesh of the shape Ω (and of the complement $D \setminus \overline{\Omega}$), see [5, 37, 50, 51]. In a nutshell, two complementary representations of a shape $\Omega \subset D$ are available at each stage of the process:

- *Meshed representation:* On the one hand, Ω (and thus $D \setminus \overline{\Omega}$) is explicitly discretized in the mesh $\mathcal{T}$ of D: $\mathcal{T}$ is a valid, conforming mesh of the total computational domain D, which can be divided into two submeshes $\mathcal{T}_\Omega$ and $\mathcal{T}_{D\setminus\overline{\Omega}}$ of Ω and $D \setminus \overline{\Omega}$, respectively.

- *Level set representation:* On the other hand, Ω is described as the negative subdomain of a level set function $\phi : D \to \mathbb{R}$ defined at the vertices of $\mathcal{T}$, see (2.1).

Dedicated numerical algorithms are then used to switch from one representation to the other. When a meshed representation of Ω is available, a corresponding level set function $\phi : D \to \mathbb{R}$ is calculated as the signed distance function to Ω at the vertices of the mesh $\mathcal{T}$, for instance by using the fast marching method, see [90]. Conversely, when Ω is described by a level set function $\phi : D \to \mathbb{R}$ on a mesh $\mathcal{T}$ of D, a new mesh $\widetilde{\mathcal{T}}$ of D where Ω is explicitly discretized can be obtained thanks to the methodology described in Section 2.5.

In the course of the numerical resolution of a shape optimization problem of the form (4.1), each representation of the shape Ω is used depending on the performed operation, see Figure 4.1:

- The finite element analyses needed to calculate the state u_Ω, and thereby a descent direction θ for the problem (4.1) can be conducted from the meshed representation of Ω.

- The motion of Ω to the next iterate $(\mathrm{Id} + t\theta)(\Omega)$ can be realized with the level set representation ϕ of Ω, by solving the evolution equation (2.2) on the mesh $\mathcal{T}$.

This approach has the following benefits:

- Accurate mechanical calculations can be realized on the exact mesh $\mathcal{T}_\Omega$ of the shape Ω, via the finite element method for instance. This mesh can be readily exported, and finite element analyses can be carried out by using an external software application in a black-box fashion; hence, this strategy totally decouples the update of Ω from the mechanical analyses needed to evaluate its performance and the sensitivities of the optimization functionals.

- Since the structural interface $\partial\Omega$ is explicitly discretized at each step of the iterative procedure, this body-fitted approach simplifies the evaluation of geometric and mechanical quantities of interest near $\partial\Omega$ (such as the perimeter, or the curvature of $\partial\Omega$, or the normal stresses applied on $\partial\Omega$).

- Since the level set method is used to realize the motion of the shape, dramatic evolutions of shapes can be accounted for, including topological changes.

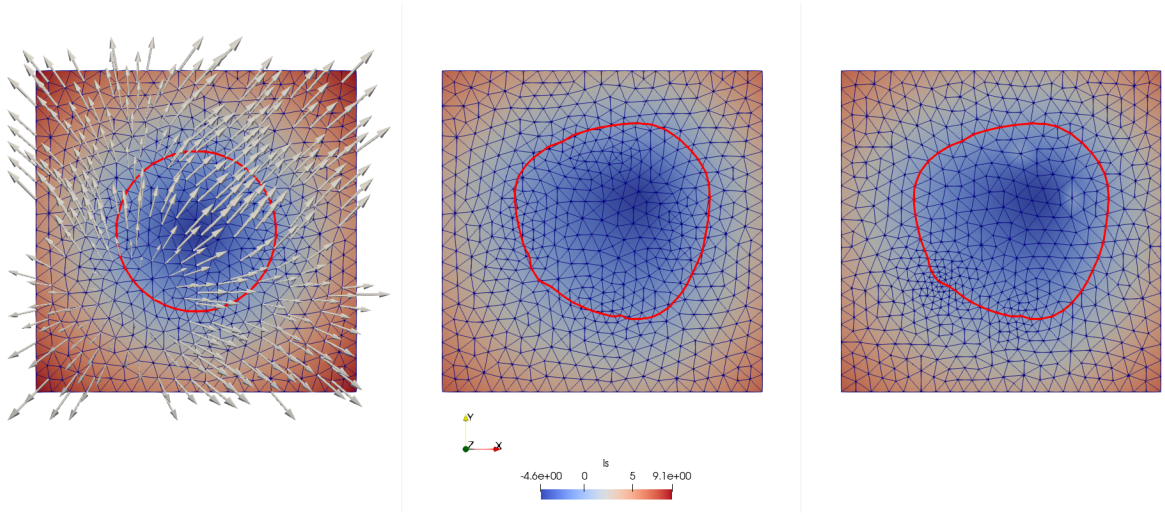

Figure 4.1. Level set advection and body-fitted remeshing for interface tracking. (Left) initial level set defined at the vertices of a large conformal mesh and associated velocity field; (middle) level set update using the Hamilton–Jacobi advection equation; (right) domain remeshed to fit the zero isovalue of the advected level set function.

4.3. **Numerical example: optimization of an elastic mechanical system**

We illustrate the proposed approach with the optimization of the design of an elastic two-component mechanical system. This numerical example was treated by using `PISCO`, a Research and Development software platform devoted to topology optimization that is under active development at `IRT SystemX` and `Safran Tech`. The industrial-grade solver `Code_Aster`[1] is used to perform the finite element analyses needed to evaluate the physical criteria and the corresponding sensitivities.

The considered setting is depicted on Figure 4.2. The computational domain D is composed of two regions: an L-shaped support corresponding to a non optimizable region, which supports the boundary conditions of the physical analysis, and a structural block, which is the optimized domain, properly speaking. These are filled with two different linearly elastic materials (the support being characterized by a higher Young's modulus than the block) and they are linked by rigid mechanical connections, represented on Figure 4.2 (b). Both regions are discretized using linear tetrahedral elements while the rigid connections are accounted for by four discrete 1-order elements. Dirichlet and linear sliding contact boundary conditions are prescribed on two regions of the boundary ∂D located respectively on the support and the block boundaries. Eventually, two load cases are considered, which are applied on another region of ∂D see Figure 4.2 (c,d).

The goal of the present experiment is to minimize the volume of the structural block (so as to design a lighter structure) under four constraints: one on the compliance of the total structure Ω and one on the Von Mises stress field for each of the two considered load cases. The Von Mises constraints are formulated using a p-norm aggregation technique which is quite classical in stress-constrained topology optimization, see e.g. [48].

We rely on the body-fitted shape and topology optimization strategy introduced in Section 4.2 to solve this problem, starting with the full structural block of Figure 4.2 (a) as initial shape. The optimized design is depicted in Figure 4.3, and the associated displacement and Von Mises stress fields for both considered load cases are reported in Figure 4.4. At convergence, all the constraints are fulfilled and the weight of the assembly has been reduced by 52%.

[1] `https://www.code-aster.org`

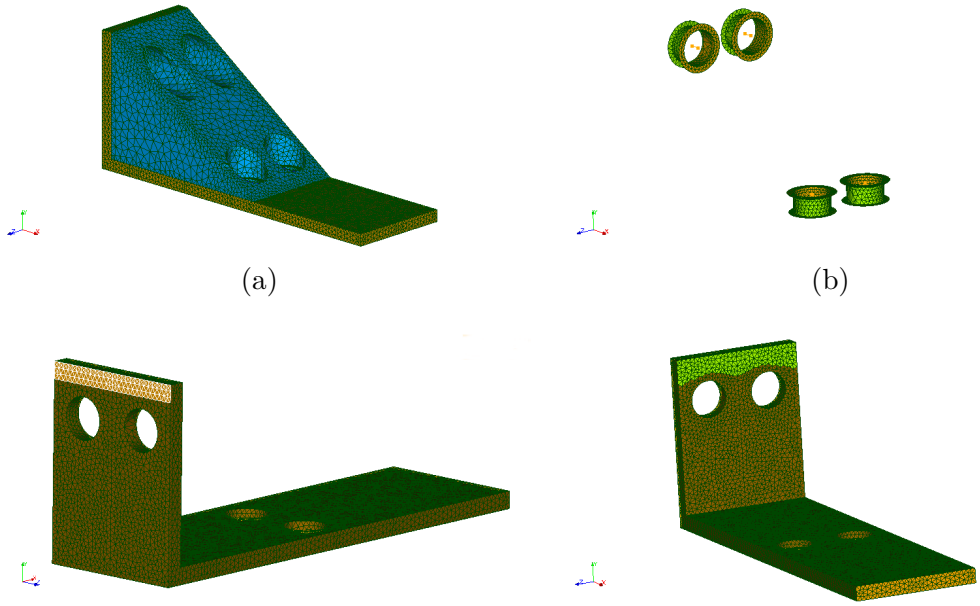

Figure 4.2. (a) The two-component mechanical system test case considered in Section 4.3; the non optimizable L-shaped support region is represented in orange and the optimized block is in blue; (b) rigid connections linking both regions; (c) clamped region of the device; (d) the surface supporting contact boundary conditions is represented in green, and that submitted to loads is in light orange.

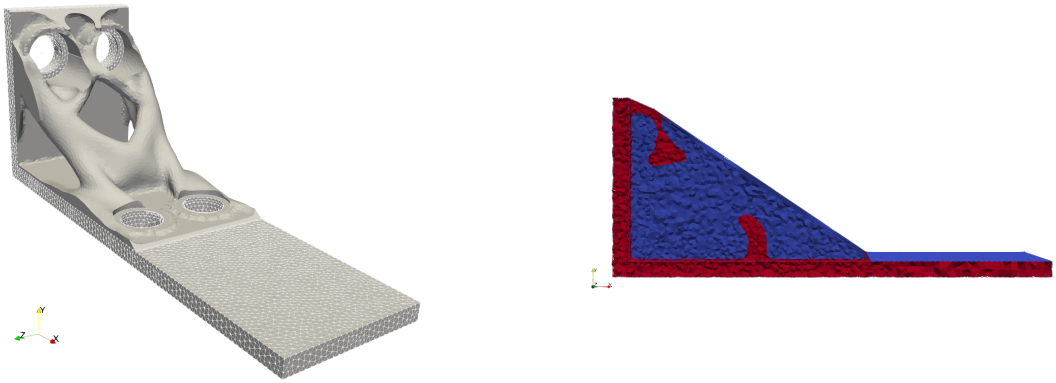

Figure 4.3. (Left) optimized shape of the structural block in the example of Section 4.3; (right) cut in the mesh of the computational domain D, divided into material and void parts, in red and blue, respectively.

Note that a full remeshing of the structural system is carried out at each step of the optimization process. As we have already emphasized, the employed body-fitted approach simplifies greatly the definition and the evaluation of the physical criteria and sensitivities, since any finite element solver can be used in a non intrusive fashion. On the other hand, the remeshing stages involved in this strategy have admittedly a huge impact over the total computational cost of the process. Thus, topology optimization would greatly benefit from the development of dedicated parallel remeshing routines to target large-scale structural optimization challenges.

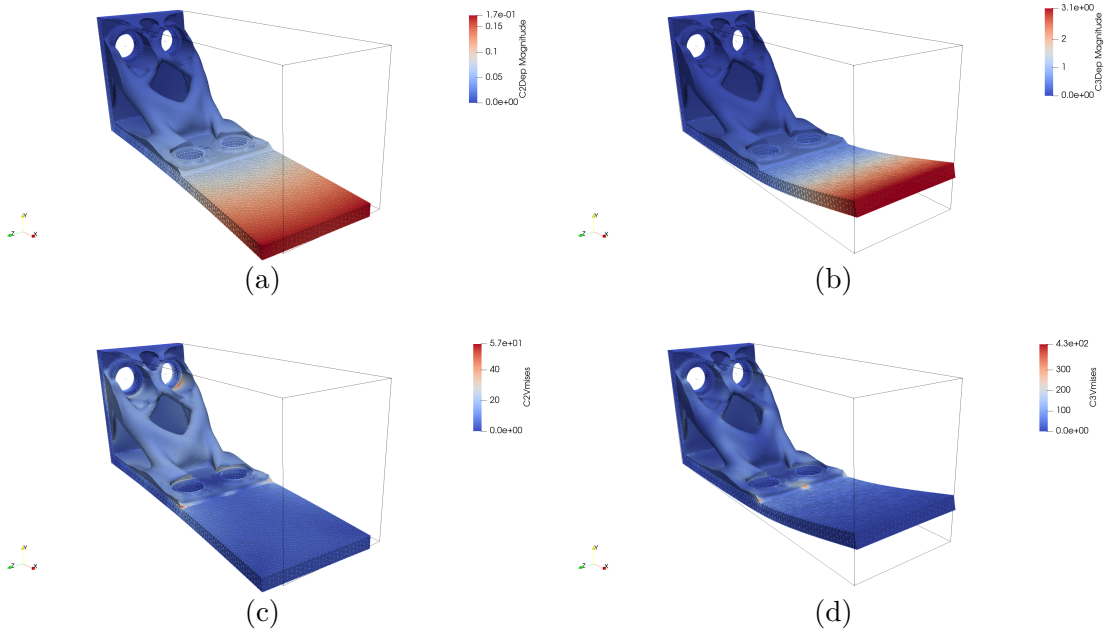

Figure 4.4. (a) Displacement field for the first load case; (b) Von Mises stress for the first load case; (c) displacement field for the second load case; (d) Von Mises stress for the second load case.

5. Conforming remeshing using level set discretization for geophysical inverse problem

5.1. Context and motivation

This section illustrates the contributions of local remeshing algorithms in geophysical modeling, and more particularly in the management and the reduction of uncertainty over the identification of geological interfaces.

The underground properties of the earth govern many natural or anthropic phenomena occurring over multiple time scales, such as mineral element migration and concentration processes, earthquake initiation and propagation, landslides, groundwater flow and contaminant transport, hydrocarbon recovery, plume formation during CO_2 sequestration, geothermal heat recovery, etc. In particular, the representation of geological interfaces is a crucial aspect of the numerical simulation of these processes, as these surfaces delimit regions containing different geological materials (rock units) characterized by specific ranges of mineralogical and physical properties. For example, in sedimentary basins, these rock units arise as layers separated by horizons and can be described at multiple nested scales, as exemplified on Figure 5.1. The geometry of these layers may contain challenging geometric features such as sharp creases or low-angle contacts between surfaces, for instance when geological layers vanish laterally because some previously deposited material has been eroded (see Figure 5.1 (c)). Under the effect of tectonic forces and subsurface fluids, cracks naturally present in the rock may grow as fractures, and eventually become faults under the accumulation of tangential displacement, which results in possibly complex juxtapositions of materials on either side of the fault, see e.g. Figure 5.1 (d).

From the numerical point of view, the simulation of subsurface phenomena such as those mentioned above relies on a model describing the features of the underground medium (and notably the geophysical interfaces) at the scale of interest, which is then meshed and populated with physical rock properties before the physical equations of interest are solved.

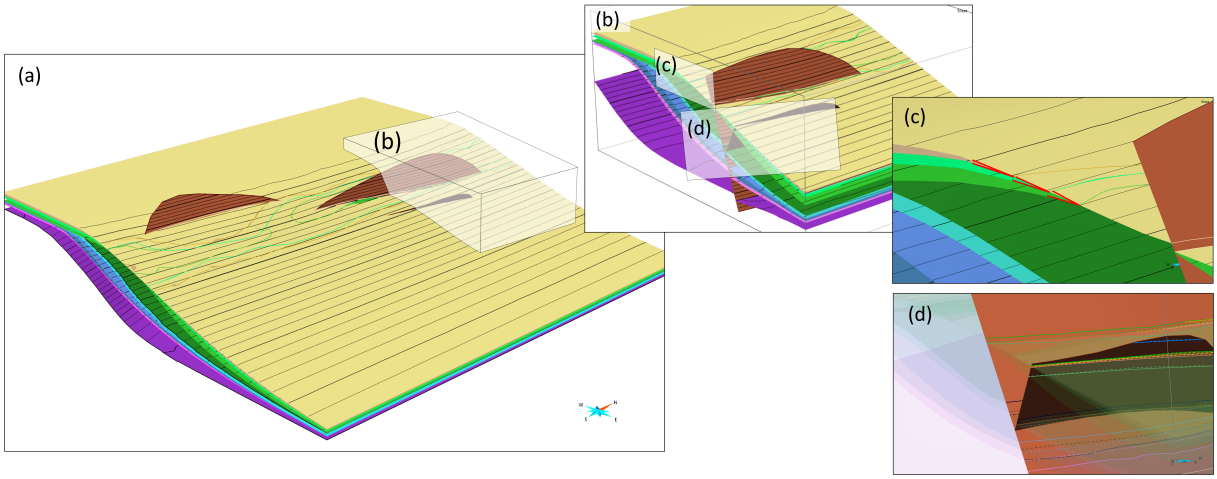

Figure 5.1. Example of a challenging geometrical configuration in a 3D geological model, as discussed in Section 5. (a) Global view of a faulted stratigraphic model; dark lines account for depth contours and colored lines in the upper horizon represent stratigraphic unconformities; (b) cut-off view; (c) the low angle contacts lines at the stratigraphic unconformities caused by erosion are displayed in red; (d) view of how horizons (in transparency) are offset by faults, forming small geometric features and low-angle intersections on cut-off lines on either sides of each fault.

The aforementioned complex underground properties of the earth are therefore pivotal in these computations. Unfortunately, they are generally inaccessible to direct observation. Rather, they must be inferred from indirect data, such as surface measurements, borehole sample analyses or in situ geophysical measurements and geophysical images. This is a highly challenging task, since such data are sparse, ambiguous and insufficient to precisely characterize the domain. Hence, geological knowledge is a key ingredient of the interpretation process and significant uncertainty exists about the existence, location, connectivity and geometry of interfaces between different rock units [101].

Numerical approaches based on the level set method (see Section 2.5) have raised a significant interest within the geoscience modeling community, as a convenient tool to cope with these uncertainties. One category of methods, referred to as "implicit structural modeling", consists in creating level set functions from typical interpretation points. In tetrahedral formulations [24, 54], this paves the way to the generation of meshes where the discontinuities under scrutiny are explicitly discretized, as described in Section 2.5, which allows to conveniently manage faults and other geological interfaces.

Other investigations have used perturbations of level set functions for generating alternative geological models representing uncertainties [33, 103]. In the perspective of reducing the related uncertainties, several deterministic or stochastic inverse methods have been formulated to take into account indirect geophysical data [56, 63, 83], but most of them feature one single level set function during the inversion process. Going further calls for inverting several and possibly finite open interfaces while preserving their geological consistency [60, 101]. Moreover, even though some Monte Carlo methods are available to address the uncertainty in the geometric configuration [18, 28], their dissemination has been hampered by the difficulty to robustly automate mesh generation for arbitrary configurations.

Using the level set method in combination with body-fitted tetrahedral meshes in the spirit of Section 2.5 is a very attractive strategy to handle these requirements: it indeed combines

geometric adaptivity, accuracy and versatility to several PDE discretization schemes such as the finite element method or the control volume finite element method.

5.2. Two-dimensional numerical example: sensitivity of a two-phase Darcy flow with respect to the existence and the geometry of fractures

In this section, we consider the 2D porous fractured network configuration depicted on Figure 5.2: we assume 6 fracture intersections along two distinct observation lines have been located. As often in geoscience practice, the performed observations do not contain the same amount of information, neither are they equally reliable: some of these data provide the fracture position and orientation while others provide at best an approximate insight into its position. From these incomplete observations, spatially exhaustive scenarii are generated, representing possible fracture configurations compatible with the observations. In the present 2D case, this task is easily achieved by hand, but many sampling methods exist to generate this type of model, based on statistical and/or physical reasoning, see for instance [19, 29, 38, 61, 102].

In each situation, the initial porous region is assumed to be filled with oil. We then simulate the injection of water in the underground, from the lower left corner of the computational domain Ω; both fluids are recovered in the upper right corner of Ω. Assuming the fluids to be incompressible and immiscible, the result of this experiment is predicted by the resolution of the equation for the pressure P:

$$\nabla \cdot \left(\left(\sum_{\alpha} \frac{k \cdot k_{r\alpha}}{\mu_\alpha} \right) \nabla P \right) = \sum_{\alpha} q_\alpha,$$

which is complemented with the saturation equation for each phase α

$$\varphi \frac{\partial (S_\alpha)}{\partial t} - \nabla \cdot \left(f_\alpha \frac{k \cdot k_{r\alpha}}{\mu_\alpha} \nabla P \right) = q_\alpha.$$

Here, φ stands for the rock porosity; k is the rock permeability, S_α, μ_α, $k_{r\alpha}$ and f_α are respectively the saturation, viscosity, relative permeability and fractional velocity of the phase α, and q_α is the volumetric source term. The numerical simulation solver is based on a control-volume finite element approach, where fractures are represented as lower-dimensional mesh elements, see [65, 73]. In this experiment, we assume for simplicity that the relative water permeability is given by the square of the water saturation. The intact rock has a porosity of 0.2. Fractures are assumed to be open (porosity of 1). The matrix and fracture permeabilities are taken equal to 10^{-15} m^2 and 10^{-9} m^2, respectively. The viscosity for oil and water are 0.45 and 1 mPa.s, respectively.

Each fracture is parametrized by its center, size and azimuth. Starting from an initial mesh of the computational domain Ω, the various fractures at stake are inserted one after another by relying on the implicit surface meshing technology described in Section 2.5: briefly, the signed distance field ϕ to the fracture is computed by a least-square optimization method [54]. A subregion of the mesh of Ω is then defined from the fracture center and size to bound the editing. Finally, the methodology of Section 2.5 is used to insert the zero level set of ϕ into the portion of the mesh corresponding to this region.

This strategy presents numerous assets, when it comes to exploring multiple possible fracture geometries and to appraising their impact on the flow. In particular, this iterative approach allows to locally perturb an existing scenario (via a perturbation on the corresponding level set function) in order to explore uncertainties over its geometry (such as the fracture shape or position), while ensuring that the mesh features adequate element size and quality.

On the contrary, the opposite strategy consisting in meshing the fracture network at once is highly challenging, since its very thin features inevitably lead to meshes containing a large number of elements, or ill-shaped mesh elements, which either slows down simulations or affect their stability. Admittedly, several simplification approaches have been proposed in this context,

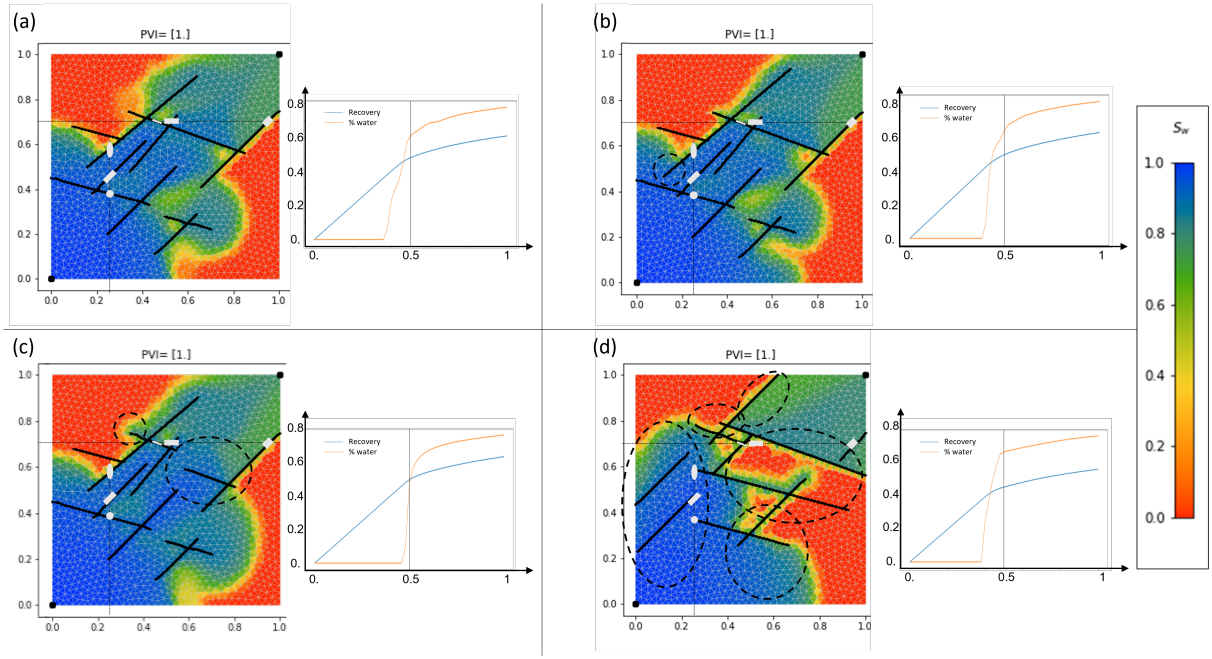

FIGURE 5.2. Several fracture network scenarii in the numerical example of Section 5.2, together with the associated multiphase porous medium behaviors. The spatial fracture observations and the associated uncertainties are displayed as grey rectangles and ellipses along thin dotted lines. The fractures are bold dark lines, which are explicitly discretized in the computational mesh. The color scale shows the water saturation. Each graph shows the water fraction and produced volume at the upper right corner, after injection of one pore volume (PVI) in the lower left corner. The dashed ellipses highlight the changes between (a) and each alternative scenario.

with the aim to approximate the target fracture geometry while preserving the mesh quality, while retaining an affordable amount of elements, see [10, 53, 64, 77].

This simplification problem is significantly easier in the implicit framework where each fracture is sequentially inserted as the 0 isosurface of the signed distance field ϕ. Indeed, low quality mesh elements can be prevented by setting the value of ϕ to 0 for the mesh nodes which are "too close" to the inserted isosurface. To minimize the perturbation of the fracture shape, this form of snap rounding is only applied to nodes located on other boundaries or at the edge of the subregion of Ω being edited, as later remeshing (Section 2.5) takes care of mesh quality issues elsewhere.

In order to illustrate the proposed workflow, we consider the example depicted on Figure 5.2: the initial scenario depicted on Figure 5.2 (a) includes 6 and 4 fractures belonging to two fracture sets of azimuth ca. 45° and 105°, respectively. In the configuration of Figure 5.2 (b), only one fracture has been extended with respect to that in (a), leaving the rest of the model unchanged. This minor perturbation has a limited effect on the production curves, and induces only a moderate change in the saturation field. The scenario of Figure 5.2 (c) involves a more drastic change with respect to (a): one fracture is replaced by two fractures. This type of operation could occur for instance in a reversible jump Monte Carlo Markov Chain transition [18]. As this change affects the fracture connectivity, the impact on the production curves and on the saturation field is significant: the water breakthrough at the production well occurs later, meaning that this fracture configuration yields a better sweeping efficiency of the injected water. The scenario of Figure 5.2 (d) represents what could occur after multiple transitions from the initial model (a): although the orientations and sizes are comparable, most fractures have been moved, removed

or replaced. Interestingly, the production curves in the scenario (d) are similar to those of (a) and (b). This illustrates the ill-posedness of this particular type of problems. This motivates the use of stochastic inverse methods rather than deterministic optimization to address subsurface uncertainty.

6. Dynamic and parallel mesh adaptation for premixed flame fronts and liquid/gas interfaces

6.1. Dynamic mesh adaptation for material interfaces

6.1.1. *Motivation*

The numerical simulation of dynamic material surfaces in flows such as turbulent flame fronts and liquid/gas interfaces has long been a thorny issue. One reason is that the reaction or interface forces occur in very thin and moving layers around the surface, so that an accurate capture of the latter requires a very fine mesh resolution in its vicinity. On the other hand, such interfaces typically undergo dramatic displacements, spanning the whole computational domain, which makes it impossible to adapt the mesh once for the whole simulation. Dynamic mesh adaptation is a more realistic alternative to this unfeasible, brute-force approach. It consists in modifying the mesh regularly so that it features a fine mesh size in the vicinity of the interface and a coarser one away from it. Such a strategy is usually hindered by two factors:

(1) The huge increase in CPU cost entailed by the mesh adaptation process, which cancels out the potential gain allowed by applying the numerical solver with a reduced number of elements.

(2) Even though the cost of mesh adaptation is affordable, another challenge comes from the compromise between the thickness of the refined layer around the interface (and thereby, the size of the mesh) and the frequency at which remeshing is conducted.

This section presents solutions to reduce the overhead of dynamic mesh adaptation by relying on a parallel implementation, and to find a compromise between the thickness of the refined layer and the remeshing frequency. From the algorithmic viewpoint, we use the `YALES2` flow solver [74] for the numerical simulation of the physical problems at stake. This solver also implements a version of the iterative remeshing-repartitioning algorithm relying on `mmg` to perform parallel mesh adaptation.

6.1.2. *A two-level parallel mesh adaptation method*

As was presented in Section 2.6, the efficiency of parallel remeshing strategies carried out on distributed architectures (in terms of the quality of the resulting mesh, notably) crucially depends on the ability of the procedure to modify the whole mesh $\mathcal{T}$. More precisely, in each submesh $\mathcal{T}_k$ (processed on rank k), not only the regions lying "far" from the interface $\mathcal{O}$ between ranks, but also those lying close to $\mathcal{O}$ have to be remeshed. While the modifications of the former regions can be done in parallel by independent calls to the mesh adaptation library, the treatment of the interface between ranks is much more delicate. As we have outlined in Section 2.6, several strategies are available to tackle this issue, and that retained in the present work is the so-called moving interface method. In a first step, the mesh is refined inside each rank, with respect to a metric whose definition is detailed in Section 6.1.3, while leaving the regions near $\mathcal{O}$ untouched. During this step, the target metric is interpolated from the old to the new mesh. This adaptation step is therefore local and does not require any communication between the ranks. In a second step, low-quality elements are sent from one rank to another. The scheduling of this movement is driven by a constrained load balancing algorithm. This operation has to ensure that the grid

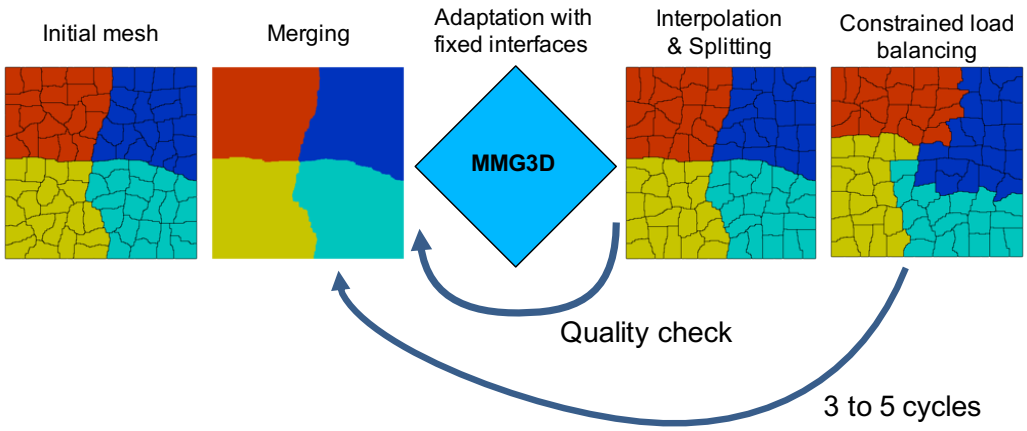

Figure 6.1. Schematic of the two-level moving interface parallel adaptation strategy developed in Section 6. The thin black lines represent the interfaces between cell groups and the different colors correspond to as many ranks.

remains properly balanced from one rank to another, i.e. that all ranks have approximately the same number of tetrahedra, and it has to enforce the movement of the interface. To achieve this, one solution is to impose strong weights at the connections (or arcs) between the elements at the interface. Thus, when minimizing the edge cut of the connectivity graph of the mesh $\mathcal{T}$, the load balancing algorithm will be strongly enticed to place the new interface away from the previous interface so as to avoid cutting heavy-weight connections. Once the elements have been moved from one rank to another, the process can be repeated until convergence. This convergence is achieved when the difference between the target and resulting metrics is below a threshold value and when the element skewness does not exceed a limit value. The performances of this method depend on the efficiency of each of the aforementioned operations, namely:

(1) The sequential mesh adaptation library;

(2) The interpolation of the data stored on the grid;

(3) The load balancing algorithm;

(4) The cell and data transfer from one rank to another;

(5) The parallel connectivity reconstruction.

All these steps may limit the performance of the overall process when a large number of cores is involved. In order to alleviate all the above potential limitations (except those concerning the sequential mesh adaptation algorithm, which we assume to be sufficiently efficient in the following), we rely on a two-level domain decomposition method. The latter is based on a slight refinement of the general parallel remeshing principles exposed in Section 2.6, which is illustrated in Figure 6.1: in a nutshell, most of the above tasks are actually conducted at a coarser scale than that of the tetrahedra $T \in \mathcal{T}$, that of groups (hereafter referred to as cell groups) composed of several thousands of tetrahedra. For instance, the load balancing algorithm is applied to the connectivity graph of the cell groups instead of that of the cells, which allows for a significant gain in CPU time as the latter graph is much lighter and easier to partition. Likewise, the cell migration operation is performed cell group by cell group and it is combined to automatic packing/unpacking with non-blocking send/receive calls to speed-up the transfer. Finally, the interpolation of numerical quantities from the initial mesh to the modified one is done at each sub-step, benefiting from the 2-level domain decomposition to locate the nearest source

tetrahedron from a given destination element. The only drawback of this two-scale strategy is that it requires additional merging and splitting algorithms in order to provide a large block to the sequential remeshing software instead of small cell groups. However, this drawback does not alter the performances of the complete workflow, and the overall gain far compensates this overhead.

6.1.3. *Choice of the metric*

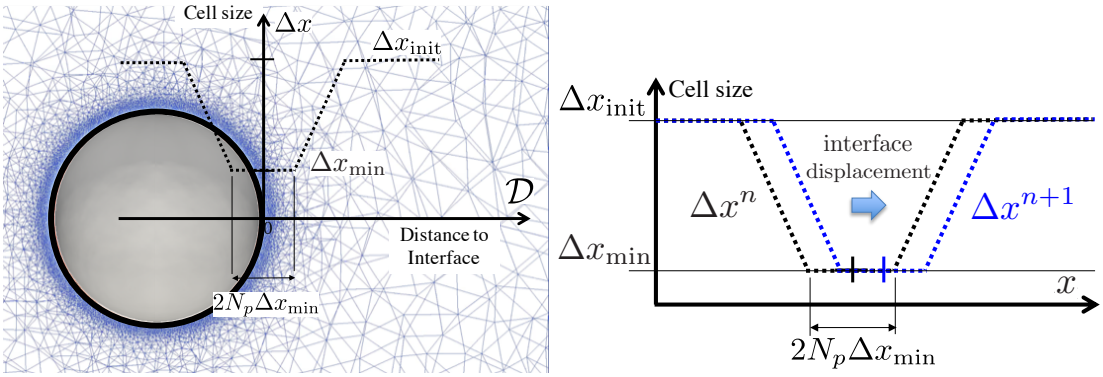

FIGURE 6.2. Illustration of the construction of the size map involved in the remeshing strategy of Section 6.1 (left) and its displacement (right).

The dynamic mesh adaptation of a moving material interface such as a flame or a liquid/gas interface requires frequent remeshing steps. When using low-dissipation and low-dispersion numerical schemes such as those used in Large-Eddy Simulation and Direct Numerical Simulation of turbulent flows, the remeshing steps have to introduce as small numerical perturbations as possible. The numerical perturbations entailed by the adaptation of a mesh $\mathcal{T}$ into a new mesh $\widetilde{\mathcal{T}}$ better suited to the further evolution of the interface mainly come from the interpolation errors of numerical quantities defined on $\mathcal{T}$ onto $\widetilde{\mathcal{T}}$, whose values near the interface are of critical importance. These perturbations can be made negligible if the remeshing occurs away from the interface, i.e. if the mesh metric is kept constant near the interface under scrutiny and if the tetrahedra nearby are frozen during this operation. Keeping the size prescription constant near the interface and controlling its variation away from it leads to define the metric as in Figure 6.2: it is set to Δx_{min} in a narrow band with thickness $2N_p\Delta x_{\text{min}}$ around the interface, and its gradient is bounded outside this region, so that the metric attains that of the initial coarse mesh "far" from the interface. With this definition and as illustrated on the left-hand side of Figure 6.2, when the interface moves of a distance less than $N_p\Delta x_{\text{min}}$, the size prescription changes where either the old or the new metric shows linear variations, while keeping a constant value at the interface location; this ensures an exact interpolation of the data within this protected zone. The remeshing effort is therefore concentrated in the region where the metric changes most, i.e. in the linear metric gradient zone.

6.2. **Numerical example: application to the lean-premixed PRECCINSTA burner**

The computational benefits of the proposed dynamic mesh adaptation strategy are first illustrated with the example of a semi-industrial lean-premixed burner, operating with a methane-air premixing under atmospheric conditions. This burner, called PRECCINSTA, has been widely used for model development and validations [15, 75]. In this burner, the turbulent flame is anchored to the injector thanks to a swirl motion, which creates large inner and outer recirculation zones. A Large-Eddy Simulation with the same models and numerics as in [15] is performed

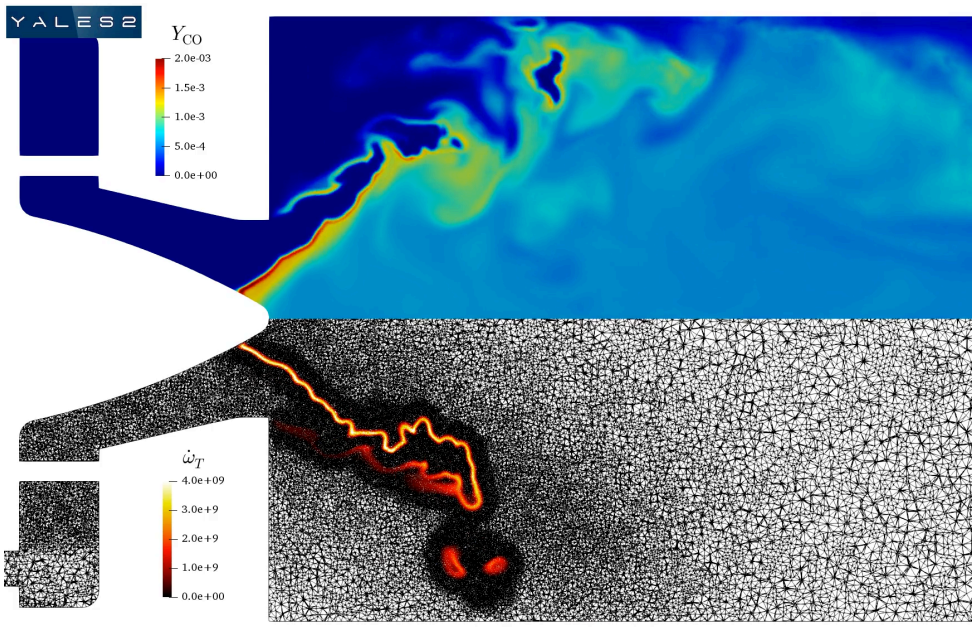

Figure 6.3. Dynamic mesh adaptation in the PRECCINSTA burner test case of Section 6.2: the instantaneous CO mass fraction and the heat release fields are displayed, together with a slide of the 3D tetrahedral mesh.

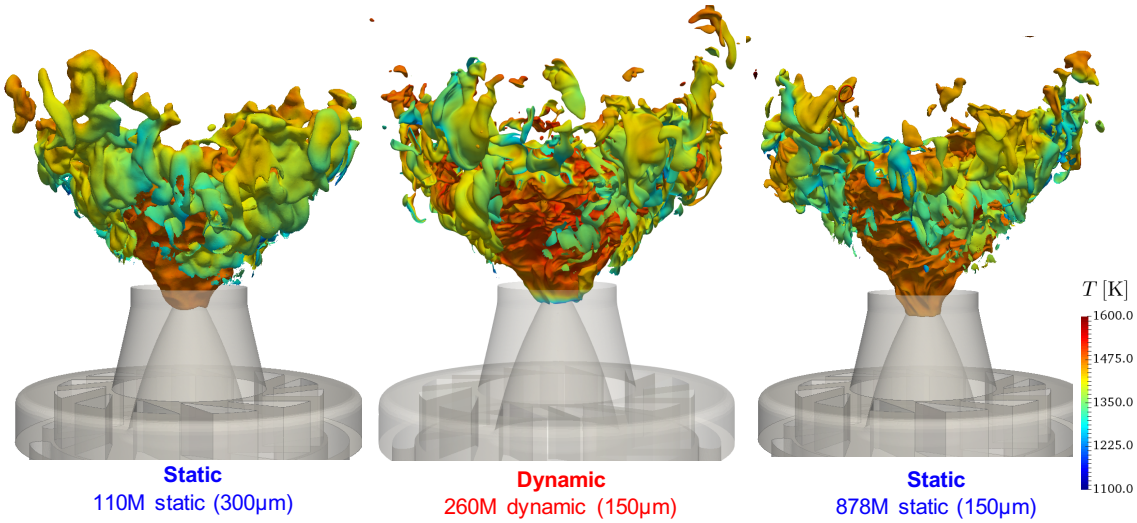

Figure 6.4. Comparison of the numerical simulations using static and dynamic meshes at several resolutions in the PRECCINSTA burner example of Section 6.2.

with the help of the dynamic mesh adaptation strategy presented in Section 6.1. The results are depicted in Figure 6.3; in there, a mesh resolution of 300 microns in the flame front and of 600 microns in the primary combustion zone are used. These coarse resolutions are chosen to better highlight the metric definition. As the flame is highly unsteady due to flame/turbulence interactions, the flame front displacement is tremendous and it eventually features topological changes, as isolated burning pockets are created. The remeshing frequency is therefore piloted by a Courant–Friedrichs–Lewy condition based on the displacement velocity, which ensures that

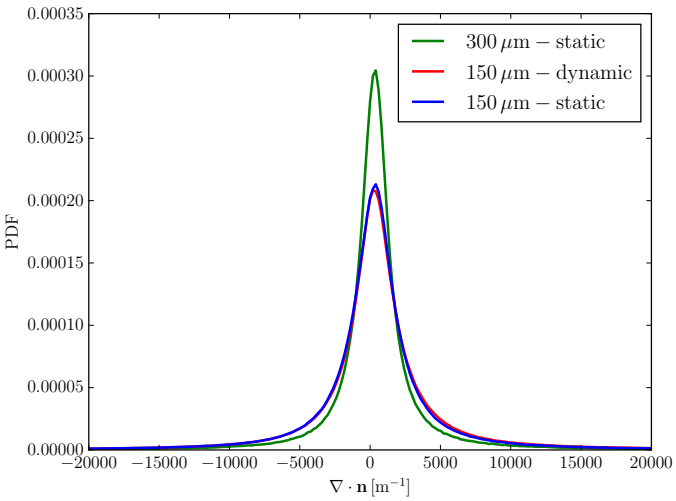

Figure 6.5. Comparison of flame front curvature distribution for static and dynamic meshes in the PRECCINSTA burner.

the flame front remains in the refined zone. The flame topology is rendered as a progress variable iso-surface colored by the temperature in Figure 6.4. In there, the left- and right-hand side subfigures correspond to two treatments of this simulation using static meshes (i.e. the computational mesh is adapted at the beginning of the process, and is then left untouched) containing 110 millions and 878 millions tetrahedra, respectively. In the center, the same situation is treated with our dynamic mesh strategy; the computational mesh is obtained with the background mesh of the 110 million cell mesh while the resolution at the flame front is the same as that of the 878 million cell mesh. The flame topology of the front obtained thanks to the dynamic strategy features small-scale wrinkles similar to the refined static case (878M) while the total CPU cost is around 4 times smaller. This observation is confirmed by the plot of the flame curvature distribution in Figure 6.5. The distributions of the dynamic and of the refined static cases are the same while the simulation based on the coarser static mesh exhibits smaller values of the curvature.

6.3. **Numerical example: application to the droplet impact on a cone**

In this section, the benefits of dynamic mesh adaptation are illustrated with the simulation of the impact of a water drop on a superhydrophobic cone characterized by its semi-angle 50° (a situation which has been studied experimentally in [26, 47]). It is an interesting case as the drop undergoes various topological changes after its impact on the cone: it subsequently takes the shape of an attached ring before bouncing and breaking into a number of smaller droplets. The modeling of the liquid/gas interface and that of the contact angle of 163° are detailed in [86]. The liquid/gas interface is captured with a conservative level set algorithm described in [41] and the contact angle is imposed through a curvature modification at the contact line. The metric definition follows the strategy exposed in Section 6.1.3, with a thickness of the refined zone tailored so that 30 cells are in the initial drop radius. The results and the mesh are presented in Figures 6.6 and 6.7. The experimental results are very well reproduced in terms of topology and of break-up dynamics. Such simulation would be intractable without dynamic mesh adaptation.

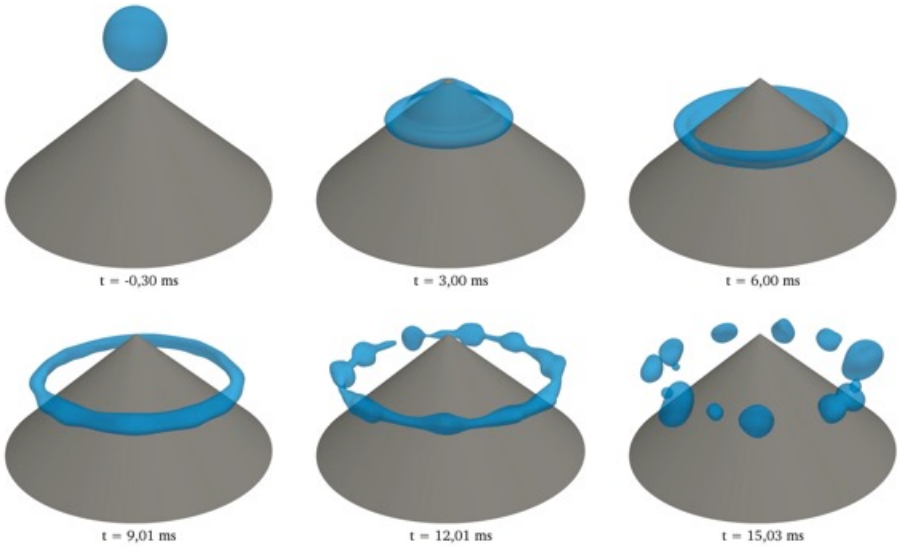

Figure 6.6. Impact of a water droplet on a superhydrophobic cone as considered in Section 6.3.

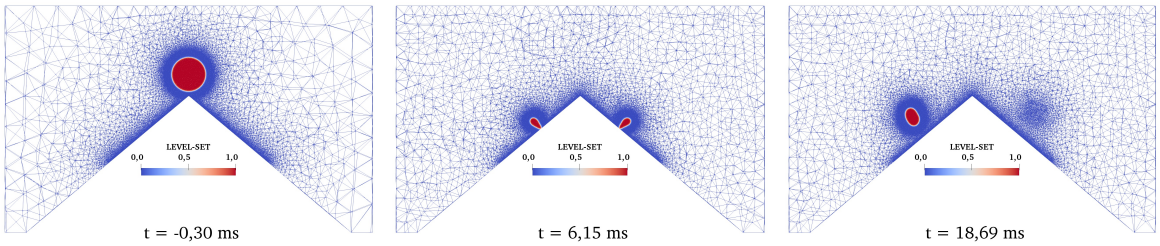

Figure 6.7. Dynamic mesh adaptation in the example of the impact of a water droplet on a superhydrophobic cone in Section 6.3.

Acknowledgements

F. Basile would like to thank Dr Fabio Naddei from ONERA for his precious advice about mesh adaptation, as well as Dr Fabien Gand from ONERA for providing the structured PPRIME mesh and fruitful discussions on hybrid RANS/LES methods. The work of Section 3 has been performed using HPC resources from GENCI-CINES (Grant 2020-A0082A11470) and internal ONERA and Airbus HPC resources. The work of C. Nardoni and F. Bordeu has been carried out in the framework of project TOP, at IRT SystemX, Paris-Saclay, France, and therefore supported by "Programme d'Investissements d'Avenir". The authors of Section 5 would like to thank Priscillia Labourg for code prototyping, and TotalEnergies and the RING Consortium sponsors for their support. Figure 5.1 was prepared with the Dionisos (Beicip) and SKUA-GOCAD (Emerson) software. The work of Section 6 has been performed using HPC resources from GENCI (Grants 2020-A0072B06880, 2021-A0092B11072 and 2021-A009A00611) and from CRIANN (Grant 2012006). C. Dapogny is partly supported by the project ANR-18-CE40-0013 SHAPO, financed by the French Agence Nationale de la Recherche (ANR). A. Froehly would like to thank the Mmg open-source consortium for partially funding her.

References

[1] Acoustic Reference Nozzle with Mach 0.97, Unheated Jet Flow. `https://www.grc.nasa.gov/www/wind/valid/arn/index.html`, Accessed: 2021-05-04.

[2] ANSA. The advanced CAE pre-processing software for complete model build up. `https://www.beta-cae.com/ansa.htm`, Accessed: 2021-05-04.

[3] `Mmg` version 5.5.2. `https://github.com/MmgTools/mmg`. SWHID: swh:1:rel:fe173a75f45f079d363d5a82204c9737550c5d79.

[4] Frédéric Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30(2):175–200, 2014.

[5] Grégoire Allaire, Charles Dapogny, and Pascal Frey. Shape optimization with a level set based mesh evolution method. *Comput. Methods Appl. Mech. Eng.*, 282:22–53, 2014.

[6] Grégoire Allaire, Charles Dapogny, and François Jouve. Shape and topology optimization. In *Geometric partial differential equations. Part II*, volume 22 of *Handbook of Numerical Analysis*, pages 1–132. Elsevier/North Holland, 2020.

[7] Grégoire Allaire, François Jouve, and Anca-Maria Toader. A level-set method for shape optimization. *C. R. Math. Acad. Sci. Paris*, 334(12):1125–1130, 2002.

[8] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.*, 194(1):363–393, 2004.

[9] Grégoire Allaire and Marc Schoenauer. *Conception optimale de structures*, volume 58. Springer, 2007.

[10] P. Anquez, M. Zakari, and G. Caumon. Comparing Three DFN Simplification Strategies for Two-Phase Flow Applications. In *ECMOR XVII*, volume 2020, pages 1–21. EAGE Publications, 2020.

[11] Timothy J. Baker. Mesh movement and metamorphosis. *Engineering with Computers*, 18(3):188–198, 2002.

[12] Francesca Basile, Jean-Baptiste Chapelier, Marta de la Llave Plata, Romain Laraufie, and Pascal Frey. A high-order h-adaptive discontinuous Galerkin method for unstructured grids based on a posteriori error estimation. In *AIAA Scitech 2021 Forum*, page 1696, 2021.

[13] F. Bassi, L. Botti, A. Colombo, A. Crivellini, M. Franciolini, A. Ghidoni, and G. Noventa. A p-adaptive matrix-free discontinuous Galerkin method for the implicit LES of incompressible transitional flows. *Flow, Turbulence and Combustion*, 105(2):437–470, 2020.

[14] Pierre Benard, Guillaume Balarac, Vincent Moureau, Cécile Dobrzynski, Ghislain Lartigue, and Yves D'Angelo. Mesh adaptation for large-eddy simulations in complex geometries. *Int. J. Numer. Methods Fluids*, 81(12):719–740, 2016.

[15] Pierre Benard, Ghislain Lartigue, Vincent Moureau, and Renaud Mercier. Large-Eddy Simulation of the lean-premixed PRECCINSTA burner with wall heat loss. *Symposium (International) on Combustion*, 37(4):5233–5243, 2019.

[16] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer, 2013.

[17] Paul-Emile Bernard, Nicolas Chevaugeon, Vincent Legat, Eric Deleersnijder, and Jean-François Remacle. High-order h-adaptive discontinuous Galerkin methods for ocean modelling. *Ocean Dynamics*, 57(2):109–121, 2007.

[18] T. Bodin, M. Sambridge, and K. Gallagher. A self-parametrizing partition model approach to tomographic inverse problems. *Inverse Probl.*, 25(5):055009, 2009.

[19] François Bonneau, Guillaume Caumon, and Philippe Renard. Impact of a Stochastic Sequential Initiation of Fractures on the Spatial Correlations and Connectivity of Discrete Fracture Networks. *J. Geophys. Res. Solid Earth*, 121(8):5641–5658, 2016.

[20] Houman Borouchaki and Paul-Louis George. *Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling*. John Wiley & Sons, 2017.

[21] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC Press, 2010.

[22] Guillaume Brès, Peter Jordan, Vincent Jaunet, Maxime Le Rallic, André Cavalieri, Aaron Towne, Sanjiva Lele, Tim Colonius, and Oliver Schmidt. Importance of the nozzle-exit boundary-layer state in subsonic turbulent jets. *J. Fluid Mech.*, 851:83–124, 2018.

[23] José G. Castaños and John E. Savage. The Dynamic Adaptation of Parallel Mesh-Based Computation. In *PPSC*, 1997.

[24] G. Caumon, G. Gray, C. Antoine, and M.-O. Titeux. Three-Dimensional Implicit Stratigraphic Model Building From Remote Sensing Data on Tetrahedral Meshes: Theory and Application to a Regional Model of La Popa Basin, NE Mexico. *IEEE Trans. Geosci. Rem. Sens.*, 51(3):1613–1621, 2013.

[25] Peter A. Cavallo, Neeraj Sinha, and Gregory M. Feldman. Parallel Unstructured Mesh Adaptation Method for Moving Body Applications. *AIAA J.*, 43(9):1937–1945, 2005.

[26] Pierre Chantelot. *Rebonds spéciaux de liquides*. PhD thesis, Université Paris-Saclay, 2018. `https://pastel.archives-ouvertes.fr/tel-02011789`.

[27] Siu-Wing Cheng, Tamal Krishna Dey, Jonathan Shewchuk, and Sartaj Sahni. *Delaunay mesh generation*. CRC Press, 2013.

[28] Nicoles Cherpeau, Guillaume Caumon, Jef Caers, and Bruno Lévy. Method for Stochastic Inverse Modeling of Fault Geometry and Connectivity Using Flow Data. *Mathematical Geosciences*, 44(2):147–168, 2012.

[29] Sung Nok Chiu, Dietrich Stoyan, W. S. Kendall, and Joseph Mecke. *Stochastic geometry and its applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, third edition edition, 2013.

[30] Nikos Chrisochoides and Démian Nave. Parallel Delaunay mesh generation kernel. *Int. J. Numer. Meth. Engng.*, 58(2):161–176, 2003.

[31] Philippe G. Ciarlet. *The finite element method for elliptic problems*, volume 40. Society for Industrial and Applied Mathematics, 2002.

[32] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, Guido Ranzuglia, et al. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy, 2008.

[33] Nicolas Clausolles, Pauline Collon, and Guillaume Caumon. Generating variable shapes of salt geobodies from seismic images and prior geological knowledge. *Interpretation*, 7(4):T829–T841, 2019.

[34] A. Colombo, G. Manzinali, A. Ghidoni, G. Noventa, M. Franciolini, A. Crivellini, and F. Bassi. A p-adaptive implicit discontinuous Galerkin method for the under-resolved simulation of compressible turbulent flows. In *7nd European Conference on Computational Fluid Dynamics*, 2018.

[35] Gaëtan Compère, Jean-François Remacle, Johan Jansson, and Johan Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Engng.*, 82(7):843–867, 2010.

[36] Charles Dapogny. *Shape optimization, level set methods on unstructured meshes and mesh evolution*. PhD thesis, Paris 6, 2013.

[37] Charles Dapogny, Cécile Dobrzynski, and Pascal Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *J. Comput. Phys.*, 262:358–378, 2014.

[38] Philippe Davy, Romain Le Goc, and Caroline Darcel. A model of fracture nucleation, growth and arrest, and consequences for fracture density and scaling: a discrete fracture network model. *J. Geophys. Res. Solid Earth*, 118(4):1393–1407, 2013.

[39] H. L. De Cougny and Mark S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *Int. J. Numer. Meth. Engng.*, 46(7):1101–1125, 1999.

[40] Sébastien Deck. Recent improvements in the zonal detached eddy simulation (ZDES) formulation. *Theor. Comput. Fluid Dyn.*, 26(6):523–550, 2012.

[41] Olivier Desjardins, Vincent Moureau, and Heinz Pitsch. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. *J. Comput. Phys.*, 227(18):8395–8416, 2008.

[42] Hugues Digonnet, Thierry Coupez, Patrice Laure, and Luisa Silva. Massively parallel anisotropic mesh adaptation. *Int. J. High Perform. Comput. Appl.*, 33(1):3–24, 2017.

[43] Cécile Dobrzynski. *Adaptation de maillage anisotrope 3d et application à l'aéro-thermique des bâtiments*. PhD thesis, Université Pierre et Marie Curie - Paris VI, 2005. `https://hal.archives-ouvertes.fr/tel-00120327`.

[44] Cécile Dobrzynski and Pascal Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th international Meshing Roundtable*, pages 177–194. 2008.

[45] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Trans. Inf. Syst.*, 74(1):214–224, 1991.

[46] Vít Dolejší. hp-DGFEM for nonlinear convection-diffusion problems. *Math. Comput. Simul.*, 87:87–118, 2013.

[47] Guillaume Durey, Quentin Magdelaine, Mathias Casiulis, Hoon Kwon, Julien Mazet, Pierre Chantelot, Anaïs Gauthier, Christophe Clanet, and David Quéré. Droplets impaling on a cone. *Phys. Rev. Fluids*, 5(11), 2020.

[48] Pierre Duysinx and Ole Sigmund. New developments in handling stress constraints in optimal material distribution. In *7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization*, page 4906, 1998.

[49] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements*, volume 159. Springer, 2013.

[50] Florian Feppon. *Shape and topology optimization of multiphysics systems*. PhD thesis, Université Paris Saclay, préparée à l'École polytechnique, 2019.

[51] Florian Feppon, Grégoire Allaire, Charles Dapogny, and Pierre Jolivet. Topology optimization of thermal fluid–structure systems using body-fitted meshes and parallel computing. *J. Comput. Phys.*, page 109574, 2020.

[52] Joseph E. Flaherty, Raymond. M. Loy, Can Özturan, Mark S. Shephard, Boleskaw K. Szymanski, James D. Teresco, and L. H. Ziantz. Parallel structures and dynamic load balancing for adaptive finite element computation. *Appl. Numer. Math.*, 26(1):241–263, 1998.

[53] André Fourno, Tri-Dat Ngo, Benoit Noetinger, and Christian La Borderie. FraC: A new conforming mesh method for discrete fracture networks. *J. Comput. Phys.*, 376:713–732, 2019.

[54] Tobias Frank, Anne-Laure Tertois, and Jean-Laurent Mallet. 3D-reconstruction of complex geological interfaces from irregularly distributed and noisy point data. *Computers & Geosciences*, 33(7):932–943, 2007.

[55] Pascal Frey and Paul-Louis George. *Mesh generation: application to finite elements*. ISTE, 2007.

[56] Christopher G. Galley, Peter G. Lelièvre, and Colin G. Farquharson. Geophysical inversion for 3D contact surface geometry. *Geophysics*, 85(6):K27–K45, 2020.

[57] Fabien Gand and Maxime Huet. On the generation of turbulent inflow for hybrid RANS/LES jet flow simulations. *Comput. Fluids*, 216:104816, 2021.

[58] G. Gassner, C. Altmann, F. Hindenlang, M. Staudenmeier, and C. D. Munz. Explicit Discontinuous Galerkin Schemes with Adaptation in Space and Time. In *36th CFD/ADIGMA course on hp-adaptive and hp-multigrid methods, VKI LS*. 2009.

[59] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *Int. J. Numer. Meth. Engng.*, 79(11):1309–1331, 2009.

[60] Jérémie Giraud, Mark Lindsay, and Mark Jessell. Generalization of level-set inversion to an arbitrary number of geologic units in a regularized least-squares framework. *Geophysics*, 86(4):R623–R637, 2021.

[61] Gabriel Godefroy, Guillaume Caumon, Gautier Laurent, and François Bonneau. Multi-scenario interpretations from sparse fault evidence using graph theory and geological rules. *J. Geophys. Res. Solid Earth*, 126(2):e2020JB020022, 2021.

[62] Antoine Henrot and Michel Pierre. *Shape Variation and Optimization*, volume 28 of *EMS Tracts in Mathematics*. European Mathematical Society, 2018.

[63] Simin Huang, Florian Wellmann, Gabriele Marquart, Michael Herty, and Christoph Clauser. Shape Optimization Methods Locating Layer Interfaces in Geothermal Reservoirs. *Energy Procedia*, 76:321–330, 2015.

[64] M. Karimi-Fard and L. J. Durlofsky. A general gridding, discretization, and coarsening methodology for modeling flow in porous formations with discrete geological features. *Adv. Water Resources*, 96:354–372, 2016.

[65] M. Karimi-Fard and A. Firoozabadi. Numerical Simulation of Water Injection in 2D Fractured Media Using Discrete-Fracture Model. In *All Days*, pages SPE–71615–MS. SPE, 2001.

[66] Tobias Leicht and Ralf Hartmann. Error estimation and hp-adaptive mesh refinement for discontinuous Galerkin methods. In *Adaptive high-order methods in computational fluid dynamics*, pages 67–94. World Scientific, 2011.

[67] Tobias Leicht, Jens Jägersküpper, Daniel Vollmer, Axel Schwöppe, Ralf Hartmann, Jens Fiedler, and Tobias Schlauch. Dlr-Project Digital-X-Next Generation CFD Solver'Flucs'. *CEAS Aeronautical Journal*, 2016.

[68] Daniel S. H. Lo. *Finite element mesh generation*. CRC Press, 2014.

[69] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

[70] Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part I: well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.

[71] Catherine Mavriplis. A posteriori error estimators for adaptive spectral element techniques. In *Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 333–342. Springer, 1990.

[72] Marek Krzysztof Misztal and Jakob Andreas Bærentzen. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.*, 31(3):1–12, 2012.

[73] J. E. P. Monteagudo and A. Firoozabadi. Control-volume method for numerical simulation of two-phase immiscible flow in two- and three-dimensional discrete-fractured media: SIMULATION OF FLOW IN FRACTURED MEDIA. *Water Resources Research*, 40(7), 2004.

[74] Vincent Moureau, Pascale Domingo, and Luc Vervisch. Design of a massively parallel CFD code for complex geometries. *C. R. Méc. Acad. Sci. Paris*, 339(2-3):141–148, 2011.

[75] Vincent Moureau, Pascale Domingo, and Luc Vervisch. From Large-Eddy Simulation to Direct Numerical Simulation of a lean premixed swirl flame: Filtered laminar flame-PDF modeling. *Combustion and Flame*, 158(7):1340–1357, 2011.

[76] F. Murat and J. Simon. Sur le contrôle par un domaine géométrique. Pré-publication du Laboratoire d'Analyse Numérique, (76015), 1976.

[77] Hussein Mustapha and Roussos Dimitrakopoulos. Discretizing two-dimensional complex fractured fields for incompressible two-phase flow. *Int. J. Numer. Methods Fluids*, 65(7):764–780, 2011.

[78] Fabio Naddei, Marta de la Llave Plata, Vincent Couaillier, and Frédéric Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods. *J. Comput. Phys.*, 376:508–533, 2019.

[79] Andrej Neifeld, Dirk Boenke, Juergen Dierke, and Roland Ewert. Jet noise prediction with Eddy relaxation source model. In *21st AIAA/CEAS Aeroacoustics Conference*, page 2370, 2015.

[80] Leonid Oliker, Rupak Biswas, and Harold N. Gabow. Parallel tetrahedral mesh adaptation with dynamic load balancing. *Parallel Comput.*, 26(12):1583–1608, 2000.

[81] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer, 2006.

[82] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[83] Dimitris Papadopoulos, Michael Herty, Volker Rath, and Marek Behr. Identification of uncertainties in the shape of geophysical objects with level sets and the adjoint method. *Comput. Geosci.*, 15(4):737–753, 2011.

[84] Michael A. Park, Adrien Loseille, Joshua Krakos, Todd R. Michal, and Juan J. Alonso. Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Towards CFD 2030. In *AIAA AVIATION Forum*. American Institute of Aeronautics and Astronautics, 2016. art. 3323.

[85] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 112, 2006.

[86] Savinien Pertant, Manuel Bernard, Giovanni Ghigliotti, and Guillaume Balarac. A finite-volume method for simulating contact lines on unstructured meshes in a conservative level-set framework. *J. Comput. Phys.*, 444:110582, 2021.

[87] Jean-François Remacle, Joseph E. Flaherty, and Mark S. Shephard. An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems. *SIAM Rev.*, 45(1):53–72, 2003.

[88] Jean-François Remacle, Christophe Geuzaine, Gaëtan Compère, and B. T. Helenbrook. Adaptive mesh generation and visualization. In *Encyclopedia of Aerospace Engineering*. 2010.

[89] Pierre Sagaut, Marc Terracol, and Sébastien Deck. *Multiscale and multiresolution approaches in turbulence-LES, DES and Hybrid RANS/LES Methods: Applications and Guidelines*. World Scientific, 2013.

[90] James A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, 1999.

[91] James A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge University Press, 1999.

[92] Jonathan Richard Shewchuk. What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures. In *Proceedings of the 11th International Meshing Roundtable*, pages 115–126. Sandia National Laboratories, 2002.

[93] Hang Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):1–36, 2015.

[94] Jan Sokolowski and Jean-Paul Zolésio. *Introduction to shape optimization*. Springer, 1992.

[95] Philippe Spalart. Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach. In *Proceedings of first AFOSR international conference on DNS/LES*. Greyden Press, 1997.

[96] Philippe Spalart and Steven Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[97] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edition, 2021. `https://doc.cgal.org/5.3/{Manual}/packages.html`.

[98] Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill. *Handbook of grid generation*. CRC Press, 1998.

[99] M. G. Vallet, F. Hecht, and B. Mantel. Anisotropic control of mesh generation based upon a Voronoi type method. In *Numerical grid generation in computational fluid dynamics and related fields*, pages 93–103. North-Holland, 1991.

[100] Li Wang and Dimitri J. Mavriplis. Adjoint-based h-p adaptive discontinuous Galerkin methods for the 2D compressible Euler equations. *J. Comput. Phys.*, 228(20):7643–7661, 2009.

[101] Florian Wellmann and Guillaume Caumon. 3-D Structural geological models: Concepts, methods, and uncertainties. *Adv. Geophys.*, 59:1–121, 2018.

[102] Chaoshui Xu and Peter Dowd. A new computer code for discrete fracture network modelling. *Computers & Geosciences*, 36(3):292–301, 2010.

[103] Liang Yang, David Hyde, Ognjen Grujic, Celine Scheidt, and Jef Caers. Assessing and visualizing uncertainty of 3D geological surfaces using level sets with stochastic motion. *Computers & Geosciences*, 122:54–67, 2019.