Insertion of Triangulated Surfaces Into a Meccano Tetrahedral Discretization by Means of Mesh Refinement and Optimization Procedures

E. Ruiz-Gironés[1*], A. Oliver[2], G. V. Socorro-Marrero[2], J. M. Cascón[3], J. M. Escobar[2], R. Montenegro[2] and J. Sarrate[1]

[1] *Laboratori de Càlcul Numèric (LaCàN), www.lacan.upc.edu, Departament d'Enginyeria Civil i Ambiental, ETSECCPB, Universitat Politècnica de Catalunya - BarcelonaTech, Jordi Girona 1-3, 08034 Barcelona, Spain.*
[2] *University Institute for Intelligent Systems and Numerical Applications in Engineering (SIANI), www.dca.iusiani.ulpgc.es/proyecto2015-2017, University of Las Palmas de Gran Canaria, Campus de Tafira, 35017 Las Palmas, Spain.*
[2] *Grupo de Investigación en Simulación Numérica y Cálculo Científico, diarium.usal.es/sinumcc/ Department of Economics and Economic History, Faculty of Economics and Business, University of Salamanca, 37007 Salamanca, Spain.*

SUMMARY

In this paper, we present a new method for inserting several triangulated surfaces into an existing tetrahedral mesh generated by the meccano method. The result is a conformal mesh where each inserted surface is approximated by a set of faces of the final tetrahedral mesh. First, the tetrahedral mesh is refined around the inserted surfaces to capture their geometric features. Second, each immersed surface is approximated by a set of faces from the tetrahedral mesh. Third, following a novel approach the nodes of the approximated surfaces are mapped to the corresponding immersed surface. Fourth, we untangle and smooth the mesh by optimizing a regularized shape distortion measure for tetrahedral elements in which we move all the nodes of the mesh, restricting the movement of the edge and surface nodes along the corresponding entity they belong to. The refining process allows approximating the immersed surface for any initial meccano tetrahedral mesh. Moreover, the proposed projection method avoids computational expensive geometric projections. Finally, the applied simultaneous untangling and smoothing process delivers a high-quality mesh and ensures that the immersed surfaces are interpolated. Several examples are presented to assess the properties of the proposed method.

KEY WORDS: Surface Insertion; Meccano Method; Tetrahedral Mesh Generation; Mesh Untangling and Smoothing; Surface Parameterization; Volume Parameterization

1. INTRODUCTION

In most of the computational simulations in applied sciences and engineering, the geometry, delimited by outer surfaces and probably including inner surfaces, is usually a given data. However, in a wide range of applications, it is not possible to generate *a priori* a mesh that contains the inner surfaces, since it may be an unknown of the problem. This is the case of immiscible multi-fluid problems in which the interface between fluids evolves in time, and its location depends on the physics of the problem. Another situation is that the exact location of the surface may be known in advance nevertheless it changes over time. In this case, a new mesh for each time step has to

*Correspondence to: Eloi Ruiz-Gironés (eloi.ruiz@upc.edu)

Contract/grant sponsor: FEDER and the Spanish Government, "Ministerio de Economía y Competitividad", and "Fondo Sectorial CONACYT SENER HIDROCARBUROS"; contract/grant number: CTM2014-55014-C3-1-R, CTM2014-55014-C3-3-R, 163723

be generated. For instance, in computational fluid dynamics, the motion of a rotating propeller is prescribed, and the surface position is known at each time step. Moreover, during an advanced stage of a design process, a new geometric feature can be added to an already discretized model. The additional surfaces impose the generation of a new discretized model that reproduces the new surfaces. For example, a new fault has to be inserted in an already meshed oil reservoir.

Several numerical strategies can be adopted to deal with this problem. It is possible to design sophisticated solvers to track the new or moving surfaces. For example, the phase-field approach consists of adding new unknown fields in the governing equations of the physical problem. The surfaces are defined as the zero level set of the additional field unknowns [1, 2, 3]. The extended finite element method (X-FEM) formulation [4, 5, 6] adds new shape functions to include discontinuities in the solution. Thus, the surface to be added is defined using these discontinuities.

Nevertheless, it is not always possible to modify the solver to include these techniques. Therefore, it is necessary to modify the mesh in order to approximate the additional or moving surfaces using the triangles of the tetrahedral mesh. A straightforward solution is to generate an entirely new mesh for the geometric model. However, this is a costly operation, especially if it has to be performed in all the time steps of a transient problem. Thus, other approaches have been developed to reduce the computational cost of generating a new mesh. On the one hand, it is possible to approximate the surface by performing a mesh moving technique, see [7, 8, 9]. First, it is selected a set of triangles to approximate the surface. Then, these triangles are moved onto the surface while keeping valid elements. To perform this process, the size of the tetrahedral mesh has to be small enough to reproduce the geometric features of the immersed surface. On the other hand, the tetrahedral mesh can be locally remeshed in order to approximate the given surface, see [10, 11, 12]. First, a triangular mesh on the surface is generated, and the tetrahedral elements that are *near* the surface are removed. Then, the created cavity is remeshed to generate a conformal mesh that includes the surface. Note that the cavity has to be large enough to accommodate the new elements, and small enough to keep the number of modified elements as small as possible.

In this work we propose a novel approach to insert a given triangulated surface into an existing tetrahedral mesh generated using the meccano method [13, 14, 15, 16]. It combines the benefits of local refinement processes and mesh moving techniques. First, we locally refine the tetrahedral mesh around the surface until the tetrahedral mesh captures the geometric features of the surface. Specifically, we use the Kossaczký method [17] that delivers a fast and robust process, since it does not depend on geometric predicates. Second, we select a set of triangles of the tetrahedral mesh to approximate the surface. To this end, we use a set of rules to ensure that a high-quality mesh can be obtained. Third, we propose a new projection algorithm based on the Floater parameterization [18] of the immersed and approximating surfaces. We highlight that we use the same parametric space for both surfaces. Therefore, we can define a mapping from the approximating surface onto the immersed one. We use this mapping to ensure that the projected nodes lie on the immersed surface. The projection step introduces inverted and low-quality elements. Thus, in the fourth step, we optimize the quality of the final mesh by using a robust untangling and smoothing process based on an optimization of a single distortion measure [19]. Specifically, we optimize a regularized version of the shape distortion measure for tetrahedral elements in which we move all the nodes of the mesh, restricting the movement of the edge and surface nodes along the corresponding entity they belong to [20]. This is achieved by expressing the coordinates of these nodes in terms of their parametric coordinates during the optimization process. Although the target of the optimization process is the distortion of the tetrahedra, the method also improves the quality of the triangles that approximate the surfaces of the model since we allow moving the surface nodes.

The proposed method has several advantages. It is independent of the initial mesh because of the refinement process that adapts the mesh to the features of the immersed surface. Moreover, we propose a fast and robust projection process that involves solving two linear problems: one for all the nodes of the immersed surface, and another one for all the nodes of the approximating surface. In this way, we avoid solving the non-linear problem involved in the geometric projection to the immersed surface of each node of the approximating surface. Finally, the applied untangling and smoothing process ensures a valid and high-quality mesh that interpolates the immersed surface.

The rest of the paper is structured as follows. In Section 2 we overview the meccano method. In Section 3 we detail the proposed algorithm to insert a given surface in a tetrahedral meccano mesh. Finally, in Section 4, we present three examples, both academic and realistic, illustrating the capabilities of the proposed method.

2. MECCANO OVERVIEW

The meccano method [13, 14, 15, 16] is an automatic tetrahedral mesh generator for complex genus-zero solids. The method requires a surface triangulation of the solid boundaries and a computational domain that coarsely approximates the solid. This computational domain is called meccano. The procedure builds an adaptive tetrahedral mesh in the meccano and deforms it to match the physical domain. For this purpose, the method combines several procedures: an automatic mapping from the boundary of the meccano to the boundary of the solid, a 3-D local refinement algorithm, and a simultaneous mesh untangling and smoothing. It is important to point out that this method also provides a continuous element-wise linear volumetric parameterization from the computational domain to the solid.

The construction of the meccano (the computational domain) is not automatic for complex solids (genus bigger than zero) and requires user intervention. We are now working on a procedure based on octree meshes that will automatically provide the meccano for more complex solids. Another alternative could be to use a polycube of the solid as the meccano. A method for its automatic construction has been recently proposed in [21, 22].

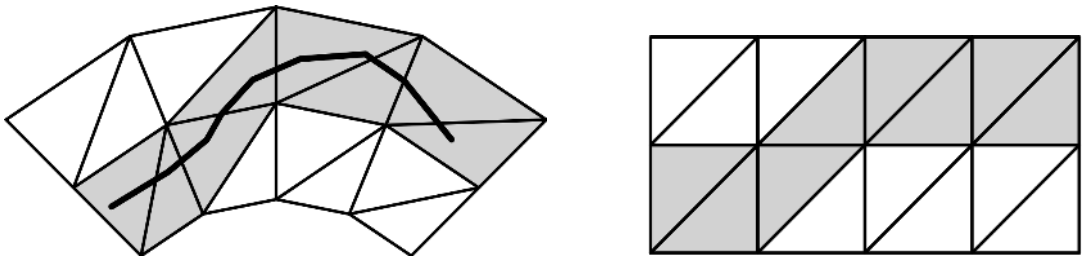
The main steps of the meccano tetrahedral mesh generation algorithm are summarized in Algorithm 1. The input data is a solid, Ω, defined by its boundary representation (surface triangulation or CAD model), and a given precision to approximate its boundary, ε.

The first step of the procedure, Line 2, is to construct a meccano, $\mathcal{D}$, that approximates the solid, by connecting polyhedral pieces. Then, in Line 3, a discrete mapping, $\mathbf{\Pi}$, between the boundary of the meccano and the boundary of the solid is computed using a procedure based on the mean value parametrization proposed by Floater in [18]. Note that this parameterization is a continuous and element-wise linear mapping. In Line 4, an initial coarse mesh of the meccano is generated, and the boundary nodes are located on the solid boundary using the mapping $\mathbf{\Pi}$. In Lines 5–9, we obtain a mesh that approximates the solid boundary with the given tolerance ε. Specifically, in Line 6, we get the list of triangles that do not correctly approximate the boundary of the solid, and then, in Line 7, we refine those triangles by dividing their adjacent tetrahedra using the Kossaczký method [17]. In Line 8, we project the new nodes onto the boundary of the solid using the mapping $\mathbf{\Pi}$. We iterate this process until there are no triangles to refine. Note that when the nodes are mapped onto the solid boundary, low-quality and inverted elements may appear. Thus, a simultaneous untangling and smoothing procedure [19, 23] is applied in order to obtain a valid and high-quality tetrahedral mesh. As commented before, the meccano method automatically provides a volumetric parameterization from the computational domain to the solid. From now on, abusing of notation, we denote this mapping as $\mathbf{\Pi}$.

The meccano method and polycube-map have some similarities. In fact, both use a coarse approximation of the domain as a computational space. However, they are different algorithms. On the one hand, the term polycube (a solid formed from a union of cubes) was first introduced by Tarini, *et al.* in [24], where they built a mapping from the boundary of an input object to the surface of a manually constructed polycube. Then, some volumetric polycube parameterizations were proposed, e.g. [25, 26]. As consequence, adaptive all-hex meshes of the input domain are produced [26]. In the last years, an effort has been done to construct automatically the polycube and to produce low distortion mapping; among others [21, 22]. On the other hand, the meccano method was first proposed in [27], and then developed in [13, 14, 15, 16]. This algorithm, as we commented before, requires a coarse computational domain, then builds a surface parameterization and combines refinement and a mesh optimization to produce an adaptive tetrahedral mesh of the input domain. As a result, a piecewise linear volumetric parameterization is obtained ($\mathbf{\Pi}$).

Algorithm 1 Meccano tetrahedral mesh generation.

function `MeccanoMesher`(Solid Ω, Real ε)
  Meccano $\mathcal{D} \leftarrow$ `getMeccano`(Ω)
  Mapping $\mathbf{\Pi} \leftarrow$ `getBoundaryMapping`($\mathcal{D}$, Ω)
  Mesh $\mathcal{M} \leftarrow$ `getInitialMesh`($\mathcal{D}$, $\mathbf{\Pi}$)
  while `distance`($\partial\mathcal{M}$, $\partial\Omega$) $> \varepsilon$ **do**
    TriangleList $\tau \leftarrow$ `getTrianglesToRefine`($\mathcal{M}$, Ω, ε)
    TriangleList $\tau' \leftarrow$`refineTriangles`(τ)
    `projectNewNodesToBoundary`(τ', $\mathbf{\Pi}$)
  end while
  `qualityOptimization`($\mathcal{M}$)
end function



(a) (b)

Figure 1. Initial meccano mesh and immersed polyline S (thick line). (a) Physical mesh; and (b) computational mesh.

3. SURFACE INSERTION ALGORITHM

In this section, we detail the proposed algorithm to insert a surface into an existing tetrahedral mesh generated using the meccano method. Although we describe the algorithm inserting a single surface, it can be extended to the insertion of several surfaces. To illustrate the process, we show a 2D analogy.

3.1. Problem Statement

Our input data is a tetrahedral mesh $\mathcal{M}$ generated by the meccano method, composed of M tetrahedra, and an immersed triangular mesh $\mathcal{S}$, containing N triangles, such that $\text{dist}(\partial\mathcal{M}, \mathcal{S}) > 0$. We assume that $\mathcal{S}$ is simply connected, single oriented, and has a boundary. Figure 1 shows the initial meccano mesh for a two-dimensional geometry, and a polyline $\mathcal{S}$ to be inserted in the mesh.

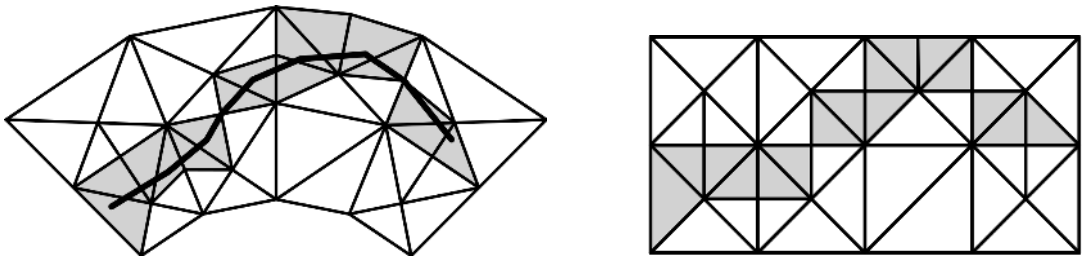
Our target is to construct a surface $\mathcal{S}_\mathcal{M}$ composed of triangles, edges, and vertices belonging to a conformal tetrahedral mesh, obtained by refining $\mathcal{M}$, which approximates $\mathcal{S}$ with a given tolerance.

3.2. Volumetric Approximation

We define $\mathcal{T}$ as the set of tetrahedra of $\mathcal{M}$ that intersect the inserted surface, $\mathcal{S}$. Then, we recursively update the set $\mathcal{T}$ by refining those tetrahedra $E \in \mathcal{T}$ such that

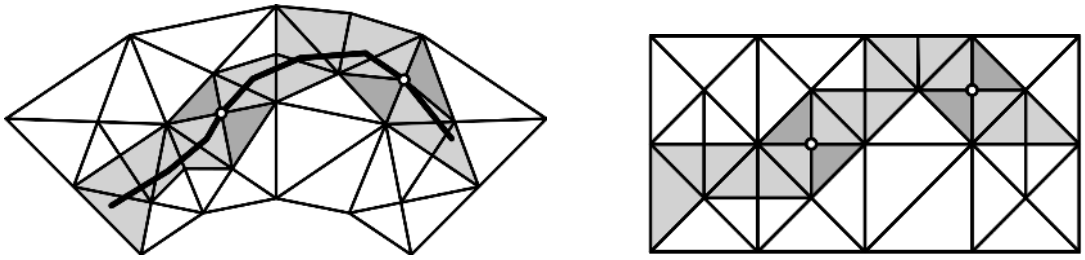
$$r^E \geq l_c, \qquad E \in \mathcal{T} \tag{1}$$

where r^E is the length of the longest edge of the element E, and l_c is a characteristic length determined by the geometric features of $\mathcal{S}$, and verifying $l_c < \text{dist}(\partial\mathcal{M}, \mathcal{S})$. That is, l_c represents an element size small enough to capture the geometric features of the surface $\mathcal{S}$. Thus we refine the tetrahedra that are larger than the desired element size. To refine the tetrahedra, we apply the Kossaczký method. The original tetrahedra are removed from $\mathcal{T}$, and the refined ones that intersect



(a) (b)

Figure 2. Refined meccano mesh around the immersed polyline $\mathcal{S}$. (a) Physical mesh; and (b) computational mesh.



(a) (b)

Figure 3. Extension of $\mathcal{T}$ to be topologically equivalent to a sphere. (a) Physical mesh; and (b) computational mesh.

$\mathcal{S}$ are added. This process is iterated until there is no tetrahedron in $\mathcal{T}$ that verifies Equation (1). Figure 2 shows the refinement process of the mesh $\mathcal{M}$ and the triangles that intersect $\mathcal{S}$.

3.3. Volumetric Approximation Healing

In order to obtain a high-quality mesh, we need to impose the following constraints to the set of tetrahedra $\mathcal{T}$. First, we add elements to $\mathcal{T}$ until its boundary is topologically equivalent to a sphere. To this end, we ensure that all the nodes that belong to the boundary of $\mathcal{T}$ define a local disk. That is, the Euler characteristic, χ, of each node on the boundary $\mathcal{T}$ verifies:

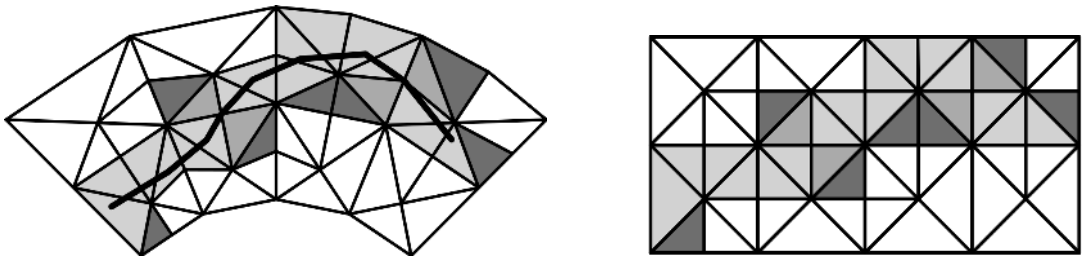
$$\chi := n_E - n_F + n_C = 1, \tag{2}$$

where n_E, n_F and n_C are, respectively, the number of edges, faces, and cells of $\mathcal{T}$ that are adjacent to the node. Thus, we add the elements adjacent to the nodes that do not verify Equation (2). Figure 3 shows this process; the two locations–nodes in this case–where the boundary of $\mathcal{T}$ is not equivalent to a sphere are highlighted, as well as the triangles added to $\mathcal{T}$.

Then, we enforce that the counterpart of the boundary faces and edges in the computational space are parallel to a coordinate plane or axis, respectively, Figure 4. This is accomplished by adding additional tetrahedra into the set $\mathcal{T}$. First, we identify the faces in the computational domain that are not parallel to a coordinate plane. Then, we refine the adjacent tetrahedra to the same refinement level. Finally, we add the new tetrahedra into $\mathcal{T}$. This process ensures that we minimize the number of added tetrahedra into $\mathcal{T}$.

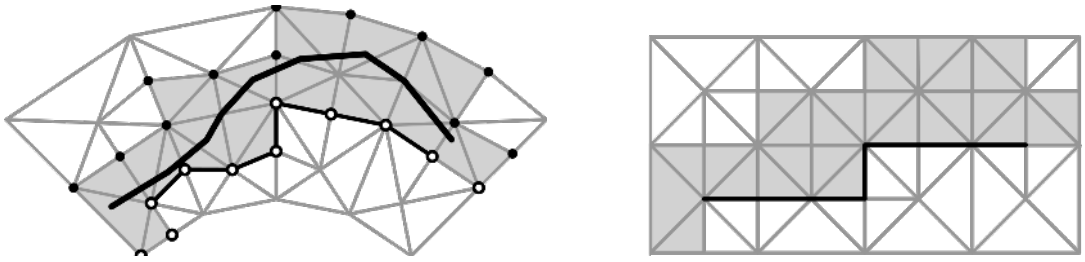
3.4. Surface Extraction

The set of tetrahedra $\mathcal{T}$ contains the given surface $\mathcal{S}$ completely in its interior. We will obtain $\mathcal{S}_\mathcal{M}$, the approximation of $\mathcal{S}$, from the boundary faces of $\mathcal{T}$. The main idea is to select the faces of the boundary of $\mathcal{T}$ that are at one *side* of the inserted surface, $\mathcal{S}$, and are not adjacent to any other surfaces of the model. We first classify the boundary nodes of $\mathcal{T}$ according to the side of $\mathcal{S}$ they are located. Figure 5a depicts the nodal classification of the boundary nodes of $\mathcal{T}$ using black and white



(a) (b)

Figure 4. Extension of $\mathcal{T}$ to get boundary edges parallel to the computational axis. (a) Physical mesh; and (b) computational mesh.



(a) (b)

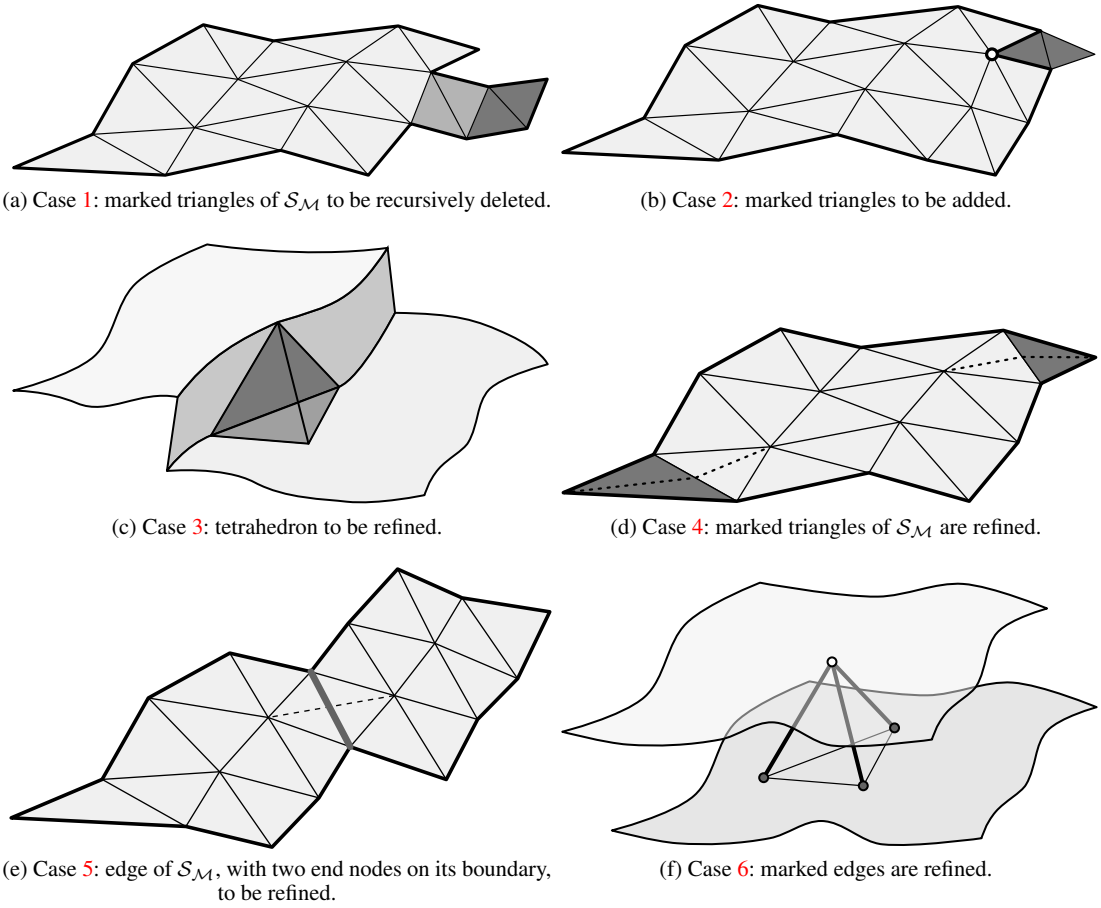
Figure 5. Initial selection of $\mathcal{S}_\mathcal{M}$. (a) Physical mesh and classification of boundary nodes of $\mathcal{T}$ according the side of $\mathcal{S}$ they are located; and (b) computational mesh.

circles. Then, we select the boundary triangles of $\mathcal{T}$ that have all the nodes classified in the same side of $\mathcal{S}$ and are not adjacent to any other surface of the model.

3.5. Surface Healing

To increase the quality of the final mesh and avoid unresolvable degenerated elements, we apply the following steps to $\mathcal{S}_\mathcal{M}$.

1. *Remove thin protrusions at the boundary of* $\mathcal{S}_\mathcal{M}$. We define thin protrusions as pairs of connected triangles of $\mathcal{S}_\mathcal{M}$ with three edges at the boundary of $\mathcal{S}_\mathcal{M}$. Thin protrusions at the boundary of $\mathcal{S}_\mathcal{M}$ may induce low-quality elements in the tetrahedral mesh. Thus, we recursively remove all thin protrusions in the surface approximation, see Figure 6a.
2. *Remove high-connectivity nodes at the boundary of* $\mathcal{S}_\mathcal{M}$. Nodes at the boundary of $\mathcal{S}_\mathcal{M}$ that are adjacent to five or more triangles in $\mathcal{S}_\mathcal{M}$ may induce low-quality elements in the final mesh. To solve this issue, we add the adjacent face to $\mathcal{S}_\mathcal{M}$, see Figure 6b. The two edges of the boundary of $\mathcal{S}_\mathcal{M}$ adjacent to the node, define a plane in the computational domain. We add to $\mathcal{S}_\mathcal{M}$ the faces of the boundary of $\mathcal{T}$ that belong to this plane and are adjacent to the boundary edges.
3. *Refine tetrahedra of* $\mathcal{M}$ *with more than one face on* $\mathcal{S}_\mathcal{M}$. In order to avoid degenerated tetrahedra, we impose that each tetrahedron contributes with only one face on the approximation $\mathcal{S}_\mathcal{M}$. Thus, we refine all tetrahedra with two faces on $\mathcal{S}_\mathcal{M}$, see Figure 6c. Note that the resulting elements have only one face on $\mathcal{S}_\mathcal{M}$, since we have imposed that the faces in $\mathcal{S}_\mathcal{M}$ are parallel to the coordinate planes in the computational space, and we are using the Kossaczký refinement.
4. *Refine faces of* $\mathcal{M}$ *with more than one edge at the boundary of* $\mathcal{S}_\mathcal{M}$. In order to avoid degenerated faces at the boundary of $\mathcal{S}_\mathcal{M}$, we impose that each face contributes to the boundary of $\mathcal{S}_\mathcal{M}$ with only one edge. To solve this issue, we refine such faces, see Figure 6d.
5. *Refine interior edges of* $\mathcal{S}_\mathcal{M}$ *with the end nodes on the boundary of* $\mathcal{S}_\mathcal{M}$. Inner edges of the inserted surface with end nodes on its boundary may induce low quality tetrahedral elements.



(a) Case 1: marked triangles of $\mathcal{S}_\mathcal{M}$ to be recursively deleted.

(b) Case 2: marked triangles to be added.

(c) Case 3: tetrahedron to be refined.

(d) Case 4: marked triangles of $\mathcal{S}_\mathcal{M}$ are refined.

(e) Case 5: edge of $\mathcal{S}_\mathcal{M}$, with two end nodes on its boundary, to be refined.

(f) Case 6: marked edges are refined.

Figure 6. Illustration of the six different cases of the surface healing process.

To avoid this, we divide those edges by refining the adjacent elements. Thus, a new node is introduced on the edge providing more flexibility to the mesh optimization procedure, see Figure 6e.

6. *Refine edges with nodes on different surfaces.* When the nodes of an edge belong to different surfaces (immersed or boundary surface), it may be difficult to optimize the quality of the mesh and obtain high-quality elements. Hence, we refine such edges of the mesh to give more freedom to the optimization process, see Figure 6f.

If the value of l_c is small enough the previous conditions are achieved as a consequence of: 1) $\mathcal{S}_\mathcal{M}$ has parallel faces to the coordinate planes in the computational domain, and 2) the properties and structure of the Kossaczký refinement.

3.6. Surface Projection

The next step consists on projecting the nodes of $\mathcal{S}_\mathcal{M}$ onto $\mathcal{S}$. At the end of this step, all of the nodes in $\mathcal{S}_\mathcal{M}$ will be located on the target surface $\mathcal{S}$.

First, we compute the Floater parameterizations [18] of $\mathcal{S}$ and $\mathcal{S}_\mathcal{M}$, denoted by φ and $\varphi_\mathcal{M}$, respectively, as

$$\varphi : \mathcal{U} \to \mathcal{S}, \qquad \varphi_\mathcal{M} : \mathcal{U} \to \mathcal{S}_\mathcal{M}, \tag{3}$$

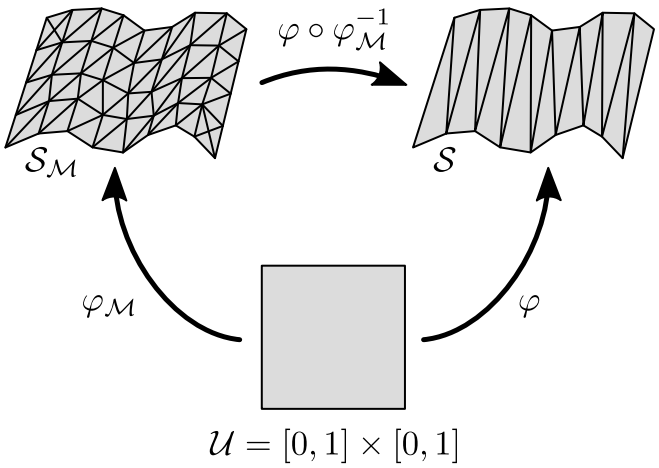


Figure 7. Surface projection diagram.

where $\mathcal{U} = [0,1] \times [0,1]$ is a common parametric space for both surfaces, see Figure 7. Then, each node of $\mathcal{S}_{\mathcal{M}}$, $\mathbf{x}_i$, is moved to the new position, $\mathbf{x}'_i$, according to:

$$\mathbf{x}'_i = \boldsymbol{\varphi} \circ \boldsymbol{\varphi}_{\mathcal{M}}^{-1}(\mathbf{x}_i). \tag{4}$$

The mapping obtained in Equation (4) is bijective due to cases 4 and 5. Specifically, we avoid triangles with three nodes at the boundary of $\mathcal{S}_{\mathcal{M}}$ and inner edges of the surface with both nodes at its boundary. Note that this projection procedure has several advantages. We avoid to solve a non-linear projection problem for each node to be projected. Instead, we solve two linear problems to compute the mappings of $\mathcal{S}_{\mathcal{M}}$ and $\mathcal{S}$. Moreover, we obtain the parametric coordinates of the projected nodes on $\mathcal{S}$. This information is necessary to perform the untangling and smoothing process, detailed in the next section.

3.7. Mesh Optimization

After the projection process, inverted and low-quality elements may appear. Thus, to obtain a valid and high-quality mesh, we apply an optimization procedure based on the simultaneous untangling and smoothing technique proposed in [19, 23]. Moreover, we add the capability to optimize the mesh quality by moving the location of inner, edge and surface nodes, according to [20]. Note that the result of the optimization problem defines a high-quality volumetric mapping between the computational domain and the physical domain in which the inserted surfaces are interpolated.

3.7.1. Regularized Shape Distortion Measure According to [28], the shape distortion measure is defined as

$$\eta_E(\boldsymbol{J\phi}) = \frac{\|\boldsymbol{J\phi}\|^2}{3|\sigma(\boldsymbol{J\phi})|^{\frac{2}{3}}}, \tag{5}$$

where $\boldsymbol{\phi}$ is the affine mapping between the ideal element E^I and the physical element, E^P, $\boldsymbol{J\phi}$ is the Jacobian matrix of $\boldsymbol{\phi}$, $\|\cdot\|$ is the Frobenius norm, and σ is the determinant. In the meccano method, the ideal element of a physical tetrahedron E_i^P is the rectangular tetrahedron that is its counterpart in the computational space E_i^I. Thus, in our case $\boldsymbol{\phi} = \boldsymbol{\Pi}$, and therefore

$$E_i^P = \boldsymbol{\Pi}(E_i^I). \tag{6}$$

This distortion measure is invariant to translation, rotation, and scale [28]. It takes values in the range $[1, \infty)$, being $\eta = 1$ when the physical and ideal element only differ in the invariants, and $\eta = \infty$ when the physical element is degenerated. The quality of an element is defined as the inverse

of the distortion

$$q = \frac{1}{\eta}. \tag{7}$$

Thus, it takes values between 0 (degenerated element) and 1 (ideal element).

The shape distortion measure presents asymptotes when $\sigma(\boldsymbol{J\phi}) = 0$, and this prevents its use in a continuous minimization process when inverted elements are present. To overcome this drawback, we use the regularized distortion measure introduced in [29, 19] for linear elements, in which σ is replaced by

$$h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right), \tag{8}$$

where δ is a small parameter that depends on the problem and can be calculated according to [23, 30]. This regularization is also used for optimizing curved high-order meshes [30, 31, 20].

3.7.2. Objective Function The simultaneous untangling and smoothing algorithm is based on the optimization of an objective function defined in terms of the regularized distortion measure of the elements adjacent to a node. Let $\mathbf{x}$ be the location of the node to be moved, and m the number of adjacent tetrahedra. Then, we define the objective function as

$$f(\mathbf{x}) = \sum_{j=1}^{m} \eta_{E_j}(\boldsymbol{J\phi}), \tag{9}$$

where E_j is the j^{th} element adjacent to the node. The derivatives of the objective function are evaluated analytically according to [32].

3.7.3. Objective Function for Surface and Edge Nodes We need to ensure that edge and surface nodes in the optimized mesh are located on the corresponding entities they belong to. Therefore, we define a new objective function that evaluates the regularized distortion measure of the adjacent tetrahedra in terms of the parametric coordinates of the node, see [20].

For surface nodes, the corresponding objective function becomes

$$f_{\boldsymbol{\varphi}}(\mathbf{u}) = f \circ \boldsymbol{\varphi}(\mathbf{u}), \tag{10}$$

where $\boldsymbol{\varphi}$ and $\mathbf{u}$ are the parameterization and the parametric coordinates of surface $\mathcal{S}$.

Each edge of the model is parameterized using the arc-length parameter, $\boldsymbol{\gamma}(s)$. Thus the objective function for edge nodes is expressed using the parameterization $\boldsymbol{\gamma}(s)$ as

$$f_{\boldsymbol{\gamma}}(s) = f \circ \boldsymbol{\gamma}(s). \tag{11}$$

Note that, the derivatives involved in the optimization process for surface and edge nodes are computed numerically due to the linear piece-wise definition of $\boldsymbol{\varphi}$ and $\boldsymbol{\gamma}$.

We highlight that in the proposed approach, we do not use a distortion measure for the surface triangles, instead we only take into account the quality of the tetrahedral elements. That is, we are computing a valid and high-quality volumetric mapping between the computational domain and the physical domain, such that the surface and edge nodes are constrained to move along the corresponding entities.

3.7.4. Optimization Approach The optimization approach is devised as a Gauss-Seidel iterative process moving one node at a time. That is, for each node we modify its position while keeping fixed the position of all the other nodes. The new position is computed by optimizing the corresponding objective function of the node. This optimization is performed using a line-search method in which the search direction is computed using Newton's method, and the step length is computed using the strong Wolfe conditions, see details in [33]. Algorithm 2 details the proposed untangling and smoothing process.

Algorithm 2 Procedure to optimize a mesh, $\mathcal{M}$.

```
1:  function optimizeMesh(Mesh M, Real maxDisp)
2:      Boolean isConverged ← false
3:      while not isConverged do
4:          Real disp ← 0
5:          for each free node, n do
6:              Function g ← getObjectiveFuncion(n)      ▷ get edge, surface or
                                                         ▷ inner objective function
7:              Real nodeDisp ← smoothNode(n,g)
8:              disp ← max{disp, nodeDisp}
9:          end for
10:         isConverged ← (disp ≤ maxDisp)
11:     end while
12: end function
```

4. EXAMPLES

In this section the presented method is applied to three different geometries. The first geometry is an academic example where a curved surface is inserted in a cube. The two last examples are taken from realistic cases where the insertion of the surface in the mesh is needed. In all the examples, we colour the elements according to their shape quality, see Equation (7). In addition, for all of them, we present a table that summarizes the shape quality statistics of the mesh elements. Specifically, we provide the number of tangled elements, and the minimum, maximum, mean and standard deviation of the element quality. We highlight that in all cases the final mesh is a valid high-quality mesh, where the insertion of the surfaces has not drastically decreased the quality of the initial mesh.

4.1. Simple Cube

The first example deals with the insertion of a sinusoidal surface inside a tetrahedral mesh for a cube, $\mathcal{M}$, see Figure 8a. Figure 8b shows the refined tetrahedral mesh and the initial approximating surface, $\mathcal{S}_{\mathcal{M}}$, extracted from triangles of $\mathcal{M}$. Note that at this stage, the candidate triangles to approximate the surface are parallel to the coordinate planes of the computational domain. Then, the nodes of $\mathcal{S}_{\mathcal{M}}$ are projected onto the immersed surface, $\mathcal{S}$, see Figure 8c. Low-quality and inverted elements appear near the projected surface, due to the movement of the nodes. Finally, the whole mesh is optimized using the presented optimization method in order to repair the tangled elements and improve the overall mesh quality, see Figure 8d. It is important to remark that the smoother moves all the interior nodes of the mesh, including the ones that are located on $\mathcal{S}_{\mathcal{M}}$. In this case, such nodes are able to slide along the surface $\mathcal{S}$. This is a key point in order to obtain a valid and high-quality tetrahedral mesh, specially around the high-curvature areas of the inserted surface, where the feasible region is small.

Figures 9a, 9b, and 9c present the distribution of the elements according to their shape quality for the mesh before inserting the surface (Figure 8a), the mesh once the nodes are projected onto the surface (Figure 8c), and the mesh after applying the proposed smoother (Figure 8d). In this example, all the elements before inserting the surface have a quality of 1.0. The projection of nodes introduces low-quality and inverted elements. Then, the smoothing process increases the overall mesh quality. Table I details the mesh quality statistics. As it has been pointed out, all the elements of the initial mesh are of the highest quality. The projection process has introduced 58 inverted elements and therefore, the minimum quality is 0. Although the maximum quality is 1, this mesh is not valid for any simulation process. Once the mesh is optimized, there are not any inverted elements, and the minimum and mean quality are increased to 0.35 and 0.92, respectively. The maximum quality is decreased to 0.99 in order to accommodate the displacement of the nodes during the optimization process. Also, the standard deviation is decreased by a factor of 3. The final mesh contains elements

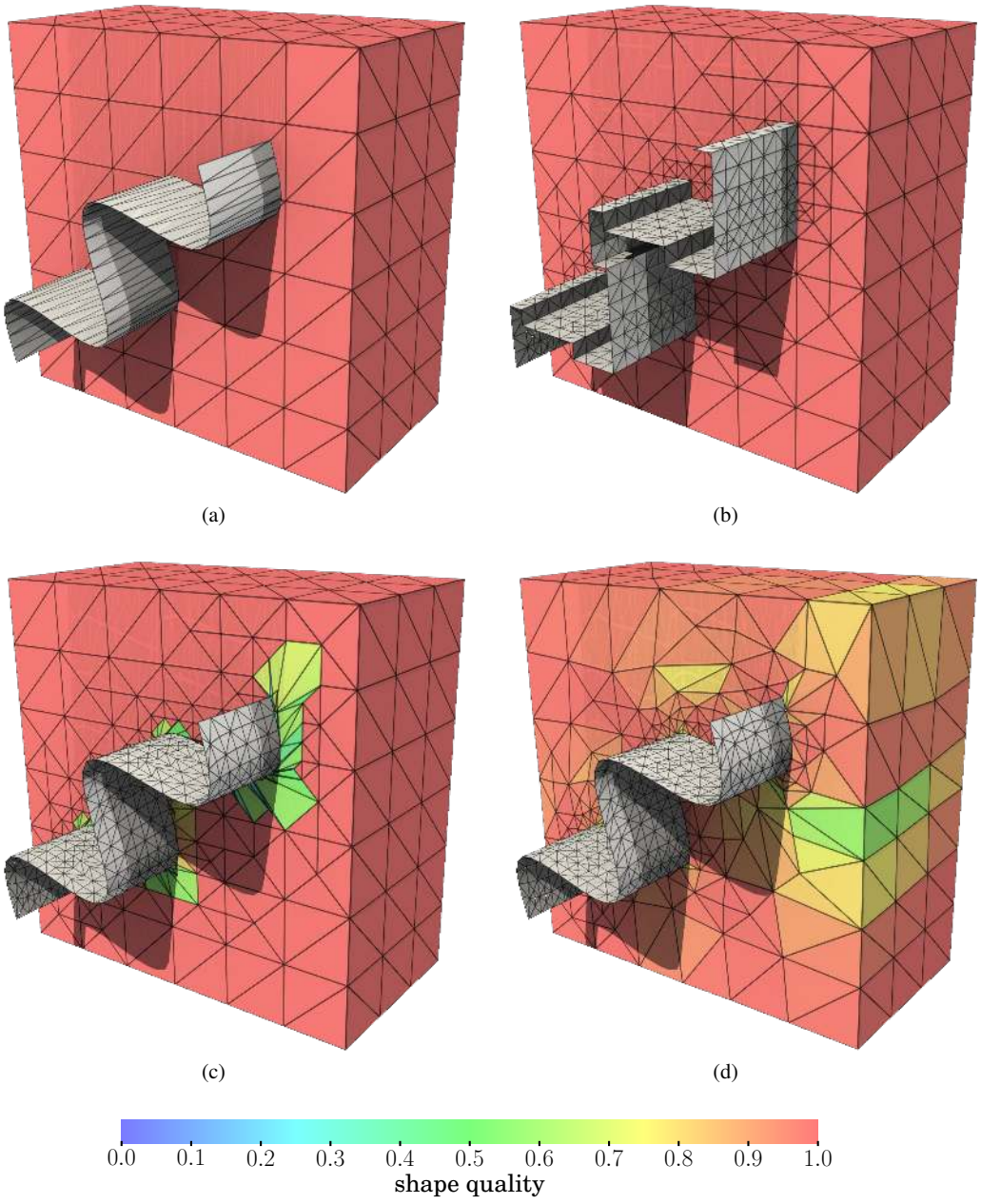
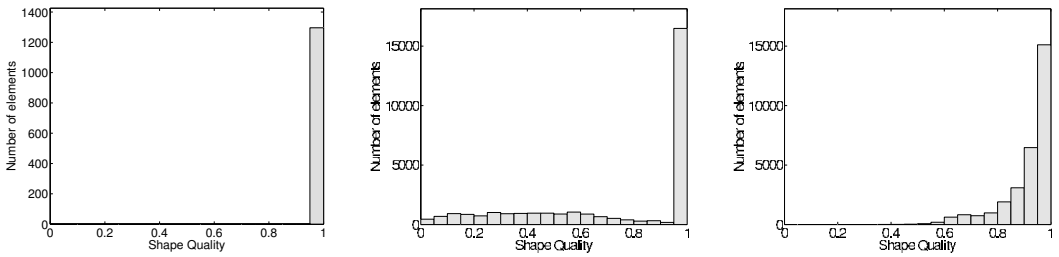


Figure 8. Approximation of an immersed sinusoidal surface, $\mathcal{S}$, in a tetrahedral mesh for a cube: (a) initial tetrahedral mesh and surface to be inserted; (b) refined tetrahedral mesh and initial $\mathcal{S}_{\mathcal{M}}$ extracted from triangular faces of $\mathcal{M}$; (c) nodes of $\mathcal{S}_{\mathcal{M}}$ projected onto $\mathcal{S}$; and (d) smoothed final mesh.

with lower quality than the initial mesh. In this example, this is expected since the initial mesh is entirely composed of ideal elements.

4.2. Faults Insertion

The second example deals with the insertion of fault surfaces into a discretized sub-soil model, see Figure 10. The tetrahedra of the mesh are not able to resolve the geometric features of the inserted



(a) (b) (c)

Figure 9. Mesh quality histogram for the approximation of an immersed sinusoidal surface: (a) initial mesh; (b) mesh with immersed surfaces before optimization; and (c) mesh with immersed surfaces after optimization.

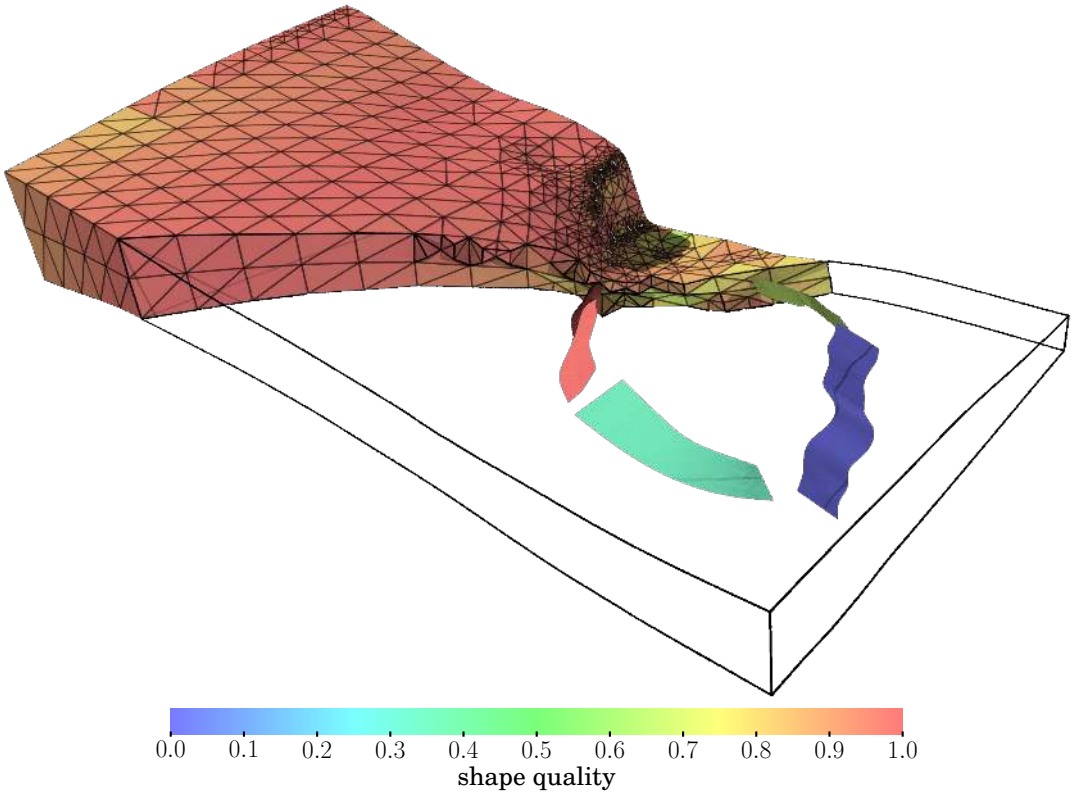


Figure 10. Initial tetrahedral mesh for a sub-soil model, elements coloured according their quality, and the four faults to be inserted.

surface. This example shows that the proposed method does not depend on the element size of the initial mesh. Figure 11 shows the different stages of the insertion surface algorithm for the last inserted fault (red surface in Figure 10). First, we refine the mesh and select the candidate triangles to approximate the surface, Figure 11a. Then, we apply the proposed projection algorithm to map the selected nodes onto the inserted surface, Figure 11b. This step generates inverted and low-quality elements. Thus, we apply the presented untangling and smoothing algorithm to repair the inverted elements and increase the overall mesh quality, see Figure 11c.

Figures 12a, 12b, and 12c show the quality histogram of the mesh before inserting the surface (Figure 11a), the mesh once the nodes are projected onto the surface (Figure 11b), and the mesh

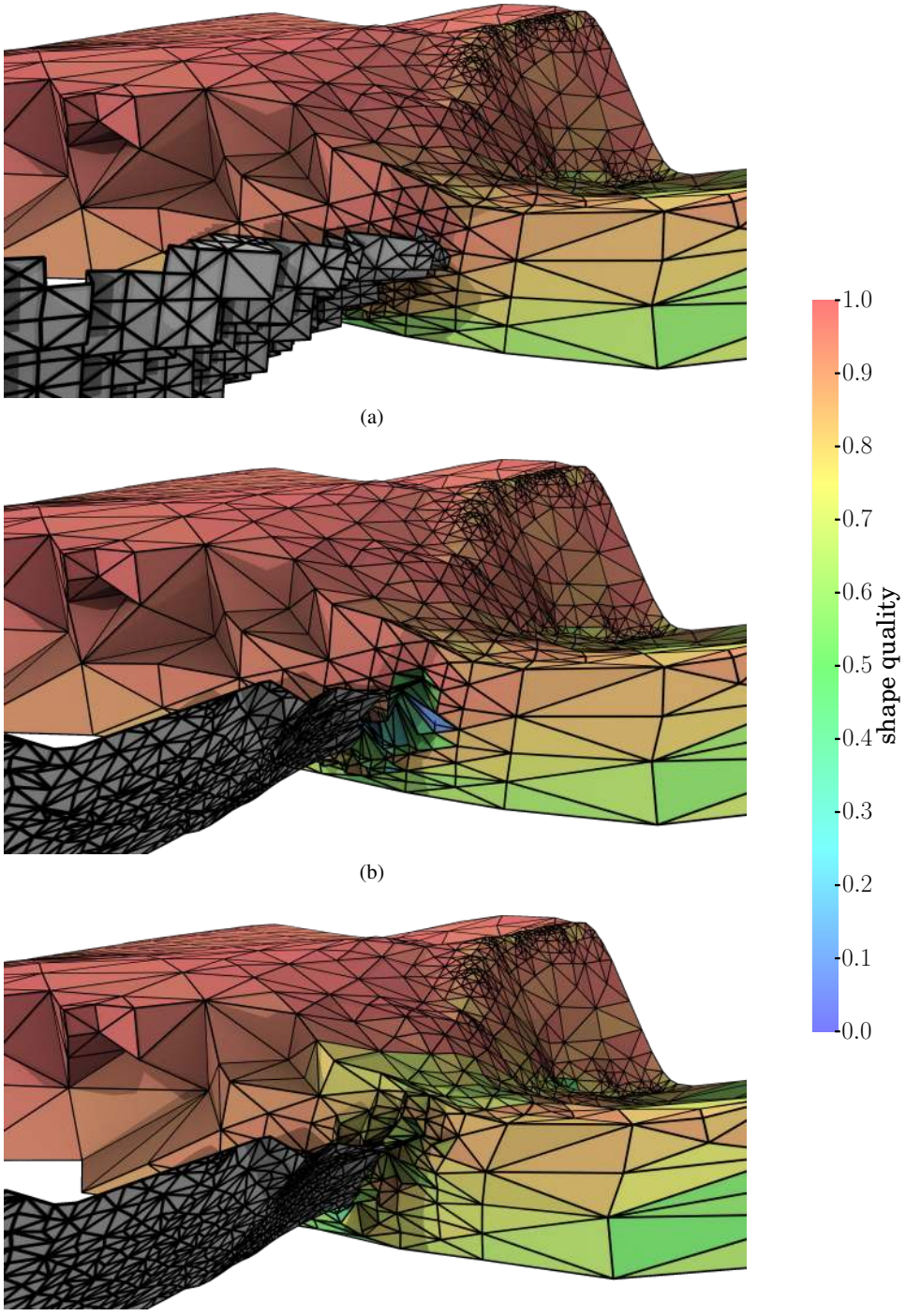


Figure 11. Approximation of the fourth immersed faults, $\mathcal{S}$, in a tetrahedral mesh for a sub-soil model: (a) refined tetrahedral mesh and initial $\mathcal{S}_{\mathcal{M}}$ extracted from triangular faces of $\mathcal{M}$; (b) nodes of $\mathcal{S}_{\mathcal{M}}$ projected onto $\mathcal{S}$; and (c) smoothed final mesh.

after applying the proposed smoother (Figure 11c). The histogram is similar for the first and third meshes where the number of elements increase with the quality, being the more frequent qualities the highest interval. The main differences is the dispersion of the qualities. The qualities in the

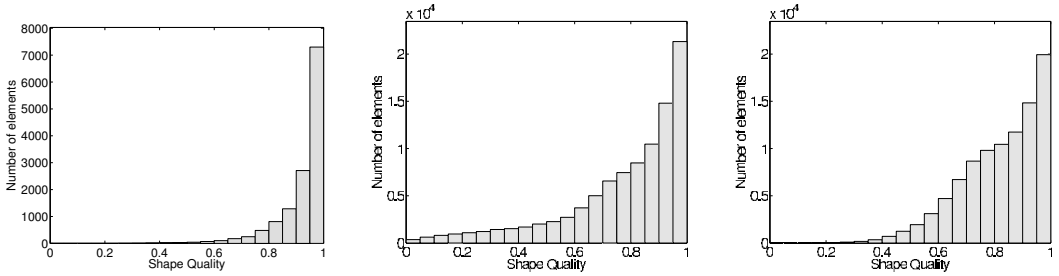


Figure 12. Mesh quality histogram for the approximation of the fourth immersed faults: (a) initial mesh; (b) mesh with immersed surfaces before optimization; and (c) mesh with immersed surfaces after optimization.

initial mesh are more concentrated towards the highest values whilst the qualities in the final mesh are more dispersed. Figure 12b shows the quality decay due to the projection process where some elements are not valid. Nevertheless, the applied smoother increases the overall mesh quality. These remarks can be observed quantitatively in Table I. Comparing the initial and final mesh quality statistics, both the minimum and mean qualities have decreased—from 0.26 to 0.13, and from 0.92 to 0.81 respectively—and the standard deviation has increased—from 0.09 to 0.14. These results agree with the qualitative observations. The optimization improvement is clear from comparing the projected and final statistics; the minimum and mean quality have increased from 0 to 0.13, and from 0.77 to 0.81, respectively, and the standard deviation has decreased from 0.21 to 0.14.

4.3. Atmospheric Layer Insertion

In several atmospheric applications it is usual to create a base mesh and then adapt it to each specific simulation. In this example, we will use a given tetrahedral mesh of the East zone of the Gran Canaria island (Spain) to simulate the transport and reaction of pollutants in the atmosphere. In particular, we are interested in the distribution of pollutant concentration on two surfaces that are an extrusion of the orography of the terrain. Therefore, we will use the proposed method to insert them and we will show that the resulting mesh is valid for simulation purposes.

4.3.1. Mesh generation Figure 13 shows the orography of the considered area in the East part of Gran Canaria island and the immersed surfaces. They are located at 1000 and 2000 meters above the terrain. Similar to the previous examples, Figure 14 illustrates the different stages of the insertion surface algorithm. In this figure, the inserted surface is the highest one (red in Figure 13). Figure 14a shows the refinement and selection step, Figure 14b presents the projection of the selected mesh onto the inserted surface where low-quality and tangled elements appear; and finally the resulting mesh, after the untangling and smoothing algorithm, see Figure 14c.

Figure 15 shows the quality histogram of the mesh before inserting the second surface, once the nodes are projected onto the second surface (Figure 14b), and after applying the proposed smoother (Figure 14c). In this case, comparing the initial and final meshes, the dispersion of the qualities has increased and the minimum quality has decreased. This observation agrees with the faults insertion example, although in this case the differences are smaller. Regarding the projected and final meshes, we highlight that the optimization process has repaired all the tangled elements, and the minimum quality has improved. The main statistical values of the mesh quality are detailed in Table I. Comparing the initial (before inserting any surface) and final meshes, we observe that the minimum quality has decreased from 0.58 to 0.32, that the maximum and mean qualities have detracted slightly from 1 to 0.99 and from 0.97 to 0.95 respectively, and that the standard deviation has increased from 0.03 to 0.08. As in the previous examples, the optimization process has improved the quality repairing 6585 non-valid elements, increasing the minimum and mean quality from 0 to 0.32 and from 0.89 to 0.95 respectively; and decreasing the standard deviation from 0.22 to 0.08.

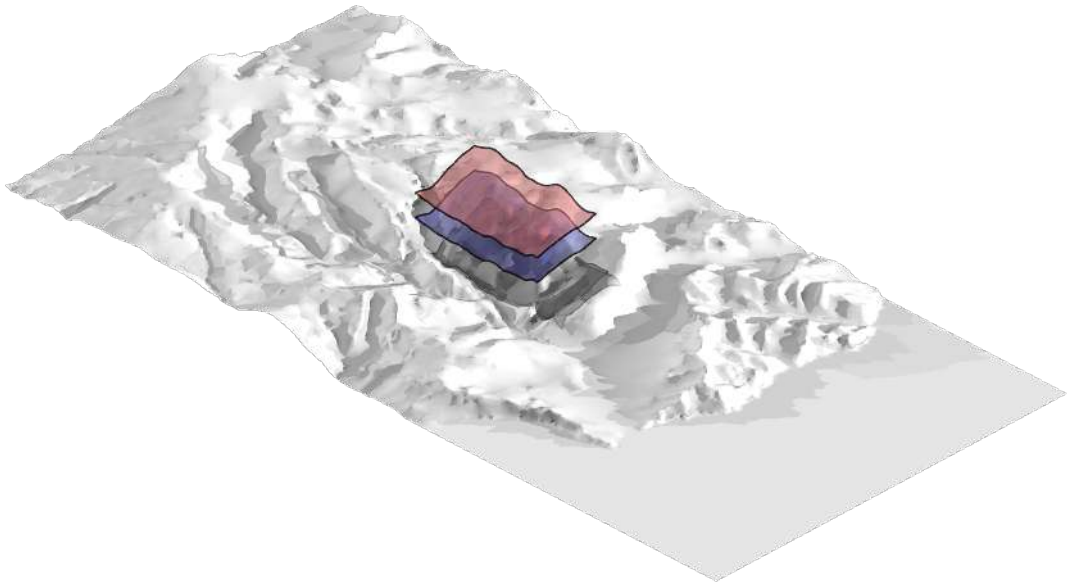


Figure 13. East zone of Gran Canaria island and the two surfaces to be inserted.

4.3.2. Transport and reaction of pollutants Once the mesh with the surfaces is generated we simulate the transport and reaction of pollutants. In this case, we assume that a pollutant leak from a ship in the sea is transported into the island by a wind field. We know that the general direction of the wind field is from East to West (from sea to the island). To generate a more realistic wind field we use a mass-consistent wind model [34, 35] that assumes constant density and verifies the continuity equation (mass conservation) in the domain and impermeability on the terrain.

Using the resulting wind field, we simulate the transport and reaction of pollutants. A finite element model designed to simulate the microscale [36, 37] is used in this example. This model uses a finite element method stabilized with least squares [38] to prevent the numerical oscillations that arise with the classical Galerkin formulation. The resulting linear system has a symmetric matrix and we solve it using the conjugate gradient method preconditioned with an incomplete Cholesky factorization density type [39, 40].

The prescribed boundary conditions for the pollutant concentration are the following: a Dirichlet condition of $0.5\,\mathrm{g\,m^{-3}}$ at a region of the sea boundary, a null Dirichlet condition in the inflow boundary—we assume that the air outside the area of interest is clean, a natural Neumann condition on the outflow boundaries, and a dry deposition velocity of $0.5\,\mathrm{m\,s^{-1}}$ in the terrain. We use typical diffusion values with a horizontal diffusion of $10\,\mathrm{m^2\,s^{-1}}$ and a vertical of $5\,\mathrm{m^2\,s^{-1}}$.

Figure 16 shows the resulting wind field and the isosurfaces of pollutant concentration after $5\,\mathrm{min}$, $30\,\mathrm{min}$ and $60\,\mathrm{min}$. Note that the impermeability condition in the terrain channels the wind field into the valleys and circle the mountains. Looking at the pollutant concentrations, we observe that the front of the plume is transported along the domain and the smaller concentrations flow up to the valleys when the pollutant plume evolves. To highlight the properties of the surface insertion algorithm, we have not applied an automatic mesh refinement to capture the plume front as suggested in [41].

The output of interest is the distribution of the pollutant on the inserted surfaces. Figure 17a shows the concentration after 30 minutes—it corresponds to Figure 16b. At this moment, the front of the $0.5\,\mathrm{g\,m^{-3}}$ concentration is crossing the bottom surface. Note that the pollutant concentration is smaller on the top layer. Figure 17b shows the distribution after one hour when the plume has developed completely—it corresponds to Figure 16c. At this stage, we observe that the distribution

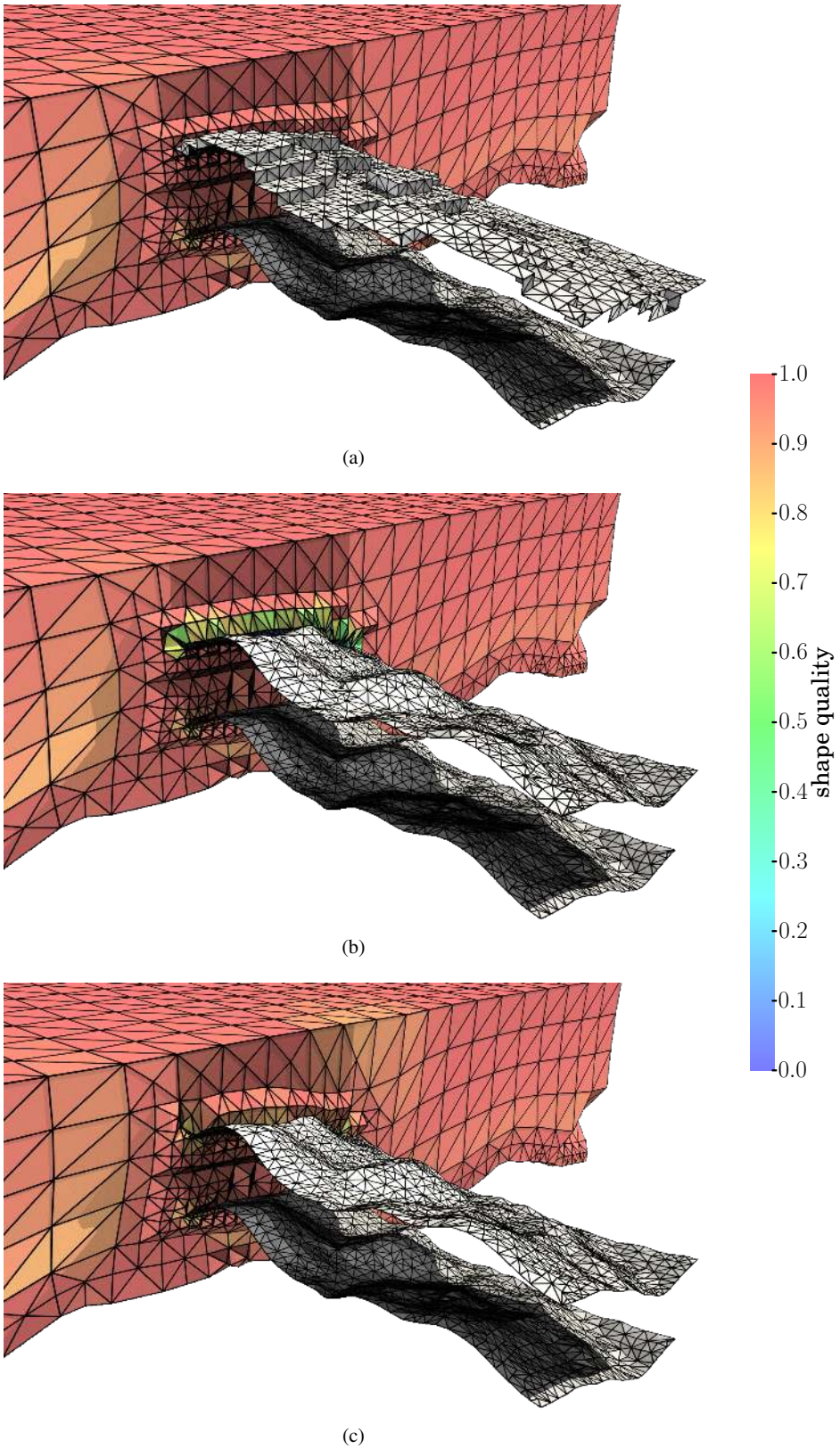


Figure 14. Approximation of the second immersed surface in the atmospheric simulation, $\mathcal{S}$, in a tetrahedral mesh for an air quality model: (a) refined tetrahedral mesh and initial $\mathcal{S}_{\mathcal{M}}$ extracted from triangular faces of $\mathcal{M}$; (b) nodes of $\mathcal{S}_{\mathcal{M}}$ projected onto $\mathcal{S}$; and (c) smoothed final mesh.

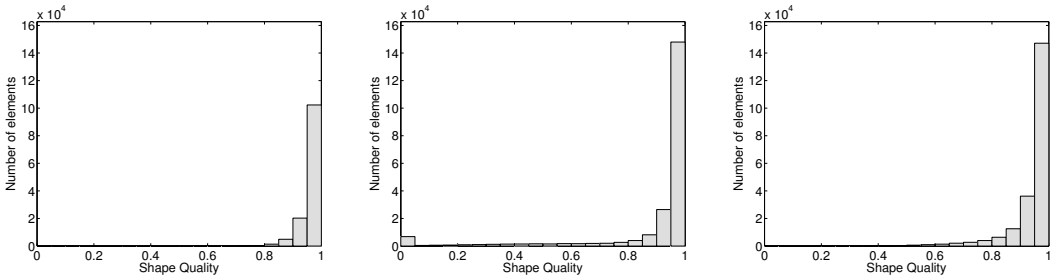


Figure 15. Mesh quality histogram for the approximation of two immersed surfaces in the atmospheric simulation: (a) initial mesh; (b) mesh with immersed surfaces before optimization; and (c) mesh with immersed surfaces after optimization.

Table I. Element quality statistics of the presented high-order meshes.

	sinusoidal			sub-soil fault			atmospheric layer		
	original	projected	smoothed	original	projected	smoothed	original	projected	smoothed
inverted	0	58	0	0	1	0	0	6585	0
min	1	0	0.35	0.26	0	0.13	0.58	0	0.32
max	1	1	0.99	0.99	0.99	0.99	1.00	0.99	0.99
mean	1	0.74	0.92	0.92	0.77	0.81	0.97	0.89	0.95
std dev	0	0.33	0.10	0.09	0.21	0.14	0.03	0.22	0.08

on both surfaces is similar, although the concentration values are smaller on the top surface due to the distance to the centre of the plume and wind direction.

This example has shown the applicability of the proposed surface insertion method where the quantity of interest is on the immersed surface.

5. CONCLUSIONS

In this work, we propose a novel approach to insert several triangulated surfaces into an existing tetrahedral meccano mesh. The inserted surfaces are approximated using triangles extracted from the tetrahedral mesh. To properly approximate the geometric features of each inserted surface, we propose to apply a refinement process to the tetrahedral mesh. Thus, we avoid the dependence of the proposed algorithm on the element size of the initial tetrahedral mesh. Then, we select the candidate triangles and project them onto each surface using a novel projection algorithm based on the Floater parameterization. The key idea is to parameterize the initial surface and the approximating surface with the same parametric space. Finally, an untangling and smoothing technique is applied to repair the tangled elements and improve the overall mesh quality. We move the nodes on the edges and surfaces (immersed and boundary surfaces) taking into account only the quality of the tetrahedral elements. To this end, we express the objective function in terms of the parametric coordinates of the moving edge and surface nodes. We have shown that the proposed method generates high-quality meshes that can be used in numerical simulations.

The implementation of the untangling and smoothing process is performed by updating the position of one node at a time. That is, we use a Gauss-Seidel iterative process to simplify the implementation and reduce the memory footprint of the optimization process. However, it could potentially take a high number of iterations to converge in a large mesh. To solve this issue, it is possible to implement a parallel version of the untangling and smoothing process in which several nodes of the mesh are moved at the same time [42]. Nevertheless, it is needed to ensure that the nodes that are moved simultaneously are not in the same elements to avoid using information that is being updated. Although we do not have a proof of the convergence of the proposed method, in

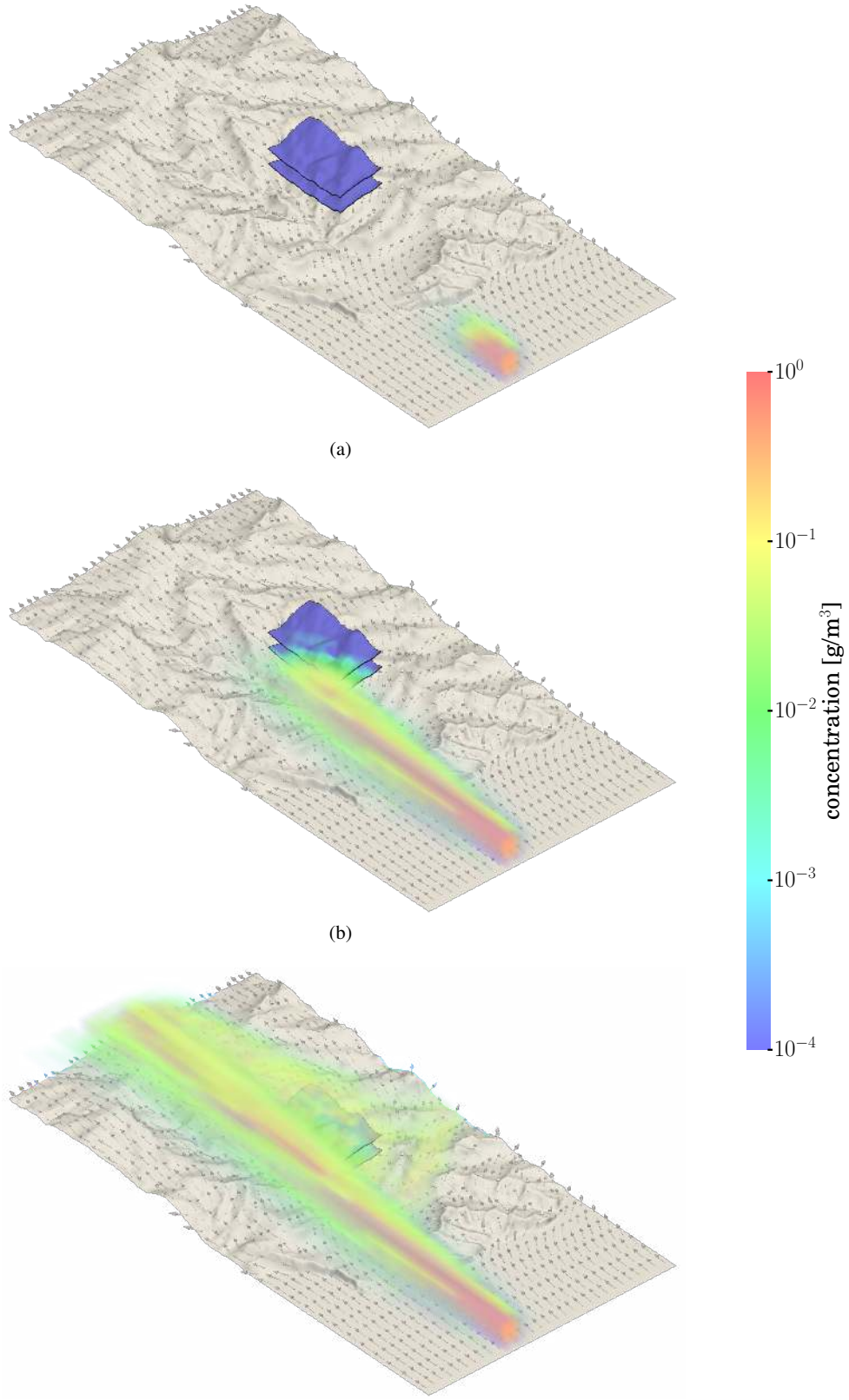


Figure 16. Slice of the isosurfaces of pollutant concentration after (a) five minutes; (b) thirty minutes; and (c) one hour.

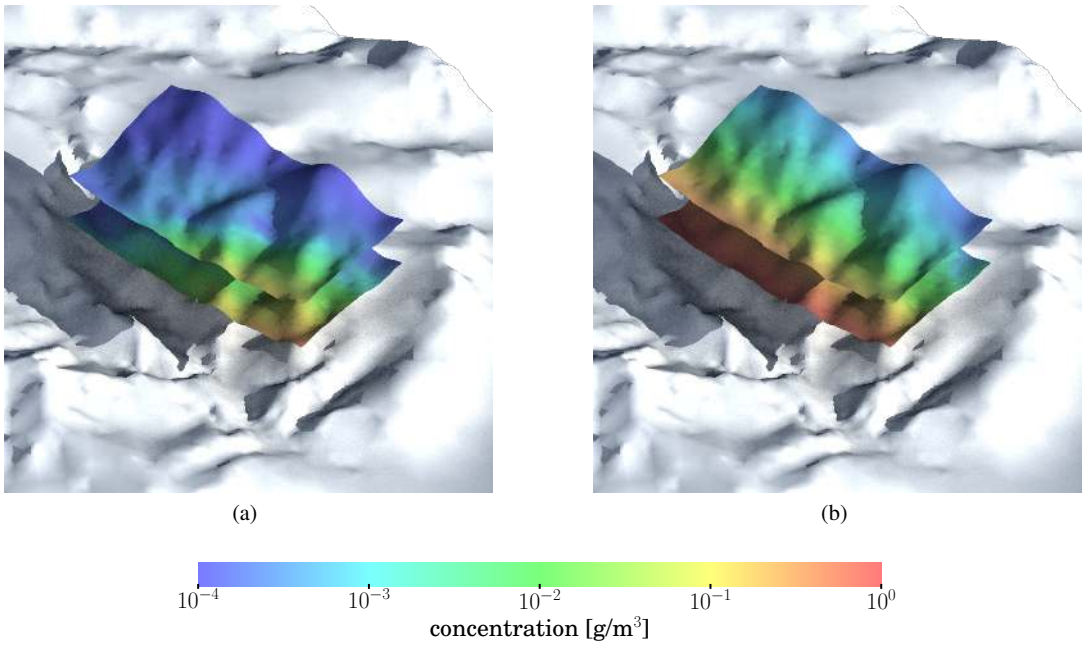


Figure 17. Pollutant concentration distribution in the inserted surfaces after (a) thirty minutes; and (b) one hour.

all the cases that we have tested the optimization-based untangling and smoother, we have obtained a valid mesh of high-quality elements.

The proposed technique to introduce a surface in a tetrahedral mesh can be applied to any mesh generated by an algorithm that satisfies the following requisites. First, it is needed a structured hexahedral mesh, subdivided into tetrahedra. Second, the tetrahedra can be refined to adapt the mesh by using the Kossaczký process. Third, a volumetric parameterization is needed. Nevertheless, in the future work, we will reduce the requirements to introduce a given surface.

Moreover, while the presented algorithm has been developed for inserting triangulated surfaces, it can deal with parameterized surfaces coming from a CAD model. The only restriction at the current version of the method is that the CAD surface has to be simply-connected (no holes) and non-trimmed, in order to define a quadrilateral parametric domain.

Several aspects of the presented method can be improved in the next future in order to deal with more complex situations. First, the proposed algorithm does not deal with the insertion of intersecting surfaces, because we do not prescribe the approximation of the intersection between these surfaces. Moreover, since we are using a square parametric domain to perform the projection and smoothing process, we require that the surfaces to be inserted have to be simply-connected surfaces with boundary. Nevertheless, one possible solution is to divide the initial surfaces into simply-connected patches with boundary.

REFERENCES

1. Munekazu Ohno and Kiyotaka Matsuura. Quantitative phase-field modeling for dilute alloy solidification involving diffusion in the solid. *Physical Review E*, 79(3):031603, 2009.
2. Elie Hachem, S Feghali, Thierry Coupez, and Ramon Codina. A three-field stabilized finite element method for fluid-structure interaction: elastic solid and rigid body limit. *International Journal for Numerical Methods in Engineering*, 104(7):566–584, 2015.
3. Fatemeh Amiri, Daniel Millán, Marino Arroyo, Mohammad Silani, and Timon Rabczuk. Fourth order phase-field model for local max-ent approximants applied to crack propagation. *Computer Methods in Applied Mechanics and Engineering*, 312:254–275, 2016.

4. N Moës, J Dolbow, and T Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999.
5. Sergio Zlotnik and Pedro Díez. Hierarchical x-fem for n-phase flow (n¿ 2). *Computer Methods in Applied Mechanics and Engineering*, 198(30):2329–2338, 2009.
6. E. Tamayo-Mas and A. Rodríguez-Ferran. A medial-axis-based model for propagating cracks in a regularised bulk. *International Journal for Numerical Methods in Engineering*, 101(7):489–520, 2015.
7. R Rangarajan and AJ Lew. Universal meshes: A new paradigm for computing with nonconforming triangulations, Jan. 2012.
8. Ramsharan Rangarajan and Adrián J. Lew. Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes. *International Journal for Numerical Methods in Engineering*, 98(4):236–264, Mar. 2014.
9. Ramsharan Rangarajan, Maurizio M Chiaramonte, Michael J Hunsweck, Yongxing Shen, and Adrian J Lew. Simulating curvilinear crack propagation in two dimensions with universal meshes. *International Journal for Numerical Methods in Engineering*, 102(3-4):632–670, 2015.
10. Daniel W. Zaide and Carl F. Ollivier-Gooch. Inserting a curve into an existing two dimensional unstructured mesh. In *Proceedings of the 22nd International Meshing Roundtable*, pages 93–107. 2014.
11. Daniel W Zaide and Carl F Ollivier-Gooch. Inserting a surface into an existing unstructured mesh. *International Journal for Numerical Methods in Engineering*, 106:484–500, 2015.
12. Aldo Bonfiglioli, Marco Grottadaurea, Renato Paciorri, and Filippo Sabetta. An unstructured, three-dimensional, shock-fitting solver for hypersonic flows. *Computers & Fluids*, 73:162–174, 2013.
13. Rafael Montenegro, José Manuel Cascón, José María Escobar, Eduardo Rodríguez, and Gustavo Montero. An automatic strategy for adaptive tetrahedral mesh generation. *Applied Numerical Mathematics*, 59(9):2203–2217, 2009.
14. Rafael Montenegro, José Manuel Cascón, José María Escobar, Eduardo Rodríguez, and Gustavo Montero. The meccano method for simultaneous volume parametrization and mesh generation of complex solids. *IOP Conference Series: Materials Science and Engineering*, 10(1):012–018, 2010.
15. Rafael Montenegro, José Manuel Cascón, Eduardo Rodríguez, José Marı Escobar, and Gustavo Montero. The meccano method for automatic three-dimensional triangulation and volume parametrization of complex solids. In B.H.V. Topping, J.M. Adam, F.J. Pallarés, R. Bru, and M.L. Romero, editors, *Developments and Applications in Engineering Computational Technology*, chapter 2, pages 19–48. Saxe-Coburg Publications, Stirlingshire, seventh edition, 2010.
16. José M. Cascón, Eduardo Rodríguez, José M. Escobar, and Rafael Montenegro. Comparison of the meccano method with standard mesh generation techniques. *Engineering with Computers*, 31(1):161–174, 2015.
17. Igor Kossaczký. A recursive approach to local mesh refinement in two and three dimensions. *Journal of Computational and Applied Mathematics*, 55(3):275–288, 1994.
18. Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.
19. José María Escobar, Eduardo Rodríguez, Rafael Montenegro, Gustavo Montero, and José María González-Yuste. Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2775–2787, 2003.
20. Eloi Ruiz-Gironés, Xevi Roca, and José Sarrate. High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation. *Computer-Aided Design*, 72:52–64, 2016.
21. Kangkang Hu and Yongjie Jessica Zhang. Centroidal voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering*, 305:405 – 421, 2016.
22. Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. Efficient volumetric polycube-map construction. *Computer Graphics Forum (Pacific Graphics)*, 35(7), 2016.
23. José María Escobar, Eduardo Rodríguez, Rafael Montenegro, Gustavo Montero, and José María González-Yuste. SUS code: simultaneous mesh untangling and smoothing code. http://www.dca.iusiani.ulpgc.es/SUScode, 2010.
24. Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. Polycube-maps. *ACM Trans. Graph.*, 23(3):853–860, August 2004.
25. Xin Li, Xiaohu Guo, Hongyu Wang, Ying He, Xianfeng Gu, and Hong Qin. Harmonic volumetric mapping for solid modeling applications. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, pages 109–120, New York, NY, USA, 2007. ACM.
26. James Gregson, Alla Sheffer, and Eugene Zhang. All-hex mesh generation via volumetric PolyCube deformation. *Computer Graphics Forum*, 30(5):1407–1416, aug 2011.
27. JM Cascon, R Montenegro, JM Escobar, E Rodriguez, and G Montero. A new meccano technique for adaptive 3-d triangulations. In *Proceedings of the 16th International Meshing Roundtable*, pages 103–120, Berlin, Germany, 2007. Springer-Verlag.
28. Patrick M. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, Jan. 2001.
29. VA Garanzha and IE Kaporin. Regularization of the barrier variational method. *Computational Mathematics and Mathematical Physics*, 39(9):1426–1440, 1999.
30. A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 103(5):342–363, 2015.
31. Abel Gargallo-Peiró, Xevi Roca, Jaume Peraire, and Josep Sarrate. A distortion measure to validate and generate curved high-order meshes on CAD surfaces with independence of parameterization. *International Journal for Numerical Methods in Engineering*, 106(13):1100–1130, 2015.
32. Eloi Ruiz-Gironés, Xevi Roca, Josep Sarrate, Rafael Montenegro, and José María Escobar. Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework. *Advances in Engineering Software*, 80:12–24, 2015.

33. J. E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial & Applied Mathematics (SIAM), Jan. 1996.
34. Luis Ferragut, Rafael Montenegro, Gustavo Montero, Eduardo Rodríguez, M.L. Asensio, and José María Escobar. Comparison between 2.5-D and 3-D realistic models for wind field adjustment. *Journal of Wind Engineering and Industrial Aerodynamics*, 98(10-11):548–558, 2010.
35. Albert Oliver, Eduardo Rodríguez, José María Escobar, Gustavo Montero, Mariano Hortal, Javier Calvo, José Manuel Cascón, and Rafael Montenegro. Wind forecasting based on the HARMONIE model and adaptive finite elements. *Pure and Applied Geophysics*, 172(1):109–120, Jan 2015.
36. Albert Oliver, Gustavo Montero, Rafael Montenegro, Eduardo Rodríguez, José María Escobar, and Agustí Pérez-Foguet. Finite element simulation of a local scale air quality model over complex terrain. *Advances in Science and Research*, 8:105–113, 2012.
37. Albert Oliver, Gustavo Montero, Rafael Montenegro, Eduardo Rodríguez, José María Escobar, and Agustí Pérez-Foguet. Adaptive finite element simulation of stack pollutant emissions over complex terrains. *Energy*, 49(0):47–60, 2013.
38. Boi-nan Jiang. *The Least-Squares Finite Element Method*. Springer Berlin Heidelberg, 1998.
39. Chih-Jen Lin and Jorge J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM Journal on Scientific Computing*, 21(1):24–45, 1999.
40. A. Rodríguez-Ferran and M.L. Sandoval. Numerical performance of incomplete factorizations for 3D transient convection–diffusion problems. *Advances in Engineering Software*, 38(6):439–450, jun 2007.
41. Lluís Monforte and Agustí Pérez-Foguet. Esquema adaptativo para problemas tridimensionales de convección-difusión. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 30(1):60–67, Jan 2014.
42. Domingo Benítez, Eduardo Rodríguez, José María Escobar, and Rafael Montenegro. Performance evaluation of a parallel algorithm for simultaneous untangling and smoothing of tetrahedral meshes. In *Proceedings of the 22nd International Meshing Roundtable*, pages 579–598. Springer, 2014.