Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems

Charles Dapogny, Cécile Dobrzynski, Pascal Frey

▶ **To cite this version:**

Charles Dapogny, Cécile Dobrzynski, Pascal Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. Journal of Computational Physics, 2014, 10.1016/j.jcp.2014.01.005 . hal-00804636

HAL Id: hal-00804636

https://hal.sorbonne-universite.fr/hal-00804636v1

Submitted on 26 Mar 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THREE-DIMENSIONAL ADAPTIVE DOMAIN REMESHING, IMPLICIT DOMAIN MESHING, AND APPLICATIONS TO FREE AND MOVING BOUNDARY PROBLEMS

C. DAPOGNY[1,2], C. DOBRZYNSKI[3,4], P. FREY [1]

[1] *UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France.*
[2] *Renault DREAM-DELT'A Guyancourt, France.*
[3] *IMB, Université de Bordeaux, 33405 Talence cedex France.*
[4] *Team Bacchus, INRIA, 33405 Talence cedex, France.*

ABSTRACT

The aim of this paper is to propose a method for dealing with the problem of mesh deformation (or mesh evolution) in the context of free and moving boundary problems, in three space dimensions. The method consists in combining two different numerical parameterizations of domains: on the one hand, domains are equipped with a computational tetrahedral mesh, and on the other hand, they are represented as the negative subdomain of a 'level set' function. We then consistently switch from one description to the other, depending on their respective convenience with respect to the operations to be performed. Among other things, doing so implies to be able to get a computational mesh from an implicitly-defined domain. This in turns relies on an algorithm for handling three-dimensional domain remeshing (that is, remeshing at the same time both surface and volume parts of a given tetrahedral mesh). Applications are considered in the fields of mesh generation, shape optimization, and computational fluid dynamics.

1. Introduction

Many time-dependent physical or mechanical phenomena occur in evolving domains. Examples of these so-called free or moving boundary problems are melting/solidification problems, multiphase fluid flows, etc... see [10] for many other illustrations. Dealing with such problems is especially difficult - as well from the theoretical point of view as from the numerical one - mainly because of the mutual influence of the physical process and the evolving geometry.

When it comes to the numerical resolution of such problems, two main classes of methods can be distinguished, depending on whether an *explicit* discretization of the considered evolving domain (or only the interface) is used in the computational formulation of the physical problem.

As parts of the first class, *Lagrangian* methods bring into play an exact mesh of the domain(s), which is updated in the course of the iterative process [28]; *front-tracking formulations* [42] [24] feature a reconstruction of the evolving boundary (or interface), which is used in the resolution of the physical problem. These methods potentially enjoy the best accuracy; however, they may turn out difficult to carry out because of the mesh update or reconstruction steps.

On the opposite, *Eulerian* methods favor the use of a fixed mesh of a computational box, which is the support for the storage of several quantities involved in the numerical resolution of the physical equations. Such a method is the level set method (see [19] or [37] for reviews), according to which the evolving domain is defined in the sense of implicit functions, and the domain evolution problem is traded for a PDE problem. These methods do not involve mesh deformation, or reconstruction; yet, their implementation may demand an approximation to the physical problem, to blend in the knowledge of the domain in the equations, from the quantities whose evolutions are tracked.

Of course, this short presentation is by no means exhaustive; what's more, hybrid methods have emerged, which retain some characteristics from both classes: thus, extended finite element methods (XFEM) [29] incorporate the imprint of a propagating discontinuity between two phases into the finite element formulation

of the problem. We should also mention the interesting two-dimensional work of Persson (see [34], chap. 5), which combines the level set method with a mesh generation tool for implicit geometries.

The main purpose of this paper is to present a method for free and moving boundary problems in three space dimensions which relies on two complementary descriptions of domains. On the one hand, they are exactly meshed, so that no approximation of the considered mechanical problems is required. On the other hand, they are considered as implicit geometries when it comes to compute their evolution, using the level set method. The core of the method is an algorithm for three-dimensional meshing of implicit geometries, which itself turns out to be an easy derivation of a more general three-dimensional domain remeshing algorithm. Note that the problem of meshing implicitly-defined domains is not new, and has been addressed from a rather different point of view in [35].

Consequently, a large part of this paper is devoted to three-dimensional adaptive domain remeshing, which is a topic of interest on its own. As we intend the domain remeshing procedure to be part of a general mesh evolution process, we do not want to assume more input information than just a tetrahedral mesh as the set of its vertices and tetrahedra - yet retaining the possibility that additional data may be supplied (e.g. labels on the boundary triangles, normal vectors at the boundary vertices, etc...). On that basis, a relevant continuous geometry has to be invented to serve as a guide throughout the remeshing operation. For instance, when a node is created on the surface part of the mesh, we expect it to belong to some kind of 'smooth model' associated to the surface, rather than on the triangulated surface itself, for obvious approximation reasons.

Generating a tetrahedral mesh out of a sole surface triangulation is a very difficult problem, which very few existing algorithms manage to tackle with suitable robustness. Hence, when it comes to remeshing a supplied tetrahedral mesh, we think it better to handle at the same time both boundary and interior parts, relying on local remeshing operators which allow to pass from one 'valid' mesh to another, rather than dropping the tetrahedral part, remeshing the surface part, then generating a new tetrahedral mesh, associated to the new surface triangulation.

The remainder of this paper is organized as follows: after presenting shortly the stakes of the remeshing problem in section 2, a surface model for creating a continuous geometry in connection with the input mesh is introduced in section 3, as well as the rules for appraising the quality of this mesh as an approximation of this geometry. Then, section 4 goes into the specifics of three-dimensional remeshing: the local mesh modification operators are described, as well as the way to drive them so as to comply with the previous geometric controls. The global remeshing strategy is outlined, and numerical examples are discussed. From this domain remeshing algorithm, an algorithm for meshing three-dimensional implicit domains is subsequently derived in section 5. Two applications of the latter algorithm are then introduced: a method for generating a tetrahedral mesh from a triangulated surface in section 6, then a generic process for free and moving boundary problems in section 7.

2. Description of the remeshing problem and main notations

Let $\mathcal{T}$ be a conforming tetrahedral mesh in $\mathbb{R}^3$ (see [7] or [22] for the basic material as regards meshing). We shall consistently identify $\mathcal{T}$ with the underlying polygonal domain. $\mathcal{T}$ is intended as an approximation of a continuous *ideal domain* Ω.

The topological boundary $\mathcal{S}_\mathcal{T}$ of $\mathcal{T}$ is a conforming triangulated surface embedded in $\mathbb{R}^3$, meant to represent $\partial\Omega$: it is referred to as the *associated surface mesh to* $\mathcal{T}$. In the following, we will assume $\mathcal{S}_\mathcal{T}$ is an *interpolating* triangulation of $\partial\Omega$, meaning the vertices of $\mathcal{S}_\mathcal{T}$ lie on $\partial\Omega$.

Mesh $\mathcal{T}$ may be inappropriate as a mesh of Ω for at least two reasons:

- The surface triangulation $\mathcal{S}_\mathcal{T}$ may be a poor geometric approximation of $\partial\Omega$, or $\mathcal{T}$ may contain elements which do not match a desired size feature, imposed by the user, or ill-shaped elements as far as mesh quality is concerned.
- $\mathcal{T}$ may be of poor mesh quality, or may not be adapted to a desired size feature.

Our goal is to remesh $\mathcal{T}$ into a new mesh $\widetilde{\mathcal{T}}$, which is a close geometric approximation of Ω within a specified range, is well-shaped, and possibly adapted to a specified local size feature.

Throughout this paper, the produced meshes are expected to be the support of physical or mechanical computations (e.g. with finite element or finite volume methods). The notion of *quality* of a tetrahedron (or

a mesh being well-shaped) is therefore oriented in this perspective. In such a context, many classical error estimates (see [9] for instance) involve the *eccentricity* $\sigma_K = \rho_K/h_K$ of elements $K \in \mathcal{T}$, where ρ_K stands for the inradius of K, and h_K for its diameter. In the sequel, we will rather rely on the following quality function $Q(K)$ of a tetrahedron K with edges $e_1, ..., e_6$, which retains the same theoretical meaning as σ_K, but shows a better numerical ability when it comes to discriminate 'good' from 'average', or 'bad' elements:

$$Q(K) = \alpha \frac{\mathrm{Vol}(K)}{\left(\sum_{i=1}^{6} ||e_i||^2\right)^{\frac{3}{2}}}, \ \alpha = 144\sqrt{3}.$$

One can show that, for any tetrahedron K, $Q(K) \leq 1$, and equality holds if and only if K is regular.

The proposed approach is a local, iterative remeshing procedure: starting from $\mathcal{T}$, and conducting operations which affect very limited areas of the meshes at hand, a sequence of meshes $\mathcal{T} = \mathcal{T}_1, ..., \mathcal{T}_k, ..., \mathcal{T}_N = \widetilde{\mathcal{T}}$ is produced, that converges towards $\widetilde{\mathcal{T}}$. It relies on four user-defined parameters ε, h_{min}, h_{max} and h_{grad}:

- ε is the tolerance over the geometric approximation of $\partial\Omega$: $\widetilde{\mathcal{T}}$ should be such that:
$$d^H(\partial\Omega, \mathcal{S}_{\widetilde{\mathcal{T}}}) \leq \varepsilon,$$
where d^H denotes the Hausdorff distance between compact subsets of $\mathbb{R}^3$.
- h_{min} (resp. h_{max}) is the minimal (resp. maximal) authorized size for an edge of $\widetilde{\mathcal{T}}$. They should be understood as 'security bounds' over the desired length scale for the resulting mesh $\widetilde{\mathcal{T}}$, and it is expected that a wise choice of parameter ε govern the size prescription for elements of $\widetilde{\mathcal{T}}$ on its own.
- h_{grad} is a control parameter for the variation of edge lengths among $\widetilde{\mathcal{T}}$: for any two edges $ap, bp \in \widetilde{\mathcal{T}}$, one should have:

$$\frac{1}{h_{grad}} \leq \frac{||b-p||}{||p-a||} \leq h_{grad}. \tag{1}$$

Whereas h_{min}, h_{max} and ε depend by essence on the length scale of $\mathcal{T}$, h_{grad} does not. Typical values for h_{grad} are $1.2, 1.3, ...$ As we shall see in the sequel, the role of the last parameter is only to control the mesh quality when mechanical computations using mesh $\mathcal{T}$ are considered.

Notations: If $p \in \mathcal{T}$ (resp. $\mathcal{S}_\mathcal{T}$) is a vertex, $\mathcal{B}(p)$ (resp. $\mathcal{B}_\mathcal{S}(p)$) is the *ball* (resp. *surface ball*) of p, that is, the set of tetrahedra of $\mathcal{T}$ (resp. triangles of $\mathcal{S}_\mathcal{T}$) sharing p as a vertex.

Similarly, if pq is an edge of $\mathcal{T}$, the *shell* $\mathcal{S}h(pq)$ of pq is the set of tetrahedra of $\mathcal{T}$ sharing pq as an edge.

3. From the triangulated surface to the ideal surface

Certainly, in remeshing $\mathcal{T}$, the associated surface triangulation $\mathcal{S}_\mathcal{T}$ plays a central role. Actually, $\mathcal{S}_\mathcal{T}$ governs on its own the accuracy of $\mathcal{T}$ as a geometric approximation of Ω.

As mentioned above, such an ideal surface $\partial\Omega$ is unknown, and so as to manage the local modifications, we need to create it from the data at hand. Actually, one could think of mainly two ways for doing so.

The first one consists in inferring a whole underlying surface to $\mathcal{S}_\mathcal{T}$ as a pre-processing stage for remeshing. This surface $\partial\Omega$ is then kept in memory, for instance under the form of a *parametrization* $\sigma : U \to \partial\Omega$ (this is the point of view in [3]). The parameter space U can be an open subdomain of $\mathbb{R}^2$ [16] [20], or the surface $\mathcal{S}_\mathcal{T}$ itself [40]. This parametrization is stored, and at each stage of the remeshing procedure, the current triangulation $\mathcal{S}_{\mathcal{T}_k}$ is compared to this guess for $\partial\Omega$ in order to appraise the geometric approximation. Such an approach is well-posed on the theoretical side: all the produced triangulations are compared to one single continuous surface; however the storage and numerous comparisons involved generally turn out quite costly.

On the contrary, we could limit ourselves with generating *local* models for $\partial\Omega$: at each stage $\mathcal{T}_k$ of the remeshing process, when an operation around a node x of $\mathcal{S}_{\mathcal{T}_k}$ is performed, a local parametrization $\sigma_U : U \to V \subset \partial\Omega$ from an open set $U \subset \mathbb{R}^2$ to a neighborhood V of x in $\partial\Omega$ is computed using local features. This approach is obviously more efficient, from the computational point of view, but it raises a difficulty: the triangulation $\mathcal{S}_{\mathcal{T}_k}$ supporting the features from which these local parametrizations are generated may change slightly from one stage to another.

In what follows, we will rely on the second approach, referring with some abuse in terminology to *the* ideal surface $\partial\Omega$ associated to the various surface meshes at hand during the process, neglecting the dependance on the triangulation $\mathcal{T}_k$ used to this end. We will see some heuristics to guarantee that the generated geometric support in a given area does not deviate along the steps of the process.

We are thus led to set rules to infer a piece of the underlying surface $\partial\Omega$ from the datum of a piece of the current discrete geometry at the investigated stage.

3.1. **Identification of the geometric features on the surface mesh.**

The first step in associating to $\mathcal{S}_\mathcal{T}$ a relevant continuous geometry consists in identifying different kinds of entities (points, edges or faces) corresponding to significant geometrical features of $\mathcal{S}_\mathcal{T}$ (and thus $\partial\Omega$) which will constrain the admissible operations carried out during remeshing. We mainly identify:

(1) *geometric edges (or points)*, or *ridges*: edges delimiting two portions of surface which intersect with a sharp angle. Ridges can be inferred from the input triangulation by relying on a threshold value on the dihedral angle between pairs of adjacent triangles.
(2) *reference edges (or points)*: edges at the interface between two triangles $T_i \neq T_j$ holding different labels, corresponding for instance to boundary conditions in a finite element, or finite volume computation.
(3) *singular points*: points which arise as endpoints of at least 3 special edges, and thus cannot be considered as 'regular points' on a ridge curve, reference curve. Such points should not be affected by the process.
(4) *(optionally) required entities*: any user-specified entity which must stay unchanged during remeshing.
(5) ordinary entities.

Once such a classification is reached, we infer from the knowledge of $\mathcal{S}_\mathcal{T}$ approximations of some geometric data attached to $\partial\Omega$, depending on the nature of the considered point - for instance (see [22], §19.2.1):

- In the neighbourhood of a regular vertex x, the ideal surface $\partial\Omega$ is smooth (at least of class $\mathcal{C}^1$) and we will rely on an approximation of $n(x)$, the unit normal vector to $\partial\Omega$ at x, pointing outwards Ω. This is achieved from the discrete surface $\mathcal{S}_\mathcal{T}$ using a formula such as:

$$n(x) \approx \frac{\sum_{T\in\mathcal{B}(x)} \alpha_T \, n_T}{\left|\left|\sum_{T\in\mathcal{B}(x)} \alpha_T \, n_T\right|\right|},$$

  where α_T are coefficients in $[0,1]$ such that $\sum_{T\in\mathcal{B}(x)} \alpha_T = 1$, and n_T is the unit normal (pointing outwards surface $\mathcal{S}_\mathcal{T}$) to a triangle T. Several choices are possible as for the values of α_T. Some authors are used to taking them all equal to one another, others take each α_T proportional to the area of T,... none of these options being clearly independent from the discretization. We retained the last one, which proves satisfactory for our purpose.
- According to the above terminology, non singular ridge vertices x of $\mathcal{T}$ are vertices belonging to *ridge curves* of $\partial\Omega$, that is curves delimiting two portions of surfaces which intersect with a sharp dihedral angle. Such points enjoy two normal vectors (one for each piece of surface), say $n_1(x), n_2(x)$, which are reconstructed in the discrete context in the same way as for regular vertices, and a tangent vector $\tau(x)$ (the tangent vector to the ridge curve), which is uniquely determined by its belonging to two distinct 'tangent' planes.

These supplementary pieces of information about the ideal surface $\partial\Omega$, approximated from the discrete geometry, will allow us to define a local surface model for $\partial\Omega$.

3.2. **Local reconstruction of the ideal surface from the discrete geometry.**

The purpose of this section is to provide the rules for inferring the local geometry of the ideal surface $\partial\Omega$ around a triangle T of $\mathcal{S}_{\mathcal{T}_k}$ from the geometric features attached to T, described in section 3.1. This proposed surface model is very reminiscent of [43].

In the following, we make the assumption that each triangle $T = (a_0\, a_1\, a_2) \in \mathcal{S}_\mathcal{T}$ accounts for a *smooth* portion of $\mathcal{S}$, whose boundaries may be ridge curves, reference curves, etc... The portion of $\partial\Omega$ associated to T is modeled as a cubic piece of surface $\sigma(\widehat{T})$, where

$$\widehat{T} := \left\{(u,v) \in \mathbb{R}^2, |\, u \geq 0,\ v \geq 0,\ w := 1-u-v \geq 0\right\}$$

stands for the reference triangle in the plane, and each component of $\sigma : \widehat{T} \to \mathbb{R}^3$ is a polynomial of total degree 3 in two variables $u, v \in \widehat{T}$. It will turn out convenient to write σ under the form of a Bézier cubic polynomial [18]:

$$\forall (u,v) \in \widehat{T},\ \ \sigma(u,v) = \sum_{\substack{i,j,k \in \{0,\dots,3\} \\ i+j+k=3}} \frac{3!}{i!j!k!} w^i u^j v^k\, b_{i,j,k}, \tag{2}$$

where the $b_{i,j,k} \in \mathbb{R}^3$ are *control points*, yet to be specified. See figure 1 for an illustration.

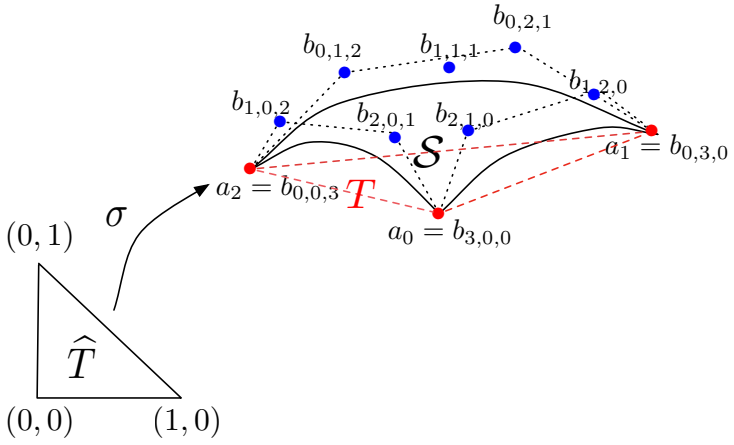


FIGURE 1. A piece of parametric Bézier cubic surface, associated to triangle T, with control points $b_{i,j,k}$.

We also denote as $\gamma_0, \gamma_1, \gamma_2$ the *boundary curves* of $\sigma(\widehat{T})$:

$$\forall t \in [0,1],\ \gamma_0(t) = \sigma(1-t,t),\ \gamma_1(t) = \sigma(0,t),\ \gamma_2(t) = \sigma(t,0).$$

The choice of the control points $b_{i,j,k}$ is dictated by the geometrical features of $\partial\Omega$ we approximated in section 3.1, or by other requirements we may want our local geometry to meet.

3.2.1. *Choice of the three 'vertex' control points.* The natural requirement that $\mathcal{S}_\mathcal{T}$ should interpolate $\partial\Omega$ prompts the choice of the three vertices of T as the three vertices of $\sigma(\widehat{T})$, that is:

$$b_{3,0,0} = a_0, \quad b_{0,3,0} = a_1, \text{ and } b_{0,0,3} = a_2.$$

3.2.2. *Choice of the six 'curve' control points.* We required $\sigma(\widehat{T})$ should be a smooth piece of surface. In particular, $\sigma(\widehat{T})$ enjoys a tangent plane $T_{a_i}\mathcal{S}$ at each vertex a_i, whose normal vector n_i should match the reconstructed geometric information at a_i.

On the other hand, it is well-known (see [18] for instance) that the whole geometry of Bézier curves and surfaces can be expressed in terms of their control points; for instance, the tangent vector at a_0 to the boundary curve γ_2 is $3(b_{2,1,0} - b_{3,0,0})$, and the tangent at a_0 to γ_1 is $3(b_{2,0,1} - b_{3,0,0})$.

Hence, the tangent plane to $\sigma(\widehat{T})$ at a_0 is the expected tangent plane $T_{a_0}\partial\Omega$ provided $b_{2,1,0}$ and $b_{2,0,1}$ are chosen in such a way that $(b_{2,1,0} - a_0)$ and $(b_{2,0,1} - a_0)$ are non colinear, and both orthogonal to n_0. Similar relations hold when it comes to a_1, a_2 and control points $b_{0,2,1}, b_{1,2,0}, b_{1,0,2}$ and $b_{0,1,2}$.

This still allows some latitude as for the choice of these coefficients. In [43], the authors propose to take, for instance, $b_{2,1,0}$ as the orthogonal projection over $T_{a_0}\partial\Omega$ of point $a_0 + (a_1 - a_0)/3$. Instead of this, we

propose to use the fact we want our local surface reconstruction to be as independent as possible from the support triangle T used for its computation to devise some heuristics as for the choice of these control points.

This means that we would like $\gamma_0, \gamma_1, \gamma_2$ to be independent on the choice of the points on these curves. Because on any Riemannian manifold two 'close enough' points are connected by a unique geodesic curve [15], a way to enforce this independency would be to choose the control points so that $\gamma_0, \gamma_1, \gamma_2$ are geodesics of $\sigma(\widehat{T})$, that is, curves with constant speed. This property, in turn can only be enforced in some kind of 'weak sense': we imposed in the case of γ_0 (similar conditions hold for γ_1, γ_2) that $\gamma_0\prime(0)$ should be colinear to the orthogonal projection of $(a_2 - a_1)$ over $T_{a_1}\mathcal{S}$, and have a fixed norm $||\gamma_0\prime(0)|| = ||a_2 - a_1||/3$, and symmetrically for $\gamma_0\prime(1)$. Doing so uniquely determines the six coefficients attached to the boundary curves.

3.2.3. *Choice of the central coefficient.* We simply take:

$$b_{1,1,1} = m + \frac{m - v}{2}, \;\; v := \frac{a_0 + a_1 + a_2}{3} \;; \;\; m := \frac{b_{2,1,0} + b_{2,0,1} + b_{1,2,0} + b_{0,2,1} + b_{1,0,2} + b_{0,1,2}}{6},$$

which guarantees that, if there exists a *quadratic* polynomial parametrization $\widetilde{\sigma} : \widehat{T} \to \mathbb{R}^3$ whose boundary curves $t \mapsto \widetilde{\sigma}(1-t, t)$, $\widetilde{\sigma}(t, 0)$ and $\widetilde{\sigma}(0, t)$ coincide with γ_0, γ_1 and γ_2 respectively, then $\sigma = \widetilde{\sigma}$ over $\widehat{T}$ [18]. Note that the choice of the central coefficient $b_{1,1,1}$ does not affect the geometry of $\gamma_0, \gamma_1, \gamma_2$.

Remarks 1.
- *The choice of the four control points along each boundary curve γ_i only involves geometric data attached to this curve. This fact implies that our rules for generating a piece of $\partial\Omega$ are consistent from one triangle to its neighbor, that is, if $T_i, T_j \in \mathcal{S}_\mathcal{T}$ share a common edge $e_{i,j}$, the underlying boundary curve associated to $e_{i,j}$ via the local parametrization generated from T_i is the same as from T_j.*
- *Actually, the proposed rules for generating portions of $\partial\Omega$ from triangles of $\mathcal{S}_\mathcal{T}$ apply accordingly when it comes to generating curves drawn on $\partial\Omega$ from edges of $\mathcal{S}_\mathcal{T}$.*

3.3. **Estimating the gap between the surface triangulation and the ideal surface.**

At some point in the process, we need to measure (or at least to control) how far each triangle T of $\mathcal{S}_\mathcal{T}$ lies from its corresponding part on $\partial\Omega$. Such a knowledge is mandatory when it comes to evaluating whether an operation performed on $\mathcal{T}$ (see section 4.1) improves or degrades the geometric approximation of $\partial\Omega$.

Putting parametrization σ under the form (2) allows to derive a close and fast control over the Hausdorff distance $d^H(T, \sigma(\widehat{T}))$ between a given surface triangle $T \in \mathcal{S}_\mathcal{T}$ and the corresponding piece of ideal surface $\sigma(\widehat{T}) \subset \partial\Omega$, which only involves geometric quantities attached to T.

Indeed, $\sigma(\widehat{T})$ is comprised in the convex hull of the control points $b_{i,j,k}$, because for all $(u, v) \in \widehat{T}$, $\sigma(u, v)$ is a convex combination of the $b_{i,j,k}$. As a consequence, one easily sees that

$$d^H(T, \sigma(\widehat{T})) \leq \max_{\substack{l=0,1,2,\\ i+j+k=3}} d(a_l, b_{i,j,k}), \tag{3}$$

and a similar estimate holds when it comes to controlling the Hausdorff distance between each edge a_1a_2, a_0a_2, or a_0a_1 of T and the corresponding boundary curve γ_0, γ_1, or γ_2 of $\sigma(\widehat{T})$.

4. Discrete three-dimensional domain remeshing

We now describe the salient features of the proposed method for remeshing an input tetrahedral mesh $\mathcal{T}$ of a domain $\Omega \subset \mathbb{R}^3$. This method relies on four parameters, ε (tolerance over geometric approximation), h_{min}, h_{max} (resp. minimum, maximum authorized edge length), and h_{grad} (mesh gradation parameter), whose precise meaning will be specified below.

4.1. **Description of the local remeshing operators.**

In this section, we dwell on an abridged description of the local operators used in the remeshing process of $\mathcal{T}$ (see [13] for internal operators, [21] for surface operators, and [11] for full details). All these admittedly rather classical operators [22] enjoy two forms, depending on whether they are applied to a surface configuration - i.e. to a configuration 'close' to surface triangles of $\mathcal{S}_\mathcal{T}$ - or to a purely internal one. If no particular attention is paid, these operators may invalidate $\mathcal{T}$ (e.g. invert some of its elements), degrade the geometric

approximation of Ω more than the prescribed tolerance, or degrade too much the mesh quality. Consequently, adequate checks have to be performed systematically.

4.1.1. *Edge split.* This is the main tool when it comes to enriching an undersampled mesh. Splitting an edge $pq \in \mathcal{T}$ consists in introducing a new point m in the mesh, then replacing pq by the two edges pm and mq, and updating the connectivities of $\mathcal{T}$ accordingly. In the context of domain remeshing, two cases arise:

- if the processed edge pq is a surface edge, i.e. $pq \in \mathcal{S}_\mathcal{T}$, let $\gamma : [0,1] \to \partial\Omega$ the associated curve to pq (see section 3.2). The new point m is taken as $m = \gamma(\frac{1}{2})$.
- if pq is an internal edge, m is simply inserted as the midpoint of p and q.

Retaining the formal assumption that the local parametrizations σ described in section 3.2 are pieces of the same ideal surface $\partial\Omega$, splitting edges on a surface triangle $T \in \mathcal{S}_\mathcal{T}$ always enhances the geometric approximation of $\partial\Omega$.

Now, there are several ways to split edges within a mesh $\mathcal{T}$: the simplest one consists in traveling all the edges in $\mathcal{T}$, and as soon as an edge meeting the splitting criterion is met, carrying out the splitting operation. However, we noticed that repeatedly splitting elements along only one edge tends in the long run to produce very ill-shaped elements.

For this reason, we favored another approach, which consists in identifying in a first step all the edges of $\mathcal{T}$ that should be split, then to proceed to splitting, resorting to patterns on each tetrahedron K, possibly reiterating the process if some edges need to be split several times, or preventing those splits that may invalidate the mesh.

4.1.2. *Edge collapse.* This is the key ingredient in removing a vertex from a mesh (which proves useful when decimating an oversampled mesh). Collapsing an edge $pq \in \mathcal{T}$ consists in merging its two endpoints into a single one : say p is collapsed onto q for simplicity. The elements of the shell $\mathcal{S}h(pq)$ disappear in the process, and all the other tetrahedra $K \in \mathcal{T}$ which had p as a vertex (i.e. $K \in \mathcal{B}(p)$), now have q instead.

This operator ought to be driven carefully, and several checks are in order, depending on the nature of the processed edge:

- Some collapses are strictly forbidden, e.g. a point p should not be collapsed onto another point q if p belongs to $\mathcal{S}_\mathcal{T}$ and q does not, a ridge point $p \in \mathcal{S}_\mathcal{T}$ should not be collapsed onto another non ridge point $q \in \mathcal{S}_\mathcal{T}$, a required vertex cannot be removed. Some of these checks may be application-dependent.
- If $pq \in \mathcal{S}_\mathcal{T}$, one must ensure that the Hausdorff distance between each new surface triangle and the corresponding part on $\partial\Omega$ is no greater than the prescribed tolerance ε (see section 3.2). Furthermore, some additional checks have to be performed, to avoid invalidating or 'folding' $\mathcal{S}_\mathcal{T}$. Eventually, checks have to be performed on the support tetrahedra $K \in \mathcal{T}$ to the concerned surface triangles $T \in \mathcal{S}_\mathcal{T}$, so that they do not result invalidated in the process.
- If pq is an internal edge, one only need check that the elements $K \in \mathcal{T}$ affected by the operation (which are those of $\mathcal{B}(p) \setminus \mathcal{S}h(pq)$) are not invalidated in the process.

4.1.3. *Edge swap.* Edge swap plays a key role in improving the overall mesh quality, and acts only on the connectivities of $\mathcal{T}$, leaving its vertices' positions unchanged. This operator is significantly different depending on whether it is applied to a surface edge or an internal one.

- If the processed edge pq lies on $\mathcal{S}_\mathcal{T}$, introducing $T_1 = pqa$, $T_2 = pqb \subset \mathcal{S}_\mathcal{T}$ the two surface triangles sharing pq, the only possibility for swapping pq consists in replacing it by ab, and updating $\mathcal{T}$ accordingly (figure 3(a-b)). Such an operation can only be performed when it is consistent with the geometry of $\partial\Omega$, and does not entail too large a gap between the resulting triangles and their corresponding piece of $\partial\Omega$. Besides, the validity of the affected tetrahedra of $\mathcal{T}$ has to be checked.
- Swapping an internal edge pq is more combinatorial [23]. In this case, the enumeration of the vertices of the elements $K \in \mathcal{S}h(pq)$ which are neither p, nor q defines an oriented pseudo-polygon $a_1 ... a_n$. This (non planar) pseudo-polygon is then triangulated, and each resulting triangulated face is connected to p and q to provide a new tetrahedralization of the area once occupied by $\mathcal{S}h(pq)$.

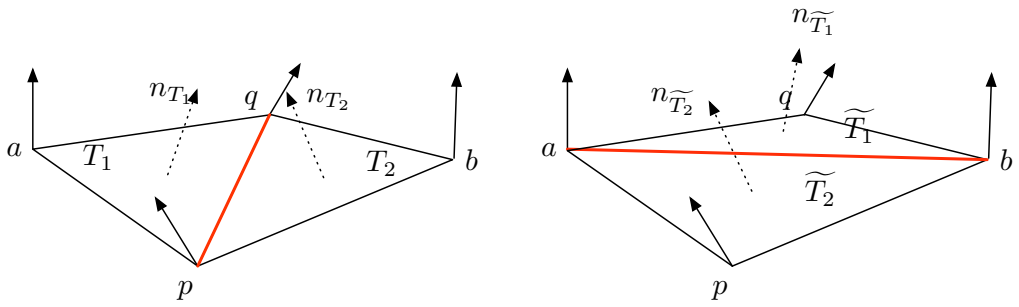


FIGURE 2. Swap of edge pq : triangles T_1, T_2 are updated to $\widetilde{T_1}, \widetilde{T_2}$, a configuration more consistent with the geometric data.

Thus, the number of possible swapped configuration equals the number of triangulations of the pseudo-polygon, that is the *Catalan number* C_n, defined as:

$$C_n = \frac{1}{n+1}\binom{2n}{n},$$

which grows dramatically with n.

So as to avoid a very tedious enumeration of the different configurations until a valid one is found, we adopted a somewhat different approach, less general yet much easier to implement. Swapping edge pq is achieved within two steps (see figure 3(c)):

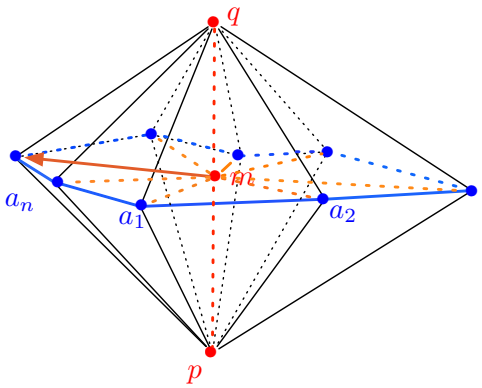


FIGURE 3. Swap of pq, introducing its midpoint m in the mesh, then collapsing it on one of the vertices of the pseudo-polygon associated to $\mathcal{S}h(pq)$.

step 1: pq is split at its midpoint m. All the connections ma_i, $i = 1, ..., n$ are created in the process.
step 2: Point m is collapsed onto one of the a_i: each one of the collapses of edges $ma_1, ..., ma_n$ is tested in turn, and the first valid operation is retained.

4.1.4. *Node relocation.* This last operator is mainly devoted to improving the quality of the mesh. A vertex $p \in \mathcal{T}$ is moved to a new position $\widetilde{p}$ so that the quality of the local configuration results improved. Computing the position of $\widetilde{p}$ follows a different heuristic depending on whether p belongs to $\mathcal{S}_\mathcal{T}$ or not:

- If $p \in \mathcal{S}_\mathcal{T}$, the surface ball $\mathcal{B}_S(p)$ is projected onto the tangent plane $T_p\partial\Omega$, and a local parametrization of $\partial\Omega$ by a part of $T_p\partial\Omega$ is generated along the lines of section 3.2. A new position is then computed on $T_p\partial\Omega$ as the center of mass of the projected ball of p onto $T_p\partial\Omega$ (of course, one may think of other choices as for this new position). Finally, the corresponding point $\widetilde{p}$ is taken on $\partial\Omega$.
- If p is not a surface point, the ball $\mathcal{B}(p)$ of p is enumerated, and $\widetilde{p}$ is taken as its center of mass.

In both cases, the resulting configuration of the vertex relocation procedure has to be checked, so that no element ends up invalidated in the process and the quality of the mesh is indeed enhanced.

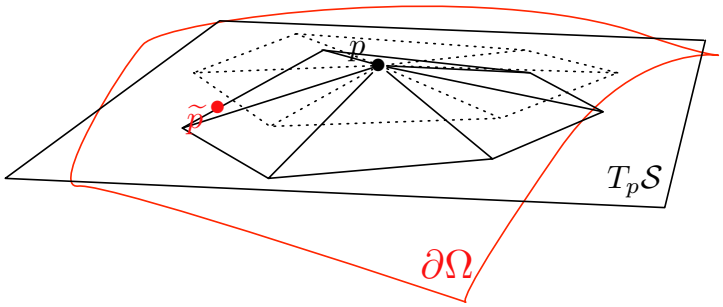


FIGURE 4. Relocation of vertex p: $\mathcal{B}_{\mathcal{S}}(p)$ is projected onto T_pS, and an optimal position is sought on T_pS, then projected onto $\mathcal{S}$ (right).

4.2. **Local size feature.**

At this point, we still lack a global vision to drive our remeshing strategy, that is to identify (and possibly classify) those edges of $\mathcal{T}$ that should be split, collapsed, or swapped. Since [21] [26], a very convenient means to encode such information has been that of a *size function* $h : \overline{\Omega} \to \mathbb{R}$, so that for each $x \in \overline{\Omega}$, $h(x)$ accounts for the local desired size for edges of $\mathcal{T}$ lying around x. The final aim of the process is then to produce a new mesh $\widetilde{\mathcal{T}}$ of Ω, whose edges pq have (as far as possible) unit length $\ell_h(pq)$ with respect to h, that is:

$$\forall\, pq \in \widetilde{\mathcal{T}},\ \ \ell_h(pq) := \int_0^1 \frac{||pq||}{h(p+t(q-p))}\, dt \approx 1.$$

In numerical practice, h is defined and stored at the vertices of $\mathcal{T}$:

- If $x \in \mathcal{S}_{\mathcal{T}}$, the size prescription $h(x)$ in a neighborhood of x stems from a heuristic based on the following theorem (see [11] for a precise statement, and a proof):

  Theorem 1. *Let $\Omega \subset \mathbb{R}^d$ a domain, and $\mathcal{T}$ a mesh, whose associated surface mesh $\mathcal{S}_{\mathcal{T}}$ is 'close' from $\partial\Omega$. Denote as d_Ω the signed distance function to Ω, and as $\mathcal{H}(d_\Omega)$ its Hessian matrix. Then*

  $$d^H(\partial\Omega, \mathcal{S}_{\mathcal{T}}) \leq \frac{1}{2}\left(\frac{d-1}{d}\right)^2 \max_{T\in\mathcal{S}_{\mathcal{T}}} \max_{x\in T} \max_{y,z\in T} \langle |\mathcal{H}(d_\Omega)(x)|yz, yz\rangle.$$

  Since, for all $x \in \partial\Omega$, $\mathcal{H}(d_\Omega)(x)$ is nothing but the second fundamental form $II_x : T_x\partial\Omega \times T_x\partial\Omega \to \mathbb{R}$, this leads us to the choice:

  $$\forall x \in \mathcal{S},\ \ h(x) = \sqrt{\frac{9\varepsilon}{2\max(|\kappa_1(x)|, |\kappa_2(x)|)}},$$

  where $\kappa_1(x), \kappa_2(x)$ are the principal curvatures of $\partial\Omega$ at x. This formula may be truncated according to the minimal and maximal authorized (Euclidean) sizes for edges of $\widetilde{\mathcal{T}}$, h_{min} and h_{max}.
- If $x \notin \mathcal{S}_{\mathcal{T}}$, there is no particular size to impose near x, whence: $h(x) = h_{max}$.

Of course, this is to be coupled with the possible datum of a user-defined size function $m : \overline{\Omega} \to \mathbb{R}$ - which may stem from an error estimate associated to a numerical method performed on $\mathcal{T}$, for instance.

Unfortunately, conforming to such a size prescription is not a sufficient criterion to guarantee the resulting mesh $\widetilde{\mathcal{T}}$ will enjoy a fine mesh quality. As noticed in [5], shocks in size prescriptions between close areas on $\mathcal{T}$ may impose ill-shaped elements to a unit mesh with respect to h. For this reason, it may be desirable to drive our remeshing operators so that two adjacent edges ap, bp in $\widetilde{\mathcal{T}}$ have Euclidean lengths satisfying

(1).To achieve this, we follow the heuristic approach in [31], noticing that any two adjacent edges ap, bp in a unit mesh $\widetilde{\mathcal{T}}$ with respect to h comply with (1) provided h fulfills the following property:

$$\forall \text{ edge } pq \in \mathcal{T}, \ \frac{|h(p) - h(q)|}{||pq||} \leq \log(h_{grad}),$$

and enforce the latter condition by truncating the size function h accordingly.

4.3. **The complete remeshing strategy.**

First and foremost, some general comments or observations based on our experience are in order:

- We thought it better to proceed both the surface and internal parts of $\mathcal{T}$ at the same time. This was motivated by the observation that a dramatically ill-shaped mesh $\mathcal{T}$ associated to an ideal domain $\partial\Omega$ may present a very nicely-shaped surface part $\mathcal{S}_{\mathcal{T}}$.
- On a different note, it turned out that tetrahedra are generally much more prone to degenerating as (surface) triangles, meaning that within very few operations, a well-shaped tetrahedron may end up nearly flat, unless the degeneracy in quality of elements is explicitly controlled and prevented in the course of each operation. For this reason, the operators presented in the previous section appear as more severely constrained as in the context of sole surface remeshing.
- It appeared that the edge swap operation has a significantly different impact between the surface and domain remeshing contexts: in the former case, edge swap helps noticeably to increase the overall quality of the elements of the mesh, but a satisfying overall mesh quality can still be obtained without using it; on the contrary, in the case of domain remeshing, we were never able to reach any good-quality resulting mesh without the (massive) use of edge swap (for surface as for internal edges). See section 4.4 for an illustration of this feature.
- The node relocation operator is not really involved in removing very ill-shaped elements from the mesh as the edge swap operator is. However, its impact on increasing the overall quality of the resulting mesh is substantial.

Now, starting from an initial tetrahedral mesh $\mathcal{T}$, and given the four parameters ε, h_{min}, h_{max} and h_{grad} introduced above, the proposed remeshing strategy rolls out as follows:

step 1: *Analysis of the surface mesh $\mathcal{S}_{\mathcal{T}}$.* Additional information about $\partial\Omega$ are inferred from $\mathcal{S}_{\mathcal{T}}$, along the lines of section 3.1.

step 2: *Rough mesh modifications for a good 'sampling' of the surface.* This first real stage of mesh modifications aims at producing a new mesh $\widetilde{\mathcal{T}_1}$ of Ω which is a nice geometric approximation, with respect to the authorized tolerance: $d^H(\mathcal{S}_{\widetilde{\mathcal{T}_1}}, \partial\Omega) \leq \varepsilon$. Starting from $\mathcal{T}$, we proceed within several (typically, five, six) iterations of the form:

(1): Identify all the edges of $\mathcal{S}_{\mathcal{T}}$ that should be split - i.e. either they have greater length than h_{max}, or the Hausdorff distance between them and the associated curves on $\mathcal{S}$ is higher than ε - then split them (using patterns).

(2): Identify all the internal edges of the mesh that should be split, then split them (using patterns).

(3): Travel all the edges of the mesh, and collapse all the ones that should, and can be collapsed.

(4): Travel all the edges of the mesh, and swap all the ones that should, and can be swapped.

Note that the splitting operation has been divided into two steps : the first one processes surface edges, while the second one only concerns internal ones. This is actually nothing but a technicality aimed at limiting the number of splitting patterns (thus the programming effort), and it does not challenge the commitment to treat the surface and internal parts of the mesh at the same time.

step 3: *Construction of the size function.* Although it may still be of poor quality, $\widetilde{\mathcal{T}_1}$ accounts for a suitable geometric approximation of Ω. It is then relevant to compute the size function $h : \Omega \to \mathbb{R}$ which relies on higher-order features of $\partial\Omega$ (see section 4.2).

step 4: *Rough mesh modifications with respect to the size function.* We proceed almost exactly as in step (2) so as to get the next mesh $\widetilde{\mathcal{T}_2}$, except on two aspects: first, we rely now on lengths measured with respect to h. More specifically, aiming at getting a new triangulation whose edges have length 1 with respect to h (which is, of course, impossible), we choose *rough* bounds $\ell_{r,min}, \ell_{r,max}$ outside which

no edge length should lie (typically, we used $\ell_{r,min} = 0.3$, $\ell_{r,max} = 2.5$). Second, we take much more caution about mesh quality as during step (2) in the control of the remeshing operators.

step 5: *Fine mesh modifications with respect to the size function.* Mesh $\widetilde{\mathcal{T}}_2$ should be 'almost good' in terms of geometric approximation of Ω and of mesh quality. We perform delicately driven operations so as to get the final mesh $\widetilde{\mathcal{T}}$. Lengths of edges are still evaluated with respect to h, except we now impose $\widetilde{\mathcal{T}}$ have no edge with length lying outside a sharper interval as before, of the form $[\ell_{f,min}, \ell_{f,max}]$(we used $\ell_{f,min} = 0.7$, $\ell_{f,max} = 1.3$). We are also even stricter as in step (4) as far as the authorized degradation in mesh quality is concerned. Another important improvement with respect to steps (2) and (4) is that we now add the vertex relocation operator to our toolbox. The iterations performed during steps (2) and (4) evolve into :

(1) Travel all the edges of the mesh, and split (now in a one-by-one fashion) all the ones that should, and can be split.
(2) Travel all the edges of the mesh, and collapse all the ones that should, and can be collapsed.
(3) Travel all the edges of the mesh, and swap all the ones that should, and can be swapped.
(4) Travel all the vertices of the mesh, and relocate all the ones that should, and can be relocated.

4.4. **Numerical examples.**

As a support to the previous developments, let us give two numerical illustrations of discrete domain remeshing.

4.4.1. *High-quality remeshing of smooth domains or mechanical parts.* Figure 5 provides a first view of the behavior of the proposed strategy for discrete domain remeshing. In this example, a very ill-shaped three-dimensional mesh is obtained from an initial STL surface triangulation in such a way as the resulting meshes have almost no interior point. This initial mesh is then remeshed into a well-shaped mesh resorting to the proposed approach. The initial mesh enjoys $1,254$ vertices; the whole computation took a few seconds, and the final mesh enjoys $8,424$ vertices. The average quality of the initial mesh is 0.0479 (the worst quality for an element is $1.e^{-6}$), whereas the average quality of the final mesh is 0.7737 (the worst quality of an element being 0.19).

In order to emphasize the huge impact of the edge swap operator in our remeshing process, we performed exactly the same test, without using edge swap. In this case, the proposed algorithm turns out unable to produce a well-shaped mesh (mainly because the collapse operator is too much constrained by very ill-shaped configurations): after a computation which lasts 100.65s, the final mesh has $46,222$ vertices. Its average quality is only 0.07, and the worst quality of an element of the mesh is $2.e^{-6}$.

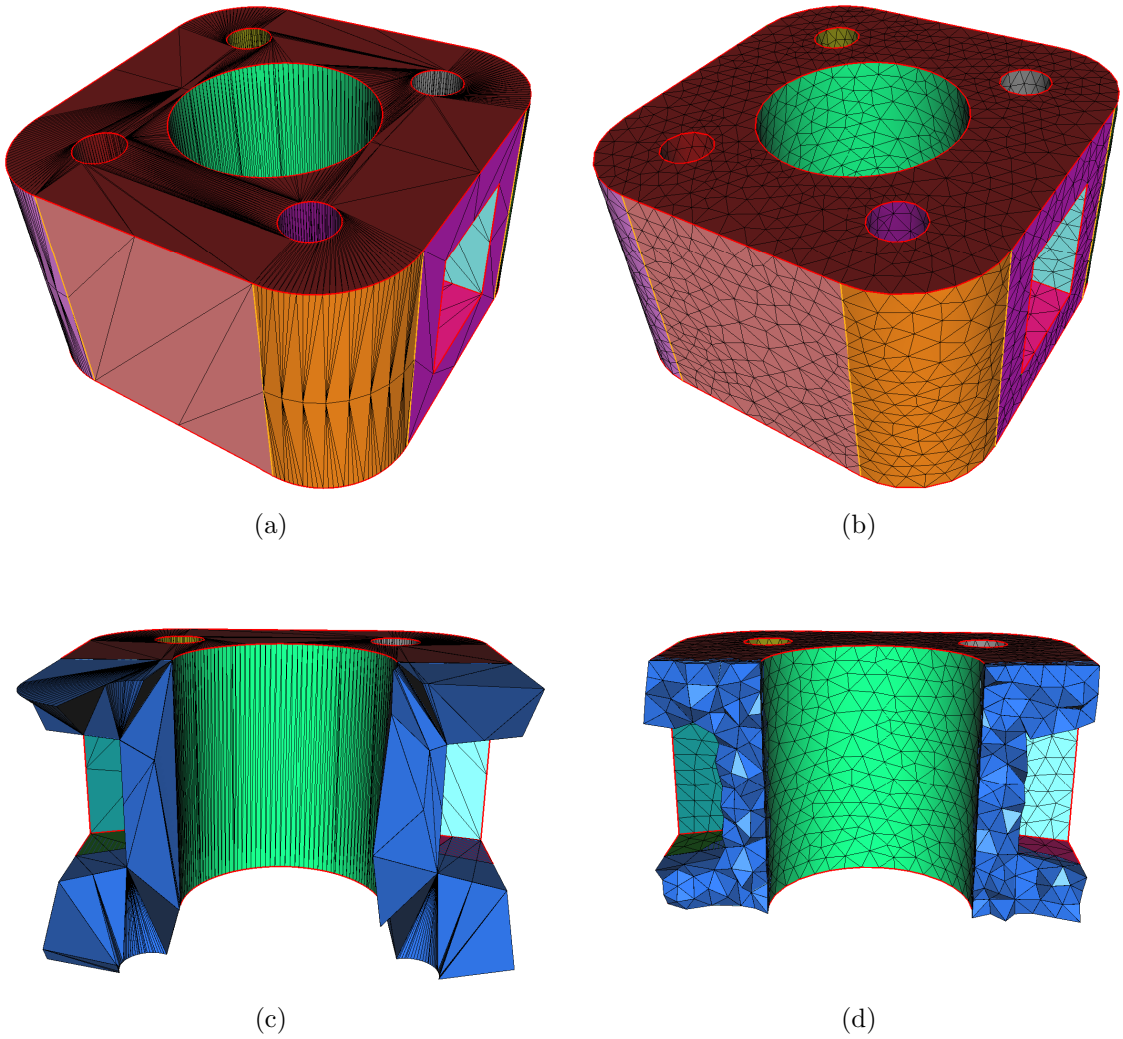
4.4.2. *Remeshing of a domain with respect to a user-defined size map.* We investigate the datum of a user-specified size map $m : \overline{\Omega} \to \mathbb{R}$ on the considered domain Ω. More specifically, once again, we dwell on the case when m is associated to the interpolation error of a smooth function.

Let $f : \overline{\Omega} \to \mathbb{R}$ a function of class $\mathcal{C}^2$, which presents 'sharp variations'. For instance,

$$\forall p = (x, y, z) \in \overline{\Omega}, \;\; f(p) = \tanh(y^3)\, \tanh((z-2)^3). \tag{4}$$

We aim at modifying the initial mesh $\mathcal{T}$ so that interpolating (linearly) f over the resulting mesh $\widetilde{\mathcal{T}}$ entails an L^∞-error controlled by a parameter ε_m, and rely on theorem 2 in [4] to cook an associated size function $m : \overline{\Omega} \to \mathbb{R}$ to such a control.

Figure 6 displays the resulting mesh after 4 remeshing procedures for adaptation to the interpolation of f - starting from a rather coarse triangulation $\mathcal{T}$ of the domain Ω depicted on the figure, with parameter $\varepsilon_m = 0.01$. The parameters chosen for remeshing are (for each one of the 4 steps) : $h_{min} = 0.2$, $h_{max} = 3$, $h_{grad} = 1.4$, and $\varepsilon = 0.3$. The whole computation took 2 minutes and 46s, and the final mesh enjoys 73701 vertices, for an average quality of 0.77.



(a) (b)

(c) (d)

FIGURE 5. Remeshing of the *model02* model, enclosed in a box of dimensions $0.199 \times 0.199 \times 0.12$. The parameters used for the computation are: $h_{min} = 0.001$, $h_{max} = 0.1$, $h_{grad} = 1.2$ and $\varepsilon = 0.001$. (a)-(c) Initial mesh $\mathcal{T}$; (b)-(d) final result $\widetilde{\mathcal{T}}$.

5. Meshing of implicitly-defined domains

Hitherto, we focused on remeshing domains $\Omega \subset \mathbb{R}^3$ supplied by means of an initial tetrahedral mesh. Here we adopt a slightly different perspective, and investigate into the case when Ω is described as the negative subdomain of a scalar function defined over the whole space.

More specifically, let $\Omega \subset \mathbb{R}^3$ a bounded smooth enough 'ideal' domain. Ω is known through the datum of an associated smooth enough implicit function $\phi : \mathbb{R}^3 \to \mathbb{R}$, i.e. the following relations hold:

$$\forall x \in \mathbb{R}^3, \quad \begin{cases} \phi(x) < 0 & \text{if} \quad x \in \Omega \\ \phi(x) = 0 & \text{if} \quad x \in \partial\Omega \\ \phi(x) > 0 & \text{if} \quad x \in^c \overline{\Omega} \end{cases} \quad ; \ \forall x \in \partial\Omega, \ \nabla\phi(x) \neq 0. \tag{5}$$

In numerical practice, we are given a mesh $\mathcal{T}$ of a 'big' computational domain $D \subset \mathbb{R}^3$ (e.g. a box), which is the support of a numerical approximation $\phi_{\mathcal{T}}$ of ϕ. For the sake of simplicity, in this paper we assume

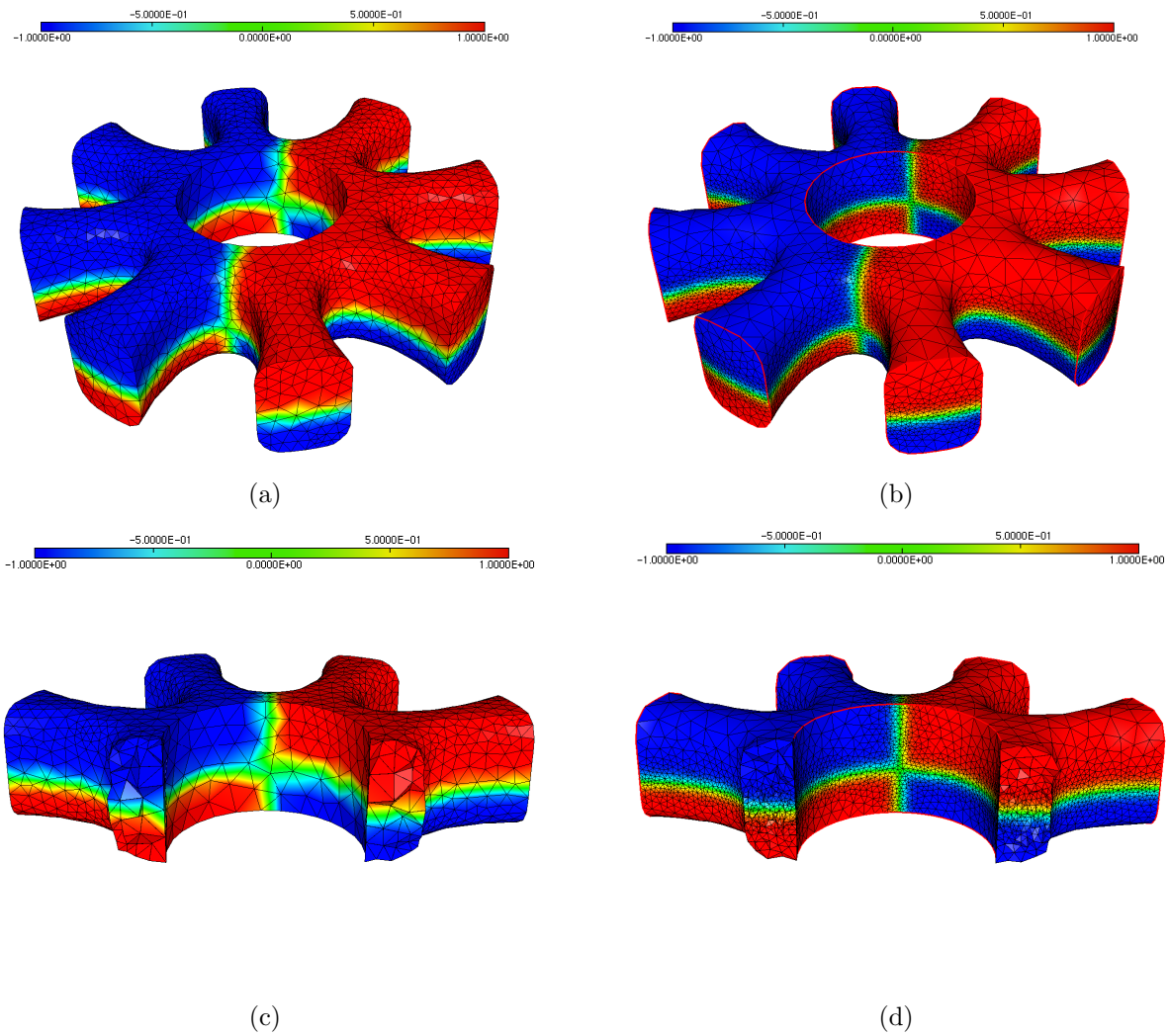


FIGURE 6. Remeshing with respect to a size map devised for the control of the interpolation error of function f given by (4); (a) The initial mesh $\mathcal{T}$, together with a color map associated to the values of f, (c) a cut in $\mathcal{T}$; (b) the resulting adapted mesh $\widetilde{\mathcal{T}}$, carrying function f, (d) a cut in $\widetilde{\mathcal{T}}$.

that $\phi_{\mathcal{T}}$ is a $\mathbb{P}^1$ Lagrange finite element function over $\mathcal{T}$, i.e. for each $K \in \mathcal{T}$, the restriction $\phi_{\mathcal{T}}|_K$ is affine.

Our problem is now to obtain from $\mathcal{T}$ and $\phi_{\mathcal{T}}$ a well-shaped mesh of Ω - more accurately of $D \cap \Omega$. The method proposed to do so is a rather straightforward extension of the remeshing algorithm described in section 4, up to the addition of one - possibly two - ingredients. Indeed, it proceeds within two main steps (see figure 7 for an illustration in two space dimensions):

(1) The 0 level set of $\phi_{\mathcal{T}}$ - say $\mathcal{S}_{\phi_{\mathcal{T}}} := \{x \in D \,|\, \phi_{\mathcal{T}}(x) = 0\}$ - is explicitly discretized in $\mathcal{T}$. A new mesh $\widetilde{\mathcal{T}}_1$ of D is obtained, which contains a mesh $\mathcal{T}_1\prime$ of $D \cap \Omega$ as a submesh.
(2) Mesh $\widetilde{\mathcal{T}}_1$ is modified, so that a new, closely approximating, well-shaped mesh $\widetilde{\mathcal{T}}$ of D is obtained, which contains a closely approximating, well-shaped mesh $\mathcal{T}\prime$ of $D \cap \Omega$ as a submesh.

Hence, this method produces a bit more than a sole mesh of $D \cap \Omega$, namely a new mesh $\widetilde{\mathcal{T}}$ of the whole computational domain D, a submesh $\mathcal{T}\prime$ of which is a mesh of $D \cap \Omega$. This will come in handy in section 7.

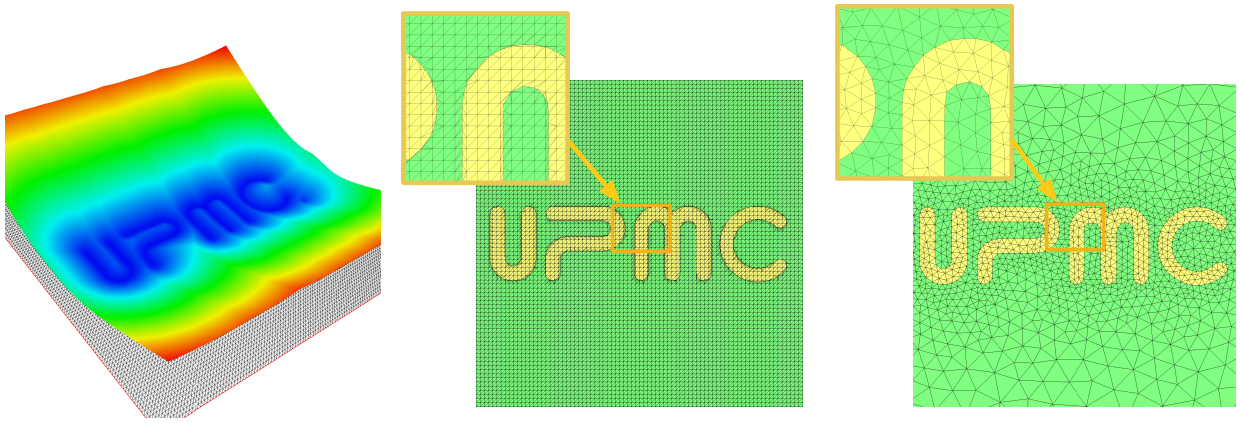


FIGURE 7. (left) Level sets of a $\mathbb{P}^1$ function $\phi_{\mathcal{T}}$ on a two-dimension mesh $\mathcal{T}$ of a box D ; (middle) the ill-shaped mesh $\widetilde{\mathcal{T}}_1$, obtained by the explicit discretization of the 0 level set of $\phi_{\mathcal{T}}$ into $\mathcal{T}$; (right) the final, well-shaped mesh $\widetilde{\mathcal{T}}$, containing a mesh of Ω (in yellow) as a submesh.

5.1. Explicit discretization of the 0 level set of $\phi_{\mathcal{T}}$ into $\mathcal{T}$.

The first step in constructing a suitable mesh of $D \cap \Omega$ boils down to enforcing an explicit discretization of $D \cap \Omega$. This is achieved through the following *marching tetrahedra* procedure, which is a well-known variation of the *marching cubes* algorithm [32]:

(1) Identify the set $\mathcal{K}$ of elements $K \in \mathcal{T}$ intersecting $\mathcal{S}_{\phi_{\mathcal{T}}}$: a tetrahedron $K = a_0a_1a_2a_3$ belong to $\mathcal{K}$ if and only if there exists $i \neq j$ in $\{0, ..., 3\}$ with $\phi_{\mathcal{T}}(a_i) \geq 0$, and $\phi_{\mathcal{T}}(a_j) \leq 0$.
(2) For an element $K = a_0a_1a_2a_3 \in \mathcal{K}$, the intersection of $\mathcal{S}_{\phi_{\mathcal{T}}} \cap K$ is a plane portion of surface. Identify the edges a_ia_j of K which intersect $\mathcal{S}_{\phi_{\mathcal{T}}}$ (i.e. such that $\phi_{\mathcal{T}}(a_i)$ and $\phi_{\mathcal{T}}(a_j)$ have different signs), and compute the coordinates of the associated intersection points m_{a_i,a_j}.
(3) Travel all the elements $K \in \mathcal{K}$, and split them, introducing the pre-computed points m_{a_i,a_j}, then using patterns (see figure 8). Up to permutations, there are four possible configurations, depending on the relative signs of the $\phi_{\mathcal{T}}(a_i)$.

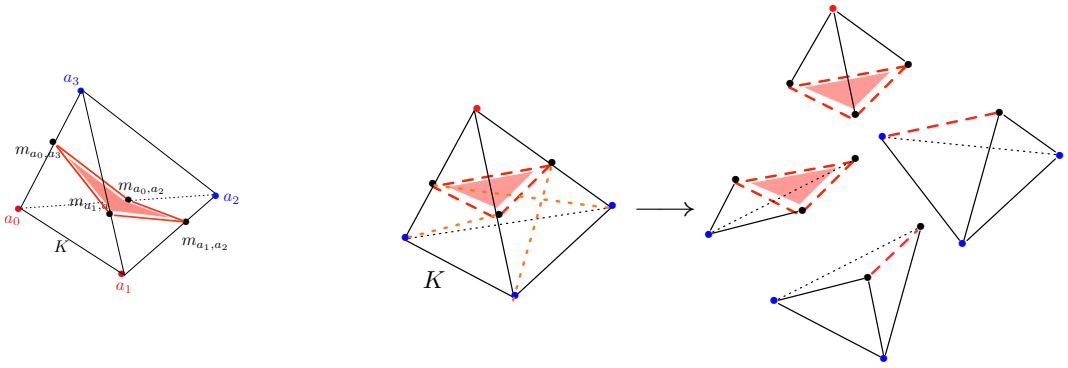


FIGURE 8. (left) One of the possible situations when $\mathcal{S}_{\phi_{\mathcal{T}}}$ (in light red) crosses an element $K \in \mathcal{T}$; (right) example of a splitting pattern for a tetrahedron $K \in \mathcal{T}$ which is crossed by $\mathcal{S}_{\phi_{\mathcal{T}}}$ in such a way as three of its vertices share the same sign (the blue ones).

This procedure delivers a new mesh $\widetilde{\mathcal{T}}_1$ of D, which is most likely very ill-shaped; yet, it is conforming and contains a mesh $\mathcal{T}_1\prime$ of $D \cap \Omega$ as a submesh.

5.2. **From discrete domain remeshing to discrete domain and subdomains remeshing.**

We now have a mesh $\widetilde{\mathcal{T}_1}$ of D, a submesh $\mathcal{T}_1\prime$ of which accounts for $D \cap \Omega$. Let us denote as $\mathcal{S}_{\widetilde{\mathcal{T}_1}}$ the associated *total* surface mesh, that is,

$$\mathcal{S}_{\widetilde{\mathcal{T}_1}} = \mathcal{E}_{\widetilde{\mathcal{T}_1}} \cup \mathcal{I}_{\widetilde{\mathcal{T}_1}},$$

where $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ stands for the collection of all the (triangular) external faces to the elements of $\widetilde{\mathcal{T}_1}$, and $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ is the set of the external faces of the elements of $\mathcal{T}_1\prime$ *which do not already belong to* $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$. In other terms, $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ is the surface mesh associated to ∂D, and $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ is the surface mesh associated to $\partial(D\cap\Omega)\backslash\partial D$ (see figure 9).

Our purpose is to remesh $\widetilde{\mathcal{T}_1}$ into a new, high-quality mesh $\widetilde{\mathcal{T}}$ of D, a submesh $\mathcal{T}\prime$ of which accounts for $D\cap\Omega$, and such that the associated total surface mesh $\mathcal{S}_{\widetilde{\mathcal{T}}}$ is a close approximation of $\partial D\cup\partial(D\cap\Omega)$, within a prescribed range ε in terms of Hausdorff distance.

This problem is very close to the problem of discrete domain remeshing, as presented in section 4. Suppose for now that the two surfaces ∂D and $\partial\Omega$ at stake are disjoint (figure 9, left). Then, the faces of $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ and $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ can be processed *independently*, in the same way as described in section 4. Admittedly, the entities belonging to $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ are *a priori* more severely constrained compared to those of $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$, since they are connected to more elements of $\widetilde{\mathcal{T}_1}$, and the several checks to be performed before applying our local remeshing operators to ensure the validity of the resulting mesh are more likely to fail. Yet, those checks read just the same in both cases.

The only real change arises when ∂D and $\partial\Omega$ do intersect one another. Then, the intersection $\mathcal{E}_{\widetilde{\mathcal{T}_1}} \cap \mathcal{I}_{\widetilde{\mathcal{T}_1}}$ is a collection of curves (figure 9, right). In such a situation, both surface meshes $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$, $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ considered separately are orientable ; yet, their reunion is not. To deal with this particular configuration, a new category of edges and vertices has to be added to the classification of section 3.1, namely that of *non manifold entities*: a non manifold edge (resp. vertex) of $\widetilde{\mathcal{T}_1}$ belongs to at least one triangle of $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$, and to another triangle of $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$. The description adopted for non manifold edges or vertices is very similar to that of ridge edges or vertices: a non singular, non manifold point x is equipped with a tangent vector $\tau(x)$ to this curve, and a different normal vector is used, depending on whether the point is processed as belonging to $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ or to $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$.

Apart from this minor change, such configurations raise no further difficulty, and can be tackled in the very same way as in the case when $\partial D \cap \partial(D \cap \Omega) = \emptyset$.

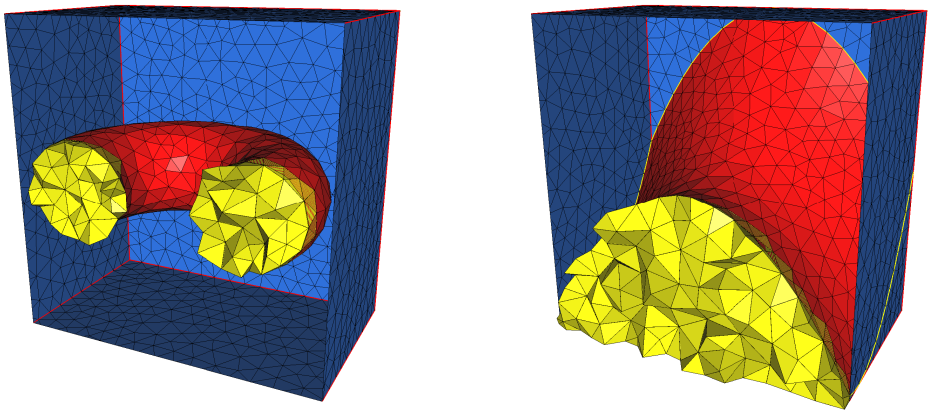


FIGURE 9. Two examples of (well-shaped, constant size) meshes of a box D, enclosing a mesh of a subdomain $D\cap\Omega$ as a submesh. In both case, the triangular faces of $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ appear in blue, and those of $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$ in red. Only the tetrahedra of $D\cap\Omega$ have been displayed (in yellow). On the left-side example, ∂D and $D\cap\Omega$ are disjoint, whereas they are not on the right-side one (the yellow edges are those edges which belong to both $\mathcal{E}_{\widetilde{\mathcal{T}_1}}$ and $\mathcal{I}_{\widetilde{\mathcal{T}_1}}$).

6. Application to mesh generation from a possibly invalid surface triangulation

In this section, we present a promising application of the previous implicit domain meshing algorithm in combination to three-dimensional mesh generation.

Let $\mathcal{S} = (T_i)_{i=1,\dots,N_{\mathcal{S}}}$ a surface triangulation of the boundary $\partial\Omega$ of an open bounded domain $\Omega \subset \mathbb{R}^3$. The classical problem of three-dimensional mesh generation consists in constructing a tetrahedral mesh of Ω, whose associated surface mesh is exactly $\mathcal{S}$. This problem is very hard in general, and very few algorithms exist that are sufficiently polyvalent and robust to deal with general enough triangulations $\mathcal{S}$.

Here, we propose a different approach, which inherently requires to drop the constraint that the initial boundary triangulation $\mathcal{S}$ should be retained through the process.

As a first stage, the initial surface triangulation $\mathcal{S}$ is embedded in a big computational domain D, equipped with a simplicial mesh $\mathcal{K}$. Choosing an arbitrary tolerance parameter $\varepsilon > 0$, the method described in [12] for computing the signed distance function to a triangulated contour and adapting the computational mesh to this function can be used to produce simultaneously:

- a new (anisotropic) mesh $\widetilde{\mathcal{K}_1}$ of D,
- an approximation of the signed distance function d_Ω to Ω as a $\mathbb{P}^1$ Lagrange finite element function $d_{\widetilde{\mathcal{K}_1}}$ on $\widetilde{\mathcal{K}_1}$, which enjoys the following property: the Hausdorff distance $d^H(\mathcal{S}_{d_{\widetilde{\mathcal{K}_1}}}, \partial\Omega)$ between $\partial\Omega$ and the piecewise affine reconstructed surface $\mathcal{S}_{d_{\widetilde{\mathcal{K}_1}}} := \left\{ x \in D \mid d_{\widetilde{\mathcal{K}_1}}(x) = 0 \right\}$ is no larger than ε.

Note that the choice of an anisotropic mesh $\widetilde{\mathcal{K}_1}$ as a support of an approximation of d_Ω stems from the concern to guarantee an accurate representation of $\partial\Omega$, using a mesh whose size is moderate.

In a second stage, the negative subdomain of $d_{\widetilde{\mathcal{K}_1}}$ is meshed, using the method presented in section 5, with the mesh $\widetilde{\mathcal{K}_1}$ of D: a new mesh $\widetilde{\mathcal{K}}$ of D is produced, which encloses a mesh $\mathcal{T}$ of Ω as a submesh.

This procedure is applied to the *Venus* model displayed on figure 10. Note that the initial surface triangulation is actually self-intersecting, and non orientable - thus no three-dimensional mesh can enjoy it as surface triangulation. The total remeshing time, from the datum of the mesh $\widetilde{\mathcal{K}_1}$ of a unit box D and the approximation $d_{\widetilde{\mathcal{K}_1}}$ to d_Ω is 10 min and 9 s, and the parameters of the computation are: $h_{min} = 0.001$, $h_{max} = 0.1$, $h_{grad} = 1.2$, $\varepsilon = 0.001$, for a final mesh $\widetilde{\mathcal{K}}$ enjoying an average quality of 0.77.

Remark 1. *Looking carefully at the result displayed on figure 10, the obtained mesh $\mathcal{T}$ of $\partial\Omega$ appears far from being completely satisfactory in terms of the accuracy of the approximation of $\partial\Omega$ by means of the associated surface mesh $\mathcal{S}_{\mathcal{T}}$. Actually, such inaccuracies are concentrated in those regions of $\partial\Omega$ where ridge edges, or singular points are present ; the main reason is that, in those areas, the signed distance function d_Ω is inaccurately computed, whatever the size of the computational mesh $\widetilde{\mathcal{K}_1}$, because the corresponding ridge edges (or singular points) do not explicitly appear in $\widetilde{\mathcal{K}_1}$. A way to deal with this problem would be to discretize those edges, or points into $\widetilde{\mathcal{K}_1}$, before computing an approximation $d_{\widetilde{\mathcal{K}_1}}$ to d_Ω on it. Thus, $d_{\widetilde{\mathcal{K}_1}}$ would amount to 0 on those edges (or points), and they would appear (up to a geometric equivalent) into $\mathcal{T}$. This latter part is an ongoing work.*

7. Application to free and moving boundary problems

We now propose a combination of the level set method with the previous algorithm for meshing implicitly-defined domains into a general framework for dealing with free and moving boundary problems, which brings into play an exact description of the considered shapes (i.e. with a mesh), while benefiting from the versatility of the level set method. For the sake of clarity, this approach is described in the context of a single evolving domain Ω among a computational box D; however, its adaptation to the case of an evolving interface Γ between two subdomains Ω^0 and $\Omega^1 := D \setminus \Omega^0$ follows readily.

7.1. **A general algorithm for free and moving boundary problems.**

Since the seminal paper [30], the level set method has been one method of choice for interface-capturing. Roughly speaking, it consists in trading the usual representation of a domain $\Omega \subset \mathbb{R}^d$ whose evolution is at

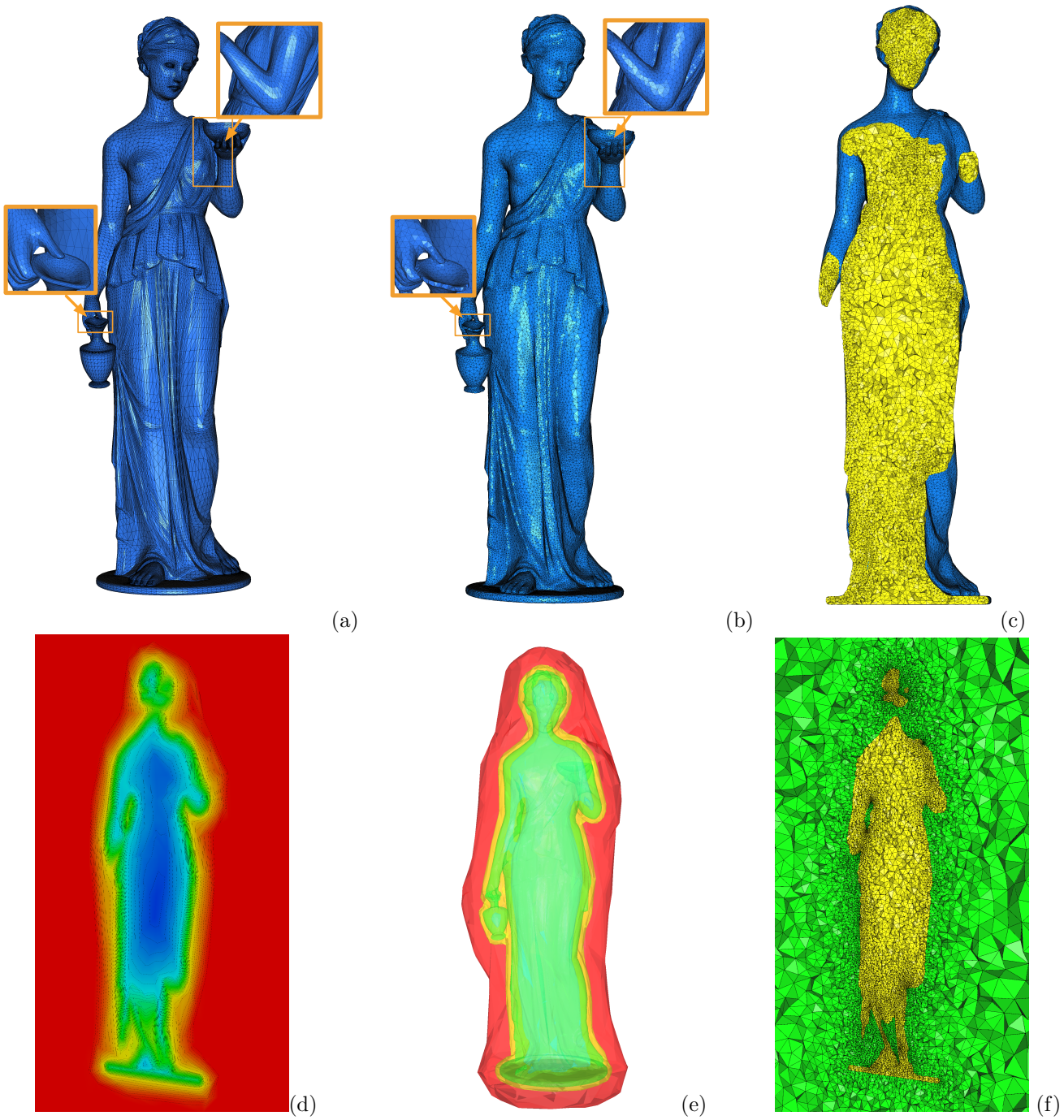


FIGURE 10. Meshing of the *Venus* model: (a) the initial surface triangulation $\mathcal{S}$, (b-c) the final three-dimensional mesh $\mathcal{T}$, (d) some isolines of the approximation $d_{\widetilde{\mathcal{K}_1}}$ of d_Ω in a cut plane of the computational mesh $\widetilde{\mathcal{K}_1}$ (around 410000 vertices), (e) some isosurfaces of $d_{\widetilde{\mathcal{K}_1}}$, (f) a cut in the final mesh $\widetilde{\mathcal{K}}$ of D, which encloses $\mathcal{T}$ as a submesh, in yellow (around 375000 vertices).

stake for an implicit representation by means of an auxiliary scalar function $\phi : \mathbb{R}^d \to \mathbb{R}$ defined so that (5) holds.

Let $\Omega(t) \subset \mathbb{R}^d$ an evolving domain, $\phi : [0,T] \times \mathbb{R}^d \ni (t,x) \mapsto \phi(t,x) \in \mathbb{R}$ an associated level set function. The evolution of $\Omega(t)$ is assumed to be dictated by a velocity field $V : [0,T] \times \mathbb{R}^d \ni (t,x) \mapsto f(t,x,\Omega(t)) \in \mathbb{R}^d$, for a given function f. The latter function may bring into play local (e.g. its mean curvature), or global features (e.g. as a combination of solutions of PDE posed on Ω); see [37] for examples. In a region $\mathcal{O} \subset [0,T] \times \mathbb{R}^d$ where ϕ is smooth enough and V is well-defined and smooth enough, a simple use of the chain-rule yields the so-called *level set advection equation*:

$$\forall (t,x) \in \mathcal{O}, \ \frac{\partial \phi}{\partial t}(t,x) + V(t,x).\nabla \phi(t,x) = 0. \tag{6}$$

In many applications, the velocity field V happens to be directed along the normal direction to the interface (or, more accurately to the level sets of ϕ), that is $\forall x, \ V(t,x) = v(t,x)\frac{\nabla \phi(t,x)}{||\nabla \phi(t,x)||}$, for some scalar field $v(t,x)$. Equation (6) is then better rewritten as a *Hamilton-Jacobi equation*:

$$\forall (t,x) \in \mathcal{O}, \ \frac{\partial \phi}{\partial t}(t,x) + v(t,x)||\nabla \phi(t,x)|| = 0. \tag{7}$$

This straightforward analysis is formal, for it can only be applied in areas where ϕ and V stay smooth enough. Actually, it is well-known that even domains evolving according to very simple vector fields may develop singularities in finite time, and the way to take into account such singularities is non-trivial and case-dependent. Most of the time, it requires more information about the physics of the evolution process, and equations (6,7) have to be understood in a weaker sense - see [6] [17] for illustrations in particular cases.

These considerations go far beyond the scope of this paper; towards simulating realistic models, where the dependence of f on Ω may be very general (we have in mind the case when computing the velocity field requires to solve one, or several PDE on Ω), we resort to the classical and heuristic 'quasi-static' approach.

The main idea of the proposed approach is to go back and forth between two complementary descriptions of domains. Let D a large computational domain in which all the considered domains are included, and $\Omega \subset D$; we will either represent Ω as:

- the negative subdomain of an associated level set function ϕ, numerically discretized on a tetrahedral mesh of D - this is the suitable description when it comes to accounting for domain evolution.
- a mesh $\mathcal{T}_\Omega$ of the whole domain D, a part of which is a mesh of Ω (i.e. a mesh of Ω is embedded in a mesh of D in a conforming way); this is the suitable description when it comes to solving mechanical problems on Ω.

More accurately, the interval $[0,T]$ is split into subintervals $[t_n, t_{n+1}]$, where the time step $\tau^n := t^{n+1} - t^n$ may change from one iteration to the other, and the evolution of $\Omega(t)$ through time is numerically achieved by the sequence $(\Omega^n)_{n=0,...,N}$ of discrete domains given by the following process:

For $n = 0, ..., N-1$, start with a shape Ω^n, given under the form of a mesh $\mathcal{T}_{\Omega^n}$ of D, a submesh $\mathcal{T}_{\Omega^n}{}'$ of which is a mesh of Ω^n.

(1) Generate the signed distance function d_{Ω^n} to Ω^n on the whole mesh $\mathcal{T}_{\Omega^n}$ of D. This requires an algorithm for computing the signed distance function on an unstructured computational mesh (see [12] for a possible approach).

(2) Retain only the part $\mathcal{T}_{\Omega^n}{}'$ of $\mathcal{T}_{\Omega^n}$ corresponding to Ω^n, and compute the velocity field $V^n(x) = f(t^n, x, \Omega^n)$. This step is the only one depending on the specificities of the considered domain evolution problem. See sections 7.2, 7.3 for two examples from mechanics.

(3) Solve the *level set advection equation* with velocity field V^n over the whole mesh $\mathcal{T}_{\Omega^n}$, and the period of time $[t^n, t^{n+1}]$:

$$\left\{ \begin{array}{cc} \frac{\partial \phi}{\partial t}(t,x) + V^n(x).\nabla \phi(t,x) = 0 & \text{for } x \in D, \ t \in (t^n, t^{n+1}), \\ \phi(0,x) = d_{\Omega^n}(x) & \text{for } x \in D, \end{array} \right. . \tag{8}$$

This requires an algorithm for solving the advection equation on an unstructured computational mesh - see for instance [36] or [39] for approaches based on the *method of characteristics*. This step produces a level set function $\phi^{n+1} := \phi(\tau^n, .)$ on $\mathcal{T}_{\Omega^n}$ associated to the new shape Ω^{n+1}.

(4) Discretize the 0 level set of ϕ^{n+1} in the mesh $\mathcal{T}_{\Omega^n}$ in the spirit of section 5 to obtain a new mesh $\mathcal{T}_{\Omega^{n+1}}$ of D, in which Ω^{n+1} is explicitly discretized.

An important feature of this algorithm is that, at each step $t^n \to t^{n+1}$, the (mechanical) application-dependent computation of the velocity field V^n and the advection process (8) are carried out on the same mesh $\mathcal{T}_{\Omega^n}$. Hence, no projection whatsoever is involved between different computational meshes, suited for different purposes.

7.2. **An application in shape optimization.**

Elaborating on the pioneering work [2] which initiated the idea of coupling shape optimization with the level set method, we focus on the use of the above algorithm in this situation. This part echoes to our previous two-dimensional work [1].

Let us briefly sketch the context - see [33] or [27] for exhaustive presentations. We are interested in *shapes*, that is bounded domains in $\mathbb{R}^d$, filled with a linear elastic material with Hooke's law A (e.g. mechanical structures). The considered shapes are clamped on a part $\Gamma_D \subset \partial\Omega$ of their boundary, and submitted to surface loads $g \in H^2(\mathbb{R}^d)^d$ on another part $\Gamma_N \subset \partial\Omega$ (we omit body forces for simplicity). Both parts Γ_D and Γ_N are not subject to optimization. The displacement field $u_\Omega \in H^1(\Omega)^d$ of a shape Ω is the unique solution to the linear elasticity system:

$$\left\{\begin{array}{cc} -\mathrm{div}\left(Ae(u)\right) = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_D \\ Ae(u).n = g & \text{on } \Gamma_N \\ Ae(u).n = 0 & \text{on } \Gamma \end{array}\right. ,$$

where $e(u) := \frac{\nabla u + {}^t\nabla u}{2}$ is the strain tensor, $\Gamma := \partial\Omega \setminus (\Gamma_D \cup \Gamma_N)$ is the *free boundary*, and n is the unit normal vector field to Ω, pointing outward.

From an initial design Ω_0, our goal is to find an optimal shape, with respect to a certain function $J(\Omega)$ of the domain. As for $J(\Omega)$, the only example we will consider is that of an aggregated sum of the compliance and the volume of the structure, namely:

$$J(\Omega) = \int_{\Gamma_N} g.u_\Omega \, ds + \ell \int_\Omega dx, \tag{9}$$

where ℓ is interpreted as a fixed Lagrange multiplier associated to a volume constraint.

When it comes to accounting for the sensitivity analysis of J with respect to the domain, we rely on *Hadamard's boundary variation method*, and consider variations of a shape Ω of the form $(I+\theta)(\Omega)$, where $\theta \in W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d)$, $||\theta||_{W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)} < 1$. A functional $F(\Omega)$ of the domain is then said to be *shape-differentiable* at Ω provided the underlying application $W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d) \ni \theta \mapsto F((I+\theta)(\Omega)) \in \mathbb{R}$ is Fréchet-differentiable at 0.

In the above context, one can prove [2] that the functional J given by (9) is shape differentiable at any smooth enough shape Ω, and its shape derivative reads:

$$\forall \theta \in W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d), \;\; dJ(\Omega)(\theta) = \int_\Gamma \left(\ell - Ae(u_\Omega) : e(u_\Omega)\right)(\theta.n)\, ds. \tag{10}$$

From (10), a descent direction for J is easily revealed: displacing Ω according to the vector field

$$\theta = \tau \left(Ae(u_\Omega) : e(u_\Omega) - \ell\right) n$$

for a small enough fictitious time $\tau > 0$ will yield a new shape with better performance with respect to J.

We may now put this notion of shape derivative in the general framework of section 7.1, that is, use the above algorithm with the velocity field given by:

$$f(t,x,\Omega) = \left((Ae(u_\Omega) : e(u_\Omega))\,(x) - \ell \right) n(x) \tag{11}$$

to produce an algorithm for optimizing the shape of an initial domain Ω_0 with respect to J. Note that we have been a bit elusive in writing (11) as such, for the right-hand side only stands for points $x \in \partial\Omega$. This expression should actually be extended to the whole space (a tubular neighborhood of $\partial\Omega$ is actually enough in numerical practice); the way such an extension should be performed is a topic on its own in shape optimization, and we limit ourselves to referring to [25].

The proposed method is appraised on the so-called *optimal mast* test case, as depicted on figure 11: a structure, embedded in a T-shaped box D of height 126, and width 40 at the bottom, 80 at the top, made of an isotropic elastic material of Young modulus $E = 1$ and Poisson ratio $\nu = 0.3$, is clamped on its base, and submitted to unit vertical loads $g = -e_z$ concentrated on the left and right arms. We minimize the objective function (9) with a Lagrange multiplier $\ell = 20$ for the volume constraint. We run 50 iterations of the above algorithm; each mesh $\mathcal{T}_{\Omega^n}$ (of the whole box) has about $15,000$ vertices, and the entire computation takes roughly 40 min. on a laptop computer. The decrease in the objective function in the course of the optimization process is displayed on figure 12. A noticeable feature of the presented computation is that the topology of the evolving shape has dramatically changed during the process, which is an inherent (and natural) achievement of the level set description of the shape evolution.

7.3. **An application in computational fluid dynamics.**

We eventually look into the numerical simulation of bifluid flows, involving interfaces characterized by large jumps of viscosity and density between the fluids that must be properly taken into account and resolved - e.g. relying on the level set method [41]. Elaborating on the previous work [8], we propose to account for the evolution of the interface between the different fluids relying on the general framework of section 7.1.

As a model problem, consider two fluids which are confined to an open, bounded computational domain $D \subset \mathbb{R}^d$, each fluid occupying a time-dependent subdomain $\Omega^i(t)$ $(i = 0, 1)$ such that:

$$\overline{\Omega^0}(t) \cup \overline{\Omega^1}(t) = \overline{D} \quad \text{and} \quad \Omega^0(t) \cap \Omega^1(t) = \emptyset \quad \text{with} \quad \Gamma(t) = \partial\Omega^0(t) \cap \partial\Omega^1(t)\,.$$

Towards an approximation of the true nonlinear moving boundary problem, we consider that, at any fixed time t, the flow of both fluids is governed by the quasi-static incompressible Stokes equations [38], which read, in each subdomain $\Omega^i(t)$:

$$\begin{cases} -\mu^i \Delta u^i + \nabla p^i = & \rho^i f^i \\ \qquad\qquad \operatorname{div}(u^i) = & 0\,, \end{cases} \tag{12}$$

where $u^i(x,t)$ is the velocity field of the fluid, $p^i(x,t)$ is the pressure, ρ^i and μ^i are the density and the dynamic viscosity of each fluid and f^i is an external force exerted on the fluid. The transient character of the solution is related to the entangled motions of the two fluids and the interface (the forces exerted on the fluid are assumed to be in a state of dynamic equilibrium as a result of a rapid diffusion of the momentum).

System (12) is completed with several boundary conditions. The surface tension effect is taken into account at the interface and conditions on the continuity of the velocity field u and on the balance of the normal stress with the surface tension across the interface are imposed on $\Gamma(t)$:

$$\begin{cases} u^0 - u^1 = 0 \\ (\sigma(u)^0 - \sigma(u)^1) \cdot n = -\gamma\,\kappa\,n \end{cases} \tag{13}$$

where $\sigma(u) = \mu(\nabla u + {}^t\nabla u) - p\,\mathrm{I}$ stands for the stress tensor, n is the unit normal vector to $\Gamma(t)$, pointing outward $\Omega^0(t)$, $\gamma > 0$ is the (constant) surface tension coefficient, and κ is the signed mean curvature of $\Gamma(t)$. Moreover, some usual Dirichlet, Neumann or mixed boundary conditions are added on $\partial D \setminus \Gamma(t)$.
Eventually, as an initial condition, the position $\Gamma(0) = \Gamma^0$ of the interface at time $t = 0$ is prescribed.

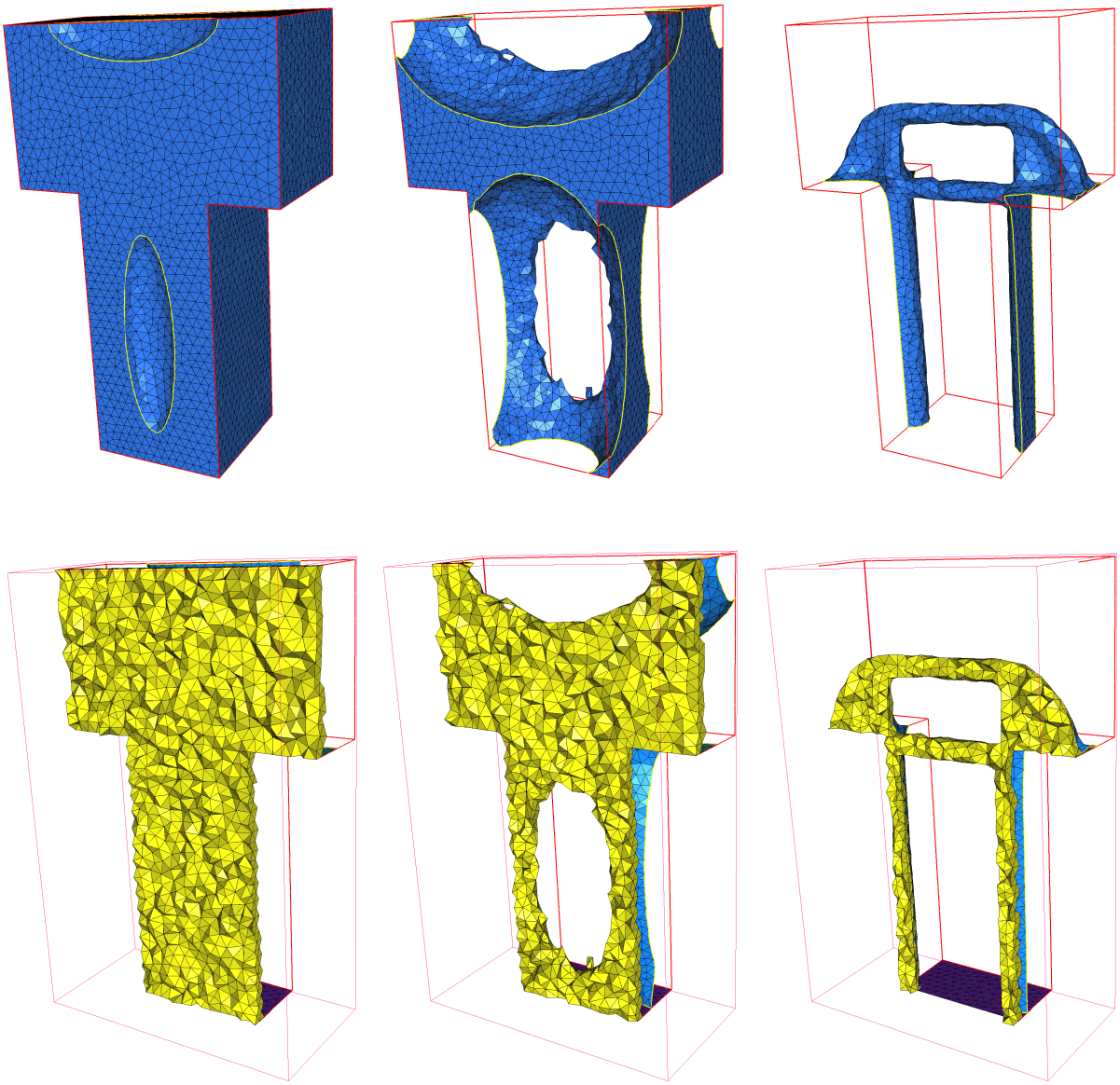


FIGURE 11. *From left to right :* Initial, 10th and final (50th) iterations of the optimal mast test-case. Only the boundary $\partial\Omega$ of each shape Ω is displayed on the upper range, and only the 'interior' part of the associated mesh $\mathcal{T}_\Omega$ of D is displayed on the corresponding cuts of the lower range.

We now rely on the general framework of section 7.1 to account for the numerical resolution of the above problem, with the velocity field:

$$f(t, x, \Gamma) = u,$$

where u is the solution of (12-13), defined on the whole computational domain D.

The proposed approach is used to resolve the *rising bubble* test case. We present the result of a numerical experiments with a single drop of primary alcohol of radius 1 mm inside a rectangular cavity $D = [0.12 \cdot 10^{-3}]^2 \times [0.3 \cdot 10^{-3}]\ m^3$ filled with water, as depicted on Figure 13. Fluid properties of the system are given in Table 1. At rest at time $t = 0$, the bubble starts to rise in z-direction. The initial

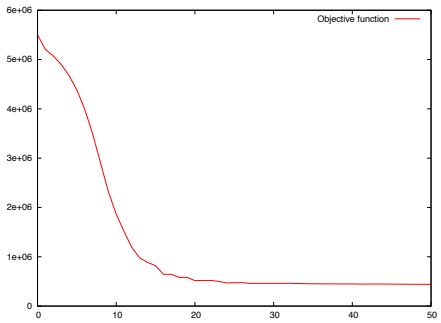


FIGURE 12. Objective function versus iteration number for the optimal mast test-case.

variable (units)	n-butanol	water
ρ (kg/m^3)	845.4	986.5
μ $(kg/m\,s)$	3.281 1e-3	1.388 1e-3

TABLE 1. *Fluid properties of the rising bubble test case.*

coarse triangulation contains $2,465$ vertices and $13,098$ tetrahedra and is refined in the vicinity of the bubble. The triangulation is adapted at each time step $t = 0.02\,s$ to follow the moving interface; the minimal size has been set to $h_{min} = 5 \cdot 10^{-5}\,m$. At time $t = 0.2\,s$ (resp. $t = 0.4\,s$), the triangulation contains $9,874$ (resp. 9400) vertices and $54,133$ (resp. 51576) tetrahedra. Figure 13 shows the droplet and a cuts through the adapted meshes at different time steps.

Acknowledgement. This work has been supported by the RODIN project (FUI AAP 13).

8. Conclusion

In this article, we have presented an iterative algorithm for remeshing an initial tetrahedral mesh, that may be ill-shaped or undersampled, into a well-shaped, adapted mesh, via local operations carried out both on the surface and interior parts. This algorithm has been used as the cornerstone in the context of tetrahedral mesh generation from a surface triangulation, and in the context of free an moving boundary problems.

The associated domain remeshing program `mmg3d`, version 5 is free and can be obtained by contacting the authors.

This work stirs several perspectives as for future work. On the one hand, the proposed domain remeshing algorithm is only able to deal with isotropic size prescriptions; we would like to extend it to the anisotropic context, for instance in view of the previous work [14], which only concerned the interior part of a domain. Sharp features recovering (at least when they are prescribed on the initial triangulated contour) in the context of mesh generation presented in section 6 also seems to be of great interest. Eventually, the applications proposed in sections 7.2 and 7.3 in the fields of shape optimization and computational fluid dynamics are subject to ongoing work, addressing more complex mechanical models.

References

[1] G. Allaire, C. Dapogny, P. Frey, *Topology and Geometry Optimization of Elastic Structures by Exact Deformation of Simplicial Mesh*, C. R. Acad. Sci. Paris, Ser. I, vol. 349, no. 17, pp. 999-1003 (2011).

[2] G. Allaire, F. Jouve, A.M. Toader, *Structural optimization using shape sensitivity analysis and a level-set method*, J. Comput. Phys., 194, pp. 363-393 (2004).

[3] P. Alliez, É. Colin de Verdière, O. Devillers and M. Isenburg, *Isotropic Surface Remeshing*, Shape Modeling International, (2003), pp 49–58.

[4] M. V. Anglada, N.P. Garcia and P. B. Crosa, *Directional Adaptive Surface Triangulation*, Computer Aided Geometric Design, 16 (1999), pp. 107–126.

[5] H. Borouchaki and P.J. Frey, *Surface meshing using a geometric error estimate*, Int. j. numer. methods engng., 58 (2003), pp 227–245.

[6] G. Barles, *Remarks on a flame propagation model*, INRIA: Technical Report, 451 (1985).

[7] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez and Bruno Lévy, *Polygon Mesh Processing*, A.K. Peters, (2010).

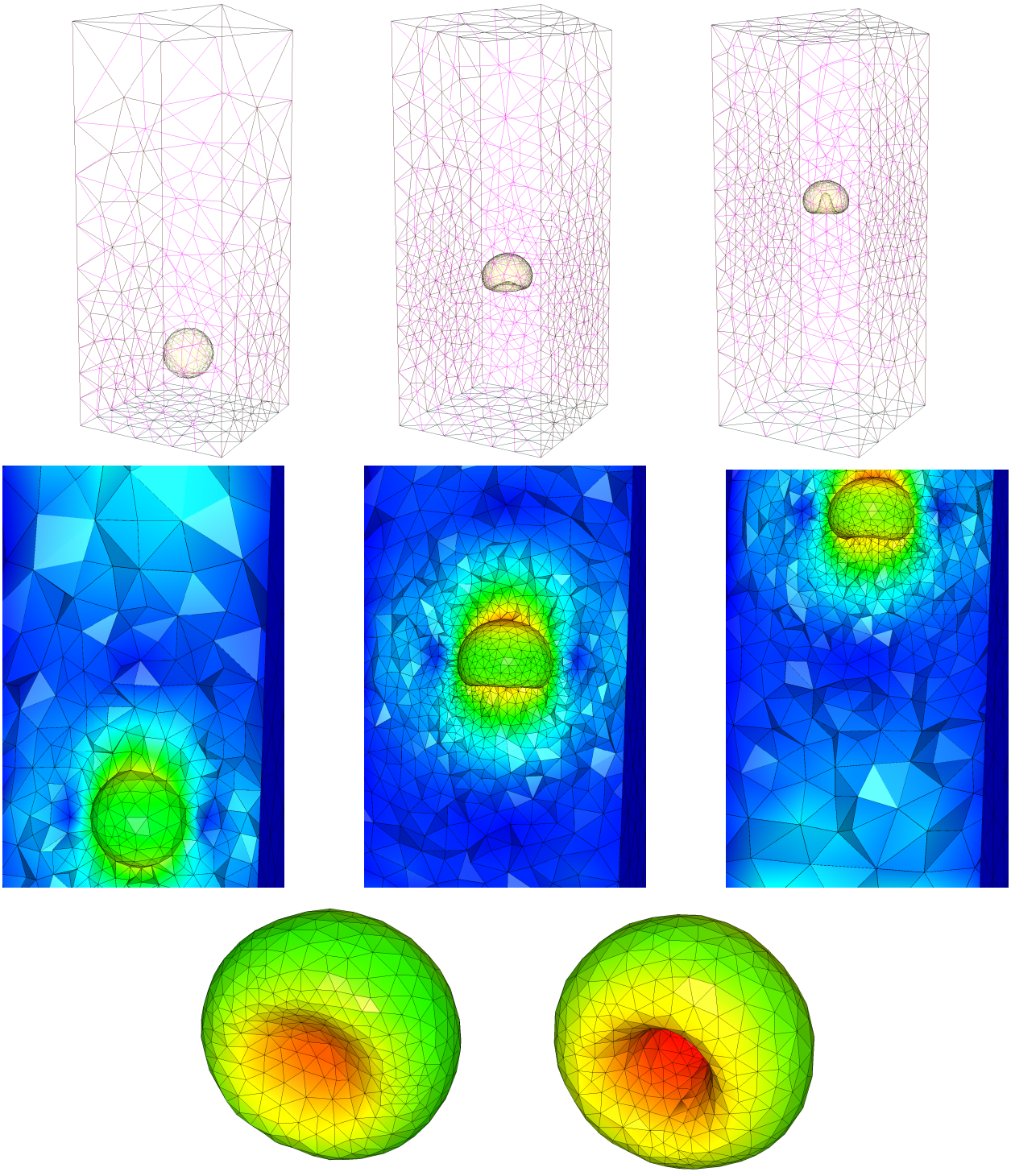


FIGURE 13. *From left to right:* initial, 10th and 20th iteration for the rising bubble test case. Top: only the boundary of the computational domain is shown. Middle: cut through the tetrahedral meshes. Bottom: enlargement around the droplet at iterations 10 and 20.

[8] C. BUI, P. FREY AND B. MAURY, *A coupling strategy based on anisotropic mesh adaptation for solving two-fluid flows*, Int. J. Numer. Meth. Fluids, 66(10), (2010), pp 1226–1247.

[9] P.G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North Holland Publishing Company, (1978).
[10] J. Crank, *Free and moving boundary problems*, Clarendon Press, Oxford, (1984).
[11] C. Dapogny, *Ph. d. thesis*, Thèse de l'Université Paris VI, (2013).
[12] C. Dapogny, P. Frey, *Computation of the signed distance function to a discrete contour on adapted triangulation*, Calcolo, Volume 49, Issue 3, pp. 193-219 (2012).
[13] C. Dobrzynski, *Adaptation de maillage anisotrope 3d et application à l'aéro-thermique des bâtiments*, Thèse de l'Université Paris VI, (2005).
[14] C. Dobrzynski and P. Frey, *Anisotropic Delaunay mesh adaptation for unsteady simulations*, Proc. 17th Int. Meshing Roundtable, Pittsburgh, (2008).
[15] M. Do Carmo, *Riemannian Geometry*, Mathematics : Theory & Applications, 2*nd* Edition, Birkhäuser, (1993).
[16] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, (1995), pp 173–182.
[17] L.C. Evans and J. Spruck, *Motion of level sets by mean curvature. I*, J. Differential Geom. Volume 33, Number 3 (1991), pp 635–681.
[18] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*, Academic Press Inc, 4th Edition, (1997).
[19] S. Osher and R.P. Fedkiw, *Level Set Methods: An Overview and Some Recent Results*, J. Comput. Phys., 169, 2, pp. 463-502 (2001).
[20] M.S. Floater and K. Hormann, *Surface parameterization: a tutorial and survey*, In Advances in Multiresolution for Geometric Modelling, (2005), pp 157–186.
[21] P.J. Frey, *About Surface Remeshing*, Proceedings, 9th International Meshing Roundtable, Sandia National Laboratories, (2000), pp 123–136.
[22] P.J. Frey and P.L. George, *Mesh Generation : Application to Finite Elements*, Wiley, 2nd Edition, (2008).
[23] P.-L. George, *Improvements on Delaunay-based three-dimensional automatic mesh generator*, Finite Elements in Analysis and Design 25 (1997) pp. 297–317.
[24] J. Glimm, J. W. Grove, X. L. Li, K.-M. Shyue, Y. Zeng, Q. Zhang, *Three Dimensional Front Tracking*, SIAM J. Sci. Comp, 19, (1995), pp 703–727.
[25] F. de Gournay, *Velocity extension for the level-set method and multiple eigenvalues in shape optimization.* SIAM J. on Control and Optim., 45, no. 1, 343–367 (2006).
[26] M.G. Vallet, F. Hecht and B. Mantel, *Anisotropic Control of Mesh Generation Based upon a Voronoi Type Method*, Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, (1991).
[27] A. Henrot and M. Pierre, *Variation et Optimisation de Formes : une Etude Géométrique*, collection Mathématiques et Applications, vol. 48, Springer (2005).
[28] D.R. Lynch, *Unified approach to simulation on deforming elements with application to phase change problems*, J. Comput. Phys., 47, pp. 387-411 (1982).
[29] N. Moes, J. Dolbow and T. Belytschko, *A finite element method for crack growth without remeshing*, Int. J. Numer. Meth. Engng. 46, (1999), pp.131–150.
[30] S.J. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[31] X. Li, J.-F. Remacle, N. Chevaugeon and M.S. Shepard, *Anisotropic Mesh Gradation Control*, Proceedings, 13th International Meshing Roundtable, Sandia National Laboratories, (2004), pp 401–412.
[32] W.E. Lorensen and H.E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, COMPUTER GRAPHICS, 21,4, (1987), pp. 163–169.
[33] F. Murat and J. Simon, *Sur le contrôle par un domaine géométrique*, Technical Report RR-76015, Laboratoire d'Analyse Numérique (1976).
[34] P.-O. Persson, *Mesh Generation for Implicit Geometries*, Ph.D. thesis, Department of Mathematics, MIT, Dec 2004.
[35] P.-O. Persson and G. Strang, *A Simple Mesh Generator in MATLAB*, SIAM Review, 46, no. 2, (2004), pp. 329–345 .
[36] O. Pironneau, *The finite element methods for fluids*, Wiley (1989).
[37] J.A. Sethian, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry,Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).
[38] P.N. Shankar, *Slow Viscous Flows*, Imperial College Press, London, U.K., (2007).
[39] J. Strain, *Semi-Lagrangian Methods for Level Set Equations*, J. Comput. Phys., 151 (1999) pp. 498–533.
[40] V. Surazhsky, P. Alliez and C. Gotsman, *Isotropic Remeshing of Surfaces: a Local Parameterization Approach*, RR-4967 INRIA (2003).
[41] Sussman M., Smereka P., Osher S., *A level set approach for computing solutions to incompressible two-phase flows*, J. Comput. Phys., 114, (1994), pp 146–159.
[42] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan, *A Front-Tracking Method for the Computations of Multiphase Flow*, J. Comput. Phys., 169, pp. 708-759 (2001).
[43] A. Vlachos, J. Peters, C. Boyd and J.L. Mitchell, *Curved PN Triangles*, Symposium on Interactive 3*d* Graphics, (2001), pp 159–166.