Adaptive deformation of 3D unstructured meshes with curved body fitted boundaries with application to unsteady compressible flows

Luca Cirrottola[a], Mario Ricchiuto[a], Algiane Froehly[b], Barbara Re[c], Alberto Guardone[d], Giuseppe Quaranta[d]

[a] *Inria Bordeaux Sud-Ouest, Team CARDAMOM, F-33405 Talence, France*
[b] *Inria Direction Générale Déléguée à l'Innovation, France*
[c] *Institute of Mathematics, University of Zürich, 8057 Zürich, Switzerland*
[d] *Department of Aerospace Science and Technology, Politecnico di Milano, 20156 Milano, Italy*

Abstract

We present an adaptive moving mesh method for unstructured meshes which is a three-dimensional extension of the previous works of Ceniceros et al. [9], Tang et al. [38] and Chen et al. [10]. The iterative solution of a variable diffusion Laplacian model on the reference domain is used to adapt the mesh to moving sharp solution fronts while imposing slip conditions for the displacements on curved boundary surfaces. To this aim, we present an approach to project the nodes on a given curved geometry, as well as an a-posteriori limiter for the nodal displacements developed to guarantee the validity of the adapted mesh also over non-convex curved boundaries with singularities.

We validate the method on analytical test cases, and we show its application to two and three-dimensional unsteady compressible flows by coupling it to a second order conservative Arbitrary Lagrangian-Eulerian flow solver.

Keywords: Constant-connectiity mesh adaptation, Unstructured meshes, Unsteady compressible flows, Conservative formulations

1. Introduction

Mesh adaptation is a powerful tool to improve the representation of complex fields for a given computational expense. In computational fluid dynamics in particular, adaptation has become nowadays a customary tool [35]. Adapting the mesh also has a relative computational overhead, which motivates the quest for efficient and robust methods.

Techniques improving the discrete representation of the fields of interest by inserting and removing mesh entities (so called h-adaptation) have proven to be quite mature [35]. However, solution transfer between meshes with different topologies may be non-trivial and may have a non-negligible computational cost, especially if conservation constraints need to be satisfied [19, 2, 24, 31, 36].

By constrast, mesh nodes relocation with constant element connectivity (so called r-adaptation), offers the possibility of a minimally intrusive coupling with existing computational mechanics solvers, as no modification of the data structures is required. As h-adaptation methods, they also provide considerable improvement in the quality of the solutions obtained, especially in unsteady simulations of traveling waves, like shock waves and water waves, where uniform refinement would be way too costly. Moreover, with r-adaptation methods devising conservative projections is much simpler. In fact, the preservation of the one–to–one mapping from the old to the new mesh entities allows the easy construction of a conservative remapping [32], or the use of Arbitrary-Lagrangian-Eulerian (ALE) formulations compliant with the Geometric Conservation Law (GCL) [46, 41].

Unfortunately, the preservation of the initial mesh topology undeniably imposes severe restrictions on nodal displacements in order to avoid mesh folding with tangled (i.e. inverted) elements, especially when the boundary exhibits singular points. Moreover, the accuracy attainable for complex solutions is limited by the initial density of mesh nodes, and less finely-tunable than in metric-based h-adaptation [1].

Anyway, the advantages brought by the effortless coupling with external flow solvers and the conservative solution remapping can counterbalance the mesh quality limitation as long as the r-adaptation technique is computationally efficient. Simply put, the error reduction brought by adapting the mesh must offset the computational overhead. A measure of this efficiency can be evaluated by comparing with the cost of a simulation run on a uniformly refined mesh, providing the same resolution of the flow field. Hybrid adaptive approaches combining well timed re-meshing and adaptive deformation at every time step, which are perhaps the ones computationally most appealing, still require the r-adaptation step to perform well.

Extensive reviews of r-adaptation can be found in [30, 8]. We focus here on methods based on the numerical solution of an elliptic partial differential equation for the position of the mesh nodes, often referred to as the mesh PDE. This equation is typically formulated to find a mapping $\boldsymbol{\xi} : \Omega_{\mathbf{x}} \to \Omega_{\boldsymbol{\xi}}$ from the physical domain to a reference (computational) one. This mapping needs to be injective and surjective in order to guarantee that the produced mesh neither folds nor breaks the domain. Historically the Winslow or homogeneous Thompson-Thames-Mastin generator [43, 44] $\Delta_{\mathbf{x}}\boldsymbol{\xi} = \mathbf{0}$ has been the basis for structured boundary-fitted grid generation (see for example the review in [42]) and it has been extended to also adapt the mesh in the domain either by adding source terms to the equation, or through a variable-diffusion approach [51]. A more general formulation of the last method has been given in [15] by means of harmonic maps and extended in [6].

These equations describe the mapping $\boldsymbol{\xi} = \boldsymbol{\xi}(\mathbf{x})$ and need to be inverted for the physical coordinates $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi})$, leading to a nonlinear system of PDEs which is iteratively solved. In order to ease the cost of the iterative solution of the inverted equations, an alternative mesh generator was proposed in [9] based on a variable-diffusion Laplace equations directly formulated in the physical domain for the mapping $\mathbf{x} : \Omega_{\boldsymbol{\xi}} \to \Omega_{\mathbf{x}}$. This generator is not based on a theoretical derivation, but on the observation that the variable-diffusion Laplacian in the reference domain is sufficient to adapt the mesh in the desired regions while the equations, which are still nonlinear, can easily be solved through a relaxation procedure. The efficiency of the method was shown in application to two-dimensional Boussinesq convection on structured grids, and the method was later applied in [38] to hyperbolic conservation laws and extended in [10] to multicomponent flows on two-dimensional unstructured grids. More recently, the same method was applied to the two-dimensional shallow water equations both in Cartesian and spherical coordinates [3, 4, 5].

Robustness to mesh folding in r-adaptation is a delicate matter, similarly critical in the context of mesh deformation related to moving boundaries, curved mesh generation and smoothing (see for example [29, 16, 45, 20, 47]). Obtaining non-singular meshes requires two main conditions to be met. The first is that the continuous map, appropriately modified to account for all boundary conditions, should verify the appropriate conditions as e.g. the non-negativity of the determinant of the deformation Jacobian. Until quite recently, sufficient conditions were known only in the framework of harmonic maps [15, 34]. Recent work by [26], has allowed to prove similar properties for other types of mesh PDEs, as e.g. some of those proposed by Huang [25] or Huang and Russel [27], by resorting to energy arguments borrowed from the theory of gradient flows. The second important aspect is that the discretization used to approximate the mesh PDE should have the appropriate "property preserving" character, so that the fully discrete moving mesh method is also guaranteed to provide non-singular meshes. This is in itself a subject of investigation. It is in general well known that discrete moving mesh methods can lead to mesh tangling even with properly chosen mesh PDEs [15], and the impact of the truncation error is stressed for example in [30]. In the setting of gradient flow maps, geometrical discretizations have been shown in [26] to answer the discrete positive Jacobian requirement. However, even in the last reference, the issue of accounting for complex curved boundaries is overlooked, even though mesh movement along a given surface does not appear to be necessarily a natural boundary condition of the PDEs considered.

In this work we proceed differently. We want to be able to handle domains with boundaries as general as possible in three space dimensions. and propose a relaxation technique embedding a geometrical limiter allowing to achieve this objective. We focus on the simple reference-domain variable-diffusion Laplacian approach originally proposed in [9], however the ideas proposed in this paper can be extended to other mesh PDEs. To the best of the authors' knowledge, very few applications of r-adaptation to three–dimensional meshes are available to date, see for example [33] as well as some simple applications in [26]. The original approach in [9] does not offer theoretical guarantees against folding, although it has been successfully used in many applications with non-convex boundaries. R-adaptation in three dimensions exhibits even stronger limitations than in two dimensions, as sufficiency results for unfolded continuum maps are typically based on requirement of smoothness and convexity of the boundary. These are easily violated especially in application to external flows, where the boundary is not convex and several singularities (like corners and ridges) are present. We have experienced that tangling is a major concern in the three-dimensional extension of these techniques. Smoothing or untangling methods for unstructured meshes already developed in the literature require nontrivial procedures [23, 45].

Our contribution is thus related to a mesh displacement method allowing to guarantee that nodes move and always remain on a given parametrizaion of curved domain boundaries, and for an *a-posteriori* limiter for the nodal displacement which, when embedded in the mesh relaxation iterations, allows to prevent the occurrence of tangled elements, thus enforcing the validity of the discrete mapping while avoiding smoothing procedures. The resulting moving mesh library `Fmg` has been developed on top of the open source platform `Mmg` [11, 12] to exploit, among other things, its built-in cubic Bézier patch representation of complex manifolds.

The paper is organized as follows. We recall the continuous mesh partial differential equations in section §2, while a thorough discussion of their numerical solution is given in section §3, including a-posteriori limiting and projections to obtain a valid mesh satisfying all the boundary conditions, and the application to unsteady simulations. We discuss the validation of the method proposed considering the adaptation w.r.t. analytical functions in two and three space dimensions in section §4, and application to two and three dimensional unsteady compressible flows are discussed in section §5 Finally, conclusions are presented in section 6.

2. Variable-diffusion Laplacian r-adaptation in the reference domain

We focus here on Laplacian-based r-adaptation, which is the mesh PDE currently implemented in the `Fmg` library. However, the ideas proposed in this paper can be immediately extended to other mesh PDEs. We recall here the continuous mesh problem, and in particular we discuss the boundary conditions, as well as the definition of the monitor functions used for adaptation.

Following [10], we look for a mapping $\mathbf{x} : \Omega_{\boldsymbol{\xi}} \to \Omega_{\mathbf{x}}$ from the reference domain $\Omega_{\boldsymbol{\xi}}$ (the original mesh) to the computational domain $\Omega_{\mathbf{x}}$ (the adapted mesh). Within the reference domain, the computational coordinates satisfy the variable-diffusion Laplace equation

$$\boldsymbol{\nabla}_{\boldsymbol{\xi}} \cdot (\omega(\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\xi}} \mathbf{x}) = \mathbf{0} \qquad \text{in } \Omega_{\boldsymbol{\xi}} \tag{1}$$

The above problem is in general a system of coupled non-linear PDEs, which needs to be complemented by appropriate boundary conditions. Nonlinearity is introduced by the monitor function $\omega(\mathbf{x})$ which depends on an external field evaluated in the computational domain. In this work, we have used a classical scalar definition for this quantity, which allows to decouple the equations for the three spatial coordinates.

In particular, given a quantity of interest $f(\mathbf{x})$, the scalar monitor function used in the examples discussed later is evaluated as

$$\omega(\mathbf{x}) = \sqrt{1 + \alpha ||\boldsymbol{\nabla}_{\boldsymbol{\xi}} f(\mathbf{x})||^2_{\gamma_\alpha} + \beta ||\mathbf{H}_{\boldsymbol{\xi}}(f)(\mathbf{x})||^2_{\gamma_\beta} + \tau ||f||^2_{\gamma_\tau}} \tag{2}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\xi}}$ and $\mathbf{H}_{\boldsymbol{\xi}}$ denote the gradient and Hessian computed on the reference domain $\Omega_{\boldsymbol{\xi}}$. Norm $||f||_\gamma$ is defined as

$$||f||_\gamma = \min\left(1, \frac{||f||}{\gamma \max(||f||)}\right). \tag{3}$$

This normalization, already used in [10], allows to introduce some saturation near the norm maximum according to the value of γ. The idea behind this normalization is to spread a little bit the peak values of the function f around the peak locations, in order to filter out small inhomogeinities in the numerical approximation of the sharp fronts of f. The above definition gives the user some control on the behaviour of the spatial mapping via the parameter pairs (α, γ_α), (β, γ_β), (τ, γ_τ). As in the original works [9, 10], these parameter pairs are not related to an error estimate, thus they are determined empirically and they are intrinsically dependent on the application. However, in our experience a short test on a few time steps is sufficient to assess the behaviour of these parameters on the whole simulation time range.

2.1. Boundary conditions

Despite the decoupling of equation (1) into separate scalar equations, even for a scalar monitor function a strong coupling of the coordinate equations may arise through the boundary conditions, especially in domains including general shapes. In particular, we will split the boundary on two parts as $\partial\Omega_{\boldsymbol{\xi}} = \Gamma^D_{\boldsymbol{\xi}} \cup \Gamma^S_{\boldsymbol{\xi}}$. A full set of Dirichlet conditions are imposed on $\Gamma^D_{\boldsymbol{\xi}}$

$$\mathbf{x} = \boldsymbol{\xi} \quad \text{on } \Gamma^D_{\boldsymbol{\xi}} \tag{4}$$

as this is the portion of the boundary that is not allowed to move. Since fixed boundary nodes are not convenient whenever adaptation is performed on flow waves approaching the boundary, we want to limit as much as possible usage of Dirichlet conditions, preferring slip boundary conditions wherever we can formulate them while preserving sharp geometrical features of the boundary. Along the *slip* boundary $\Gamma^S_{\boldsymbol{\xi}}$ the coordinate positions are constrained to move along a given parameterized domain. For manifold surfaces, we assume to have a known parameterization, for example in the form $\gamma^S(\mathbf{x}) = 0$. This provides one position constraint relating the d spatial coordinates. Thus, $d-1$ additional conditions are required, which are here taken as the null normal stress conditions parallel to the local tangent space spanned by $\{\hat{\boldsymbol{\tau}}^S_j\}_{j=1,d-1}$. This gives the boundary conditions:

$$\left.\begin{aligned} \gamma^S(\mathbf{x}) &= 0 \\ \hat{\mathbf{n}}^S \cdot (\omega(\mathbf{x})\boldsymbol{\nabla}\mathbf{x}) \cdot \hat{\boldsymbol{\tau}}^S_j &= 0 \quad \forall j = 1, d-1 \end{aligned}\right. \quad \text{on } \Gamma^S_{\boldsymbol{\xi}} \tag{5}$$

In three dimensions, the slip boundary $\Gamma^S_{\boldsymbol{\xi}}$ can be further generalized as the union of multiple manifolds with one-dimensional boundaries joining them. This leads to intersection curves where two sets of equations 5 should formally be satisfied at the same time, constraining the displacement to happen along the curve tangent direction. Points where multiple intersection curves meet are corner points and no displacement is possible, thus Dirichlet conditions are imposed on (and only on) these points. The approach used here for the numerical enforcement of boundary conditions is discussed in the following section.

3. Discrete equations, a-posteriori limiting, slip on curved boundaries

We discuss here the implementation choices made in the `Fmg` library, namely the discretization of the mesh PDEs, as well as their iterative solution. Both the a-posteriori limiting of the displacement and the implementation of the slip boundary conditions are strongly tied to the relaxation iterations, and for this reason all the steps are discussed in this section. More specifically, to relieve the complexity of satisfying the boundary conditions (5), we formulate the discrete approximation by means of an iterative multiple-corrections procedure embedding the following three elements:

1. A finite element approximation of the variational form of (1) with natural (Neumann) boundary conditions;
2. An a-posteriori limiter for the nodal displacement enforcing local mesh valididy;
3. A boundary correction in the local normal direction to enforce the first of (5) by projecting on the manifold parametrization at hand.

The intertwining of the a-posteriori limiter of the displacement, of the update of the local boundary normals, and of the projection on the parameterized manifold is essential for the proposed approach to provide valid adapted meshes both in the volume and on the boundaries.

Concerning the representation of these, the `Fmg` library we developed makes use of the point-normal curved triangles parameterization proposed in [49], which is based on cubic Bézier patches with quadratically varying normals. For this we leverage the implementation provided by the open source platform `Mmg` [11, 12]. However note that the form of the manifold parametrization is not a necessary ingredient of our method. Other high order approximations can be used.

3.1. Finite element approximation: Dirichlet and natural boundary conditions

The discrete equations are built starting from a linear finite element approximation of the problem embedding natural (Neumann) boundary conditions, which corresponds to the simple variational form

$$\int_{\Omega_{\boldsymbol{\xi}}} \omega(\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\xi}} v \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}} \mathbf{x} \, \mathrm{d}\Omega_{\boldsymbol{\xi}} = \mathbf{0}, \quad \forall v \in H^1(\Omega_{\boldsymbol{\xi}}) \,. \tag{6}$$

Note that this statement satisfies the null normal stress conditions in (5) but neither the remaining one in the system (the belonging to the surface), nor any Dirichlet conditions eventually assigned. Dirichlet conditions are strongly imposed on the solution space, while the enforcement of the belonging to a slip surface will be detailed in the following 3.4) Note also that ω being a scalar quantity, the above equations provide uncoupled nonlinear variational statements for each component of $\mathbf{x}$.

The projection of (6) on the linear finite element space leads to the nonlinear algebraic system

$$\mathbf{K}(\mathbf{x})\mathbf{x}^{\nu} = \mathbf{0}, \quad \nu = 1, \ldots, d \tag{7}$$

having introduced the array of unknown node positions $\mathbf{x}^\nu = [x_i^\nu]$ for each space component ν, with d the number of space dimensions, and where the stiffness matrix has the standard entries

$$K_{ij}(\mathbf{x}) = \int_{\Omega_{\boldsymbol{\xi}}} \omega(\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\xi}} \phi_i \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}} \phi_j \, \mathrm{d}\Omega_{\boldsymbol{\xi}} \tag{8}$$

with $\{\phi_i\}_{i\geq 1}$ the linear base functions spanning the solution space. Please note that (7) is a set of decoupled systems, one for each spatial direction, as shown by the fact that K_{ij} are scalar entries. Note also due to the consistency of the finite element space with Dirichlet conditions, Dirichlet nodes are not included in the above sum.

In practice, by defining the displacement $\boldsymbol{\delta} = \mathbf{x} - \boldsymbol{\xi}$, the system is not written as in (7), but as

$$\mathbf{K}(\mathbf{x})\boldsymbol{\delta}^\nu = -\mathbf{K}(\mathbf{x})\boldsymbol{\xi}^\nu \tag{9}$$

which is better suited for the iterative corrections described in the following sections.

3.2. Scalar correction iterations

We introduce an iterative procedure which, while avoiding mesh tangling in 2D and 3D, and accounting for the directional coupling inherent to (5), retains the scalar structure of the decoupled variational form. Note however that the corrections proposed can be easily adapted to other iterative solution methods (as well as mesh PDEs).

The basic iteration used in our method starts from a standard diagonal Jacobi relaxation to handle the nonlinearity of (9)

$$K_{ii}^{[k]} \boldsymbol{\delta}_i^{[k+1]} = -\sum_{\substack{j\in\mathcal{B}_i \\ j\neq i}} K_{ij}^{[k]} \boldsymbol{\delta}_j^{[k]} - \sum_{j\in\mathcal{B}_i} K_{ij}^{[k]} \boldsymbol{\xi}_j \tag{10}$$

where $\mathcal{B}_i$ denotes the ball[1] of node i and vector $\boldsymbol{\delta}_i^{[k]} = [\boldsymbol{\delta}_i^\nu]_{\nu=1,\dots,d}^{[k]}$ is now the vector made of the space components of the displacement of node i at iteration k (the same notation will be used for vectors $\mathbf{x}_i$ and $\boldsymbol{\xi}_i$). Again, we stress that the above iteration is in fact a set of d relations for the components of the displacement. The matrix entries $K_{ij}^{[k]}$ depend on monitor function ω at iteration k (cf. equation (8)), which in turn depends on the scalar field f evaluated at the actual positions $\mathbf{x}_i^{[k]}$, according to equation (2). In our current implementation, the re-evaluation is performed by linearly interpolating the scalar field f at the current nodal positions $\mathbf{x}_i^{[k]}$ through a standard search algorithm based on barycentric coordinates.

In our implementation we added and removed the term $K_{ii}^{[k]}\boldsymbol{\delta}_i^{[k]}$, to obtain the following iterations

$$\boldsymbol{\delta}_i^{[k+1]} = \boldsymbol{\delta}_i^{[k]} - \frac{1}{K_{ii}^{[k]}} \sum_{j\in\mathcal{B}_i} K_{ij}^{[k]} \mathbf{x}_j^{[k]} \tag{11}$$

[1] In the common lexicon of the mesh generation community, the *ball* $\mathcal{B}_i$ of a mesh node i is defined here as the set of elements sharing vertex i [**?**]. To ease the notation, with a slight abuse of terminology we say that a node j *belongs* to the ball of node i if it is a vertex of an element belonging to the ball, and we denote this statement as $j \in \mathcal{B}_i$.

which are initialized with $\boldsymbol{\delta}_i^{[0]} = \mathbf{0}$.

The last step is the computation of the new nodal positions as follows

$$\mathbf{x}_i^{[k+1]} = \mathbf{x}_i^{[k]} + \widetilde{\Delta\mathbf{x}}_i^{[k+1]}\left(\boldsymbol{\delta}_i^{[k+1]}, \{\mathbf{x}_i^{[k+1]}\}_{j<i}, \{\mathbf{x}_i^{[k]}\}_{j\geq i}\right) \tag{12}$$

where $\widetilde{\Delta\mathbf{x}_i}^{[k+1]}$ are limited increments obtained by a-posteriori correcting $\boldsymbol{\delta}_i^{[k+1]}$ to account for both mesh validity and boundary conditions (both Dirichlet and slip wall). In both cases, these corrections are local, albeit not only dependent on node i, and non-linear w.r.t. $\mathbf{x}$. The nonlinearity is readily handled in the iterations by using the last nodal positions available.

3.3. A-posteriori corrections for mesh validity enforcement

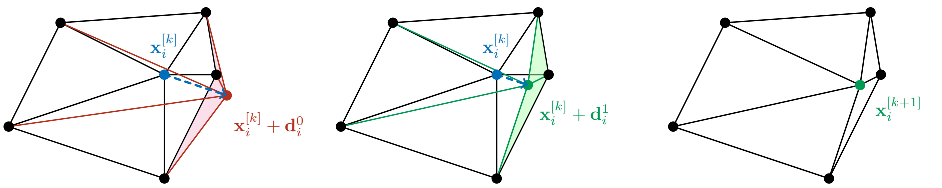
The Laplacian model in the reference domain does not guarantee that the Jacobian of the mapping is strictly positive everywhere, thus leading to the occurrence of tangled (invalid) mesh elements. In two space dimensions, our experience has shown that in most cases carefully tuning the monitor function $\omega(\mathbf{x})$ allows to solve this issue. This is not the case in three dimensions, where tangling occurs much more often.

To cope with this, we have devised an a-posteriori limiter to the nodal displacements which is activated whenever the displacement of a node causes the occurrence of an element whose volume is below a given threshold. This condition is of course implicit, in the sense that it couples the positions of all the mesh nodes. However, it can be easily embedded in an iterative setting. In particular, in our implementation we relax and update each nodal position $\mathbf{x}_i^{[k+1]}$, one after the other. As illustrated on figure 1 (for simplicity in 2D), each displacement $\boldsymbol{\delta}_i^{[k+1]}$ is limited according to the validity of the configuration in the current ball $\mathcal{B}_i^{[k,k+1]}$ obtained using new values $\{\mathbf{x}^{[k+1]}\}_{j<i}$, for nodes updated before i, and old positions $\{\mathbf{x}_j^{[k]}\}_{j>i}$ for nodes not updated yet. This relax-update step has a *Gauss-Seidel* flavour, as the position of each node is updated based on the values of the previously-treated coordinates. In practice, the displacement of node i is iteratively limited by a factor $\mu_i^{s_{\max}}$ as follows

$$\begin{aligned}
&\mathbf{d}_i^0 = \boldsymbol{\delta}_i^{[k+1]} + \boldsymbol{\xi}_i - \mathbf{x}_i^{[k]} \\
&\mathbf{d}_i^{s+1} = \begin{cases} \mu_i \mathbf{d}_i^s & \text{if } \min_{K\in\mathcal{B}_i^{[k,k+1]}} |\Omega_K| < \epsilon \\ \mathbf{d}_i^s & \text{otherwise} \end{cases} \qquad \forall s \in [0,\ldots,s_{\max}-1] \\
&\widetilde{\Delta\mathbf{x}}_i^{[k+1]} = \mathbf{d}_i^{s_{\max}}
\end{aligned} \tag{13}$$

The limiter is thus the result of local sub-iterations, which are stopped when a volume greater than ϵ is guaranteed for every element in the ball. This check allows to enforce the validity of every intermediate mesh configuration, effectively preventing the occurrence of invalid elements at a reasonable computational cost with respect to smoothing or untangling procedures for unstructured meshes [23, 45].

It must be remarked that for interior nodes in general one iteration of the above procedure is enough, while more iterations are required when applying the limiter within the projection step enforcing the boundary conditions (cf. next section). For simplicity here the same value $\mu_i = 0.5$ has been adopted for all the nodes. In this work, this value has proven to be effective in locally preventing tangling while allowing the position of blocked nodes to be naturally relaxed by the successive application of r-adaptation at the next time steps, as the monitor function moves with the flow.



(a) Invalid displacement. (b) Relaxed displacement. (c) Updated configuration.

Figure 1: Two-dimensional illustration of the nodal displacement limiting. 1a: Proposed displacement for node i right after the Jacobi iteration $k+1$, that would produce inverted elements (in red). 1b: Relaxed displacement producing valid elements (in green). 1c: Updated configuration.

3.4. A-posteriori corrections on Dirichlet and slip boundaries

The decoupling of the spatial coordinates obtained by initially accounting for natural boundary conditions only is particularly convenient in terms of computational cost and simplicity of implementation. It allows to store and assembly only a single smaller stiffness matrix to be used for every space coordinate, instead of a matrix of 3×3 blocks. However, the resulting nodal displacements need to be corrected to account for conditions on Dirichlet and slip boundaries. This is achieved by the projection step discussed in this section, which is easily embedded in the scalar iterations. The description is given for slip wall boundaries, of which Dirichlet nodes are a particular case.

In the `Fmg` library boundary geometries are handled by means of curved point-normal triangles [50], i.e. piecewise cubic Bézier patches for the boundary position and quadratic for the boundary normal vector, relying on the implementation provided in `Mmg` [11, 12]. In this setting an implicit surface representation of the slip boundary reading in the continuous case

$$\gamma^S(\mathbf{x}) = 0 \quad \text{on } \Gamma^S_{\boldsymbol{\xi}} \tag{14}$$

is approximated by the explicit piecewise parametric representation

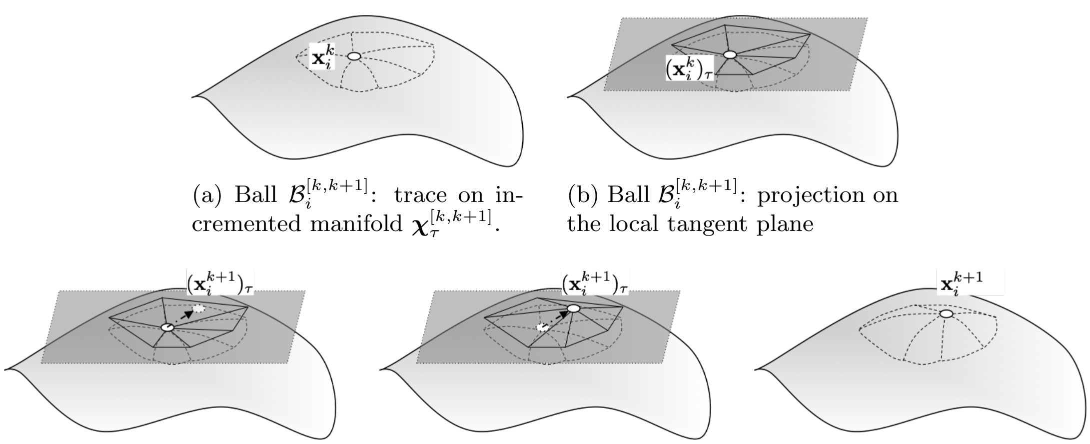
$$\begin{gathered} \boldsymbol{\chi}_\tau[\mathbf{x}_j, \hat{\mathbf{n}}_j] : \Sigma \to \Gamma^S_{\boldsymbol{\xi}}, \quad \Sigma = [0,1] \times [0,1],\ \Gamma^S_{\boldsymbol{\xi}} \subset \mathbb{R}^3 \\ \mathbf{x} = \boldsymbol{\chi}_\tau[\mathbf{x}_j, \hat{\mathbf{n}}_j](\mathbf{w}), \quad \mathbf{w} \in \Sigma \end{gathered} \tag{15}$$

which is defined for each triangle τ in the triangulation of the slip boundary, from the positions and unit normals $\{\mathbf{x}_i, \hat{\mathbf{n}}_i\}_{i \in \tau}$ of the nodes of the triangle. Similarly, the Bézier patches also allow to evaluate surface normals as

$$\begin{gathered} \boldsymbol{\eta}_\tau[\mathbf{x}_j, \hat{\mathbf{n}}_j] : \Sigma \to \Gamma^S_{\boldsymbol{\xi}}, \quad \Sigma = [0,1] \times [0,1],\ \Gamma^S_{\boldsymbol{\xi}} \subset \mathbb{R}^3 \\ \hat{\mathbf{n}} = \boldsymbol{\eta}_\tau[\mathbf{x}_j, \hat{\mathbf{n}}_j](\mathbf{w}), \quad \mathbf{w} \in \Sigma \end{gathered} \tag{16}$$

For some applications, as for example external aerodynamics, handling curved geometries is a necessity. As a consequence, the geometric approximation becomes an integral part of the numerical method. In particular, in three dimensions even the simplest combination of boundary surfaces easily leads to intersection curves. Since sharp edges (ridges) in the initial geometry need to be preserved as well as corners (intersections of multiple ridges), nodes cannot cross a ridge, but they are only allowed to move tangentially to it, and displacement of a corner node cannot happen. As outlined in section 2.1, slip boundary conditions bring a position constraint expressing the belonging of the node to the surface, and a null normal stress condition on the local tangent plane. The latter being already fulfilled by the Neumann conditions naturally imposed on the weak formulation, only the former is of our interest here. Although node belonging to an intersection curve can be formally expressed as the belonging to the two surfaces sharing the curve, this is not practical from an implementation point of view. Slip boundary conditions need thus to be specialized to the chosen geometry approximation and to distinguish among regular curved surfaces, ridges, and corners.

In the following, the boundary treatment is detailed for the supported geometrical features: manifold surfaces, ridges (i.e. intersections of two manifold surfaces) and corners (intersections of two or more ridges). Note that different geometrical representations involving other local or global manifold parametrizations can be easily embedded in the algorithm.



(a) Ball $\mathcal{B}_i^{[k,k+1]}$: trace on incremented manifold $\boldsymbol{\chi}_\tau^{[k,k+1]}$.

(b) Ball $\mathcal{B}_i^{[k,k+1]}$: projection on the local tangent plane

(c) Ball $\mathcal{B}_i^{[k,k+1]}$ and displacement projected on the tangent

(d) Updated Ball $\mathcal{B}_i^{[k,k+1]}$ projected on the tangent plane

(e) Updated Ball $\mathcal{B}_i^{[k,k+1]}$: trace on manifold $\boldsymbol{\chi}_\tau^{[k,k+1]}$

Figure 2: Illustration of the slip boundary projection procedure.

Manifold surfaces. The procedure adopted here to handle slip conditions along manifolds for a node i consists in iteratively projecting the point position on the surface, updating the Bézier patches, and limiting at the same time the displacement to ensure mesh validity. Tangling can tipically occur on surface triangles if too large displacements are allowed, but also the adjacent volume elements can tangle when a point is projected on a concave boundary. For this reason the mesh validity check is always performed on volume elements.

In the first step, we work based on the partially updated ball $\mathcal{B}_i^{[k,k+1]}$, which allows to build a local updated geometrical model. As before, this model is evaluated using the new updates for nodes already processed, and value from the previous iteration for the remaining ones. This provides the incrementally updated geometry model $\boldsymbol{\chi}_\tau^{[k,k+1]} = \boldsymbol{\chi}_\tau[\{\mathbf{x}_j^{[k+1]}, \hat{\mathbf{n}}_j^{[k+1]}\}_{j<i}, \{\mathbf{x}_j^{[k]}, \hat{\mathbf{n}}_j^{[k]}\}_{j\geq i}]$ (cf. (15)). In particular, as shown on figures 2-(a) and 2-(e), this allows to identify the trace of $\mathcal{B}_i^{[k,k+1]}$ on the updated manifold, and its projection on the local tangent plane.

The second step consists of four coupled ingredients:

1. projection of the displacement provided by the Jacobi iteration onto the local tangent plane, leading to an approximate tangent displacement $(\boldsymbol{\delta}_i^{k+1})_\tau$; and preliminary nodal position $(\boldsymbol{x}_i^{k+1})_\tau$, as shown on figure 2-(c);
2. identification of the element containing the new node position, based on baricentric coordinates interpolation, as shown on figures 2-(c) and 2-(d);
3. Bézier interpolation $\boldsymbol{\chi}_\tau^{[k,k+1]}(\mathbf{w})$ on the geometrical model, as shown on figure 2-(e);
4. limiting of the displacement based on the minimum element volume, as discussed in section 3.3.

The iterations providing the final displacement, and hence position, are similar to (13):

$$
\begin{aligned}
&\mathbf{d}_i^0 = \boldsymbol{\chi}_\tau^{[k,k+1]}\left(\mathbf{w}(\mathbf{x}_i^{[k+1]})_\tau\right) - \mathbf{x}_i^{[k]} \\
&\mathbf{d}_i^{s+1} = \begin{cases} \boldsymbol{\chi}_\tau^{[k,k+1]}\left(\mathbf{w}(\mathbf{x}_i^{[k]} + \mu_i \mathbf{d}_i^s)\right) - \mathbf{x}_i^{[k]} & \text{if } \min_{K\in\mathcal{B}_i^{[k,k+1]}} |\Omega_K| < \epsilon \\ \mathbf{d}_i^s & \text{otherwise} \end{cases}, \\
&\qquad \forall s \in [0, \ldots, s_{\max} - 1] \\
&\widetilde{\Delta\mathbf{x}}_i^{[k+1]} = \mathbf{d}_i^{s_{\max}}
\end{aligned}
\tag{17}
$$

We stress again that since the piecewise patches depend on both node positions and unit normals, the position update is always accompanied by the re-evaluation of the unit normal vectors through the analogously defined model $\boldsymbol{\eta}_\tau^{[k,k+1]}$ (cf. (16)). This is omitted from (17) to keep a lighter notation.

Ridges. The displacement check and projection on boundary ridges is handled exactly in the same way as for manifold surfaces. The main difference is that now the parametric space is replaced with a curve parametrisation which is one-dimensional $\Sigma \subset \mathbb{R}$. Thus all operations previously performed on the tangent plane are performed by projection on the tangent line, and normal vectors of both the manifold surfaces joining at the ridge are stored and updated in the geometrical model.

Corners. These are the only allowed Dirichlet nodes, thus corners verify exactly the boundary condition, and are not included in the discrete variational form 3.1. In this specific case, displacement is not allowed as they are already on the exact geometry, and the condition imposed is

$$\mathbf{x}_i^{[k+1]} = \boldsymbol{\xi}_i \tag{18}$$

3.5. Unsteady mesh adaptation through restarted iterations

Following [39, 10], dynamic mesh adaptation during the time evolution of a fluid flow simulation is performed by repeating the steady adaptation procedure described in the previous section at each time step, without the explicit formulation of a differential equation in time for mesh motion. This simplifies the coupling with existing flow solvers.

In this case of fixed boundary domains, the reference mesh $\boldsymbol{\xi}$ is constant in time, while the computational mesh $\mathbf{x}(t^{(n+1)})$ is the r–adaptation of the (fixed) reference mesh. Thus, the displacement at each time step $n+1$ is initialized with the value achieved at the last Jacobi iteration K achieved in the previous time step n

$$\boldsymbol{\delta}_i^{[0](n+1)} = \boldsymbol{\delta}_i^{[K](n)} \tag{19}$$

so that successive Jacobi iterations during time evolution are effectively accumulated on the nodes positions.

4. Validation via adaptation on analytical functions

We consider here a series of analytical tests allowing to measure the effectiveness of the method. As shown in section 2, we recall here that the mesh adaptation model can be governed by the number of iterations n_{it} plus the three parameter pairs (α, γ_α), (β, γ_β), (τ, γ_τ), representing the intensity and the normalization constant of the solution gradient, the solution Hessian, and the solution itself in the definition of the monitor function. In this work, we have not seen specific benefits in mixing all three parameter pairs, so we will explicitly report only the values for the used pairs, while values not shown are assumed to be zero. As elucidated in [10], since the reference domain Laplacian model in multiple dimensions is not derived from an error equidistribution principle, its numerical solution until convergence is not strictly required to reach satisfactory mesh adaptation and, in practice, a number of Jacobi iterations in the order of $\mathcal{O}(10)$ are generally sufficient to reach the desired adaptation. The number of iterations n_{it} will be reported for each case.

In section 4.1 adaptation is performed on a a steady Gaussian-like function, in order to test the convergence order on the interpolation error. In section 4.2 adaptation is performed on an unsteady analytical moving front passing over a sphere, in order to assess the capability of the model to preserve the validity of the mesh over intersecting curved boundaries throughout the time simulation.

4.1. Steady adaptation in a square and a cube

We consider the approximation of the function

$$\rho = e^{\theta\psi^2}, \qquad \psi = \|\mathbf{x}\|^2 - R^2 \tag{20}$$

with $\theta = 40$, $R = 0.75$. We consider both a two and three dimensional variant of the problem, the first defined on a square domain $[-2, 2] \times [-2, 2]$, the second on the cube $[-2, 2] \times [-2, 2] \times [-2, 2]$. This solution is plotted in figures 4a and 4b. In both cases we consider a series of simplicial meshes with a uniform mesh size distribution, and different average edge size h, whose details are shown in tables 1 and 2. The above function is chosen in order to test capability of the models to adapt on a circle represented by a smooth solution field, before their application to solutions with sharp/discontinuous features. The mesh PDE parameters are set to $(\tau, \gamma_\tau) = (5000, 1.0)$ in 2D, and to $(\alpha, \gamma_\alpha) = (500, 0.1)$ in 3D. Also note that the a-posteriori limiter for the displacement is only applied in 3D, which is the case in which tangling is more often occurring.

On these meshes, we measure the L^2-error convergence of the $\mathbb{P}^1$ interpolation $\Pi\rho$

$$||e||_{L^2} = \left(\int_\Omega |\rho - \Pi\rho|^2 \, \mathrm{d}\Omega \right)^{\frac{1}{2}} \tag{21}$$

We plot the observed trends in figures 3a and 3b. It can be seen that in two dimensions it is easier to preserve, quite independently from the number of iterations n_{it} performed, the second order convergence rate of the $\mathbb{P}^1$ interpolation, with an error reduction for a given number of nodes shown in table 3, but a high number of iterations on a coarse mesh can actually increase the error.

In three dimensions, while the error on the adapted meshes is considerably lower (table 4), the number of Jacobi iterations has to be increased to preserve the second order rate. Some adapted meshes obtained from the $h = 0.1$ and $h = 0.05$ initial meshes are visualized in figures 4 to help understand these two phenomena. Taking as example the three-dimensional case, as the initial mesh is refined from $h = 0.1$ to $h = 0.05$ in figure 4, it can be appreciated that the displacement produced by the same number of iterations and the same adaptation parameters is smaller. This has two consequences. The first consequence is that a high number of iterations on coarse meshes can excessively stretch the mesh elements (as shown in figure 4g) in an orthogonal pattern, due to the uncoupling of the Laplacian model in the coordinate directions, possibly increasing the approximation error on the adapted mesh (as seen in table 3 for the 2D case for the coarsest meshes). The second consequence is that more iterations are needed on fine meshes to preserve the second order rate, as shown in figure 3b. In three dimensions, the a-posteriori limiter also contributes to this effect by constraining the allowed displacement of each node inside its ball at each iteration.

These effects can be appreciated by observing the trend for the tetrahedron quality

$$Q = \frac{\left(\sum_{j=1}^{6} l_j^2 \right)^{3/2}}{\alpha |\Omega_K|} \tag{22}$$

where l_j is the length of each edge of the element, $|\Omega_K|$ its volume, and α the normalization factor to get $Q = 1$ on a regular tetrahedron with unit edges. Since r-adaptation inevitably introduces some anisotropy which is not taken into account in our quality measure, we expect the quality to be somewhat degraded in the adapted regions. Anyway, a too high percentage of bad quality elements, when sharp solution fronts are quite

h	0.0125	0.025	0.05	0.075	0.1	0.15
Nb. of nodes	135550	34310	8560	3993	2213	1015
Nb. of elements	271098	68618	17118	7984	4424	2028

Table 1: Mesh data for the 2D square convergence analysis.

h	0.0375	0.05	0.075	0.1	0.15
Nb. of nodes	319830	140264	44521	20604	6727
Nb. of elements	1844811	802080	237458	106130	32308

Table 2: Mesh data for the 3D cube convergence analysis.

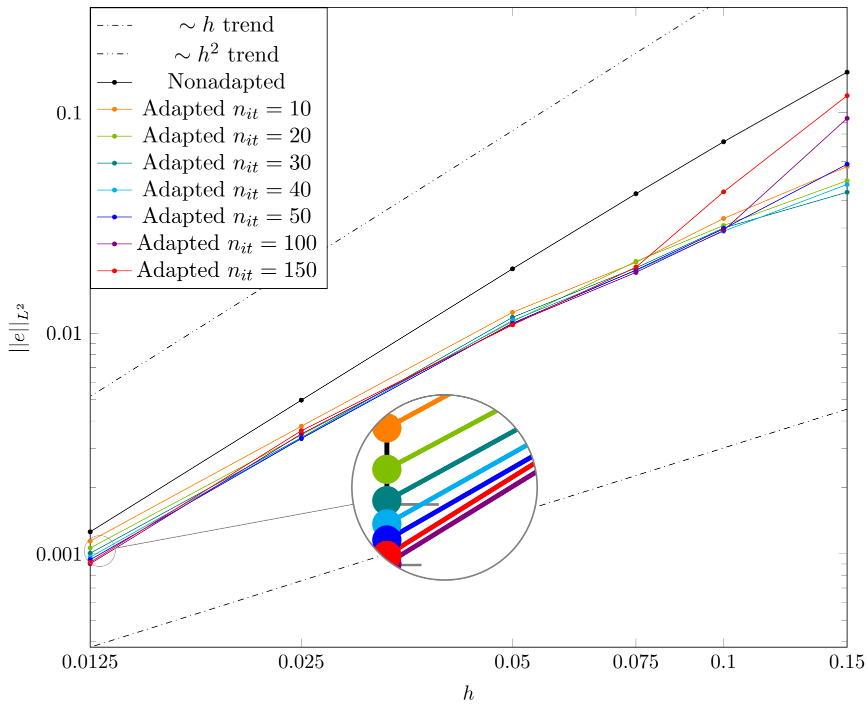
h	$\mathcal{E}^{[0]}$	$\mathcal{E}^{[10]}$	$r^{[10]}$	$\mathcal{E}^{[150]}$	$r^{[150]}$
0.15	1.525974e-01	5.693340e-02	62.6905 %	1.194055e-01	21.751 %
0.1	7.379553e-02	3.313339e-02	55.1011 %	4.372964e-02	40.742 %
0.075	4.288518e-02	2.097308e-02	51.0948 %	1.991813e-02	53.555 %
0.05	1.958636e-02	1.243068e-02	36.5340 %	1.090836e-02	44.306 %
0.025	4.974168e-03	3.788809e-03	23.8303 %	3.614722e-03	27.330 %
0.0125	1.258864e-03	1.142442e-03	9.2482 %	9.152757e-04	27.294 %

Table 3: Interpolation errors $\mathcal{E}^{[k]} = ||e^{[k]}||_{L^2}$ for the 2D square convergence analysis, for 10 and 150 iterations, and reduction $r^{[k]} = (1 - \mathcal{E}^{[k]}/\mathcal{E}^{[0]})$ with respect to the nonadapted case.

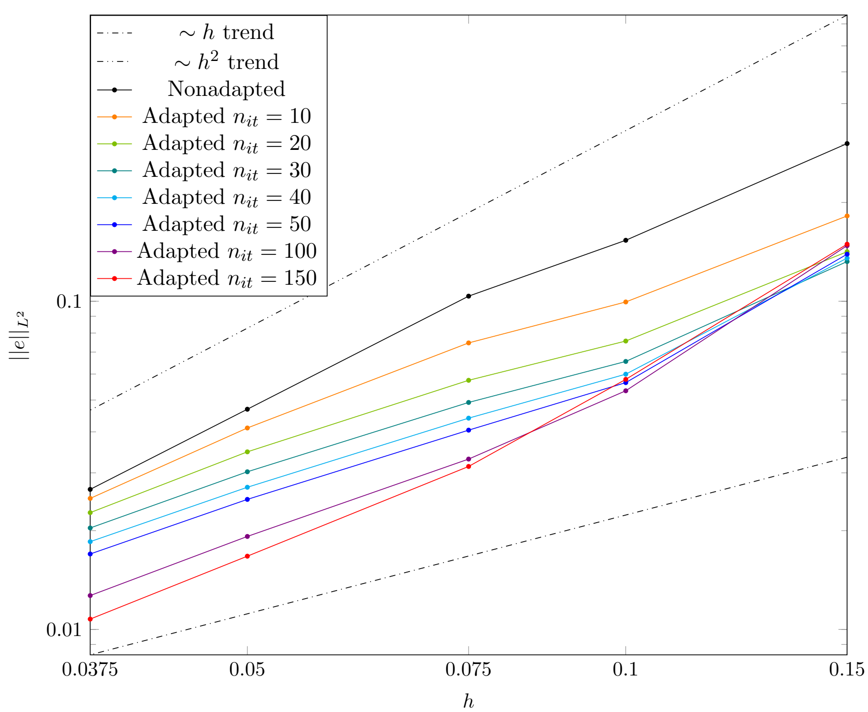
h	$\mathcal{E}^{[0]}$	$\mathcal{E}^{[10]}$	$r^{[10]}$	$\mathcal{E}^{[150]}$	$r^{[150]}$
0.15	3.023667e-01	1.693936e-01	43.9774 %	1.450969e-01	52.013 %
0.1	1.533983e-01	9.494413e-02	38.1061 %	5.919961e-02	61.408 %
0.075	1.036390e-01	7.284977e-02	29.7082 %	2.832881e-02	72.666 %
0.05	4.687948e-02	4.084946e-02	12.8628 %	1.424675e-02	69.610 %
0.0375	2.671484e-02	2.499870e-02	6.4239 %	9.579343e-03	64.142 %

Table 4: Interpolation errors $\mathcal{E}^{[k]} = ||e^{[k]}||_{L^2}$ for the 3D cube convergence analysis, for 10 and 150 iterations, and reduction $r^{[k]} = (1 - \mathcal{E}^{[k]}/\mathcal{E}^{[0]})$ with respect to the nonadapted case.

localized in the domain, can be a sign that the mesh is stretched also in smooth solution regions, possibly worsening the error reduction performances. In figure 5 we plot the evolution of the histograms of the elements quality with the number of iterations for the $h = 0.1$ and $h = 0.05$ meshes. The excessive stretch observed in figure 4g corresponds to a significantly degradation of the elements quality for the $h = 0.1$ mesh, expecially when increasing the number of iterations, with more than 24% of elements having $Q < 0.2$ for 150 iterations, much higher than for the $h = 0.05$ (less than 10%).

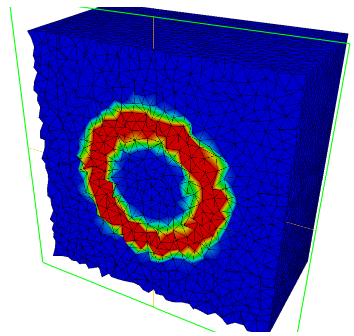


(a) Interpolation error trend for the 2D square test case.

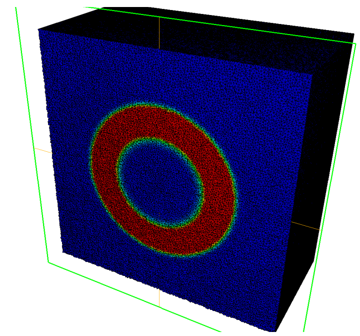


(b) Interpolation error trend for the 3D cube test case.

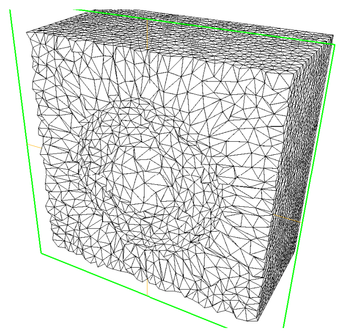
Figure 3: Interpolation error convergence with mesh adaptation for the square and cube analytical test cases.



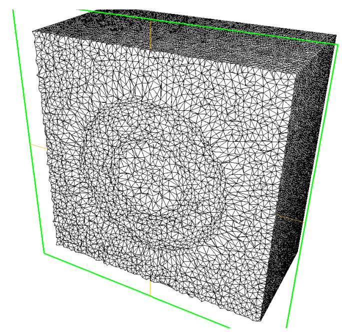
(a) Monitor function, $h = 0.1$ mesh.



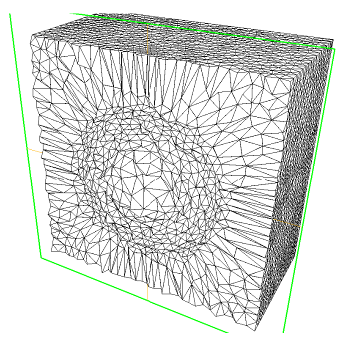
(b) Monitor function, $h = 0.02$ mesh.



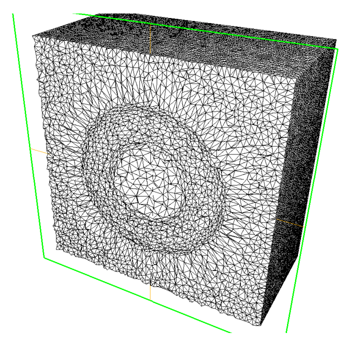
(c) Adapted mesh $h = 0.1, n_{it} = 10$.



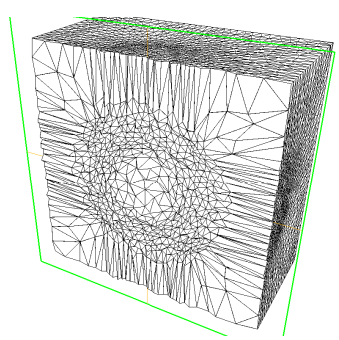
(d) Adapted mesh $h = 0.05, n_{it} = 10$.



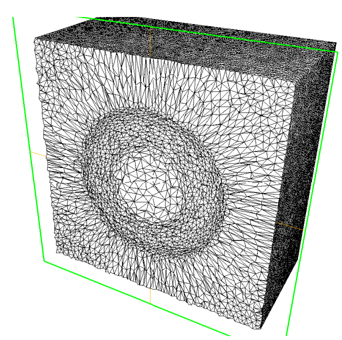
(e) Adapted mesh $h = 0.1, n_{it} = 30$.



(f) Adapted mesh $h = 0.05, n_{it} = 30$.



(g) Adapted mesh $h = 0.1, n_{it} = 100$.



(h) Adapted mesh $h = 0.05, n_{it} = 100$.

Figure 4: Monitor function (top row) and volumic cuts in the adapted meshes (second to last row) for the cube test case, for different number of iterations, on the $h = 0.1$ and $h = 0.05$ initial meshes.

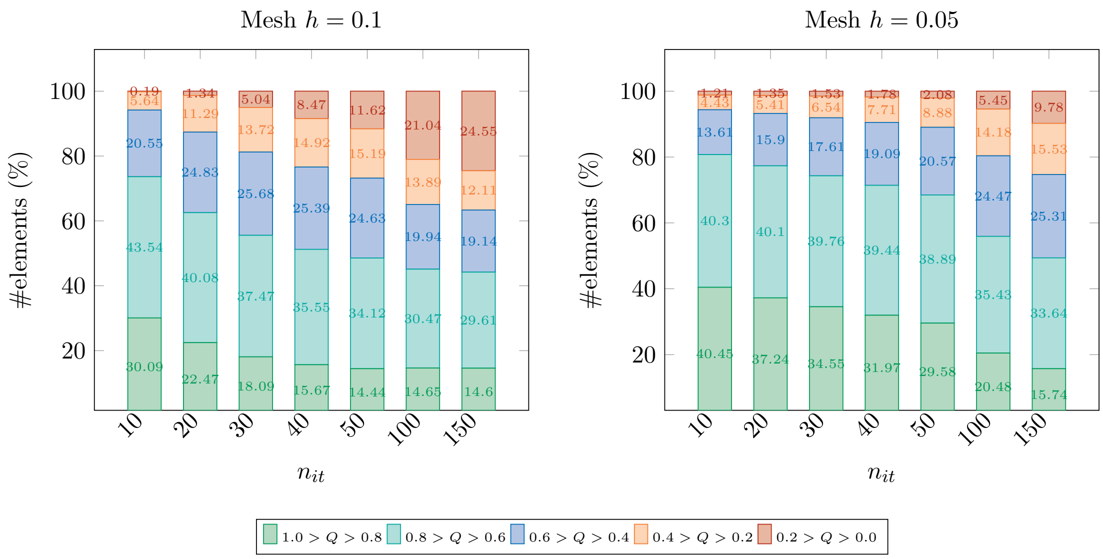


Figure 5: Evolution of mesh elements quality Q with the number of iterations n_{it} for the 3D cube test, for the $h = 0.1$ and $h = 0.05$ meshes.

4.2. Moving front passing over a spherical boundary

The algorithm was tested by adapting over a moving front defined as

$$\rho\left(X(x,t)\right) = \begin{cases} 1 & \text{if } X(x,t) < 0 \\ 0.5\cos\left(s\pi X(x,t) + 1\right) & \text{if } X(x,t) \in [0,\delta] \\ 0 & \text{if } X(x,t) > \delta \end{cases} \tag{23}$$

with

$$X(x,t) = s(x - x_0 + vt) \tag{24}$$

and scaling $s = 20$, initial position $x_0 = 0.7$, speed $v = 0.2$, front thickness $\delta = 0.005$. Unsteady mesh adaptation is performed on this analytical solution every $\Delta t = 0.25$.

The setup is shown in figures 6a and 6b. The domain is a quarter cylinder of radius 1.5 along the x-axis with $x \in [-1.5, 1.5]$, surrounding a quarter sphere centered at the origin with radius 0.5. This case is designed to test as many geometrical sources of mesh tangling as possible before the application to fluid flow simulations, as it contains at the same time curved surfaces, ridges (the intersection of the sphere with each symmetry planes) and corners (the intersections of the sphere with both the symmetry planes), and a sharp solution moving over the geometry. Adaptation is performed with $(\alpha, \gamma_\alpha) = (40, 0.1)$, with 30 Jacobi iterations, on an uniform mesh with edge size $h = 0.05$. The number of nodes and elements is reported in table 5, as this is the same base mesh that will be used for the shock-sphere interaction simulations in the next section.

The obtained meshes are shown in figure 6 showing in particularly that the method is able to preserve a valid mesh both when the front is passing over the surface of the sphere (figures 6e, 6f) and most importantly when it hits and leaves the sphere (figures 6c, 6d and 6g, 6h respectively). Without the a-posteriori limiter, that effectively blocks excessive deformation near the corners and in the first layer of elements above the curved surface, it was impossible to complete the simulation without the occurrence of tangled elements.

Remarks on mesh folding and the purpose of the a-posteriori limiter. As discussed in section 1, there is no analytical proof for the validity of the meshes produced by our model neither in the continuum nor in the discrete setting. Examples of folded meshes have indeed already been reported in the literature for several other methods [15, 30]. Mesh folding has not been reported for the variable-diffusion Laplacian in the reference domain in two dimensions [9, 38, 10], but in [9] the authors themselves remark that there is no theoretical reason against its occurrence. In three dimensions, we have found that it is quite frequent to produce folded elements for too strong adaptation parameters or on concave boundaries when the limiter presented in the previous section is not applied. An example of the first situation is given in figure 7a, where an inverted element is produced just outside of the most refined region. An example of tangling on a concave boundary is given in figure 7b, where two points on the surface are blocked and cannot move without folding the adjacent elements (the volume limiter is not applied, but displacement on the surface is limited on the surface ball in order to allow the projection on Bézier patches), and one element near the lower circle is folded.

In the numerical simulations presented in the next section, all of which have concave boundaries, tangling was observed whenever a shock wave hit or developed on the front of the object, without limiter. Since this happened in the first instants of the simulations, we have found that the straightforward three-dimensional extension of the original variable-diffusion Laplacian method in the reference domain [9] would simply be unpractical on those cases without an additional limiting or correction step to avoid mesh folding.

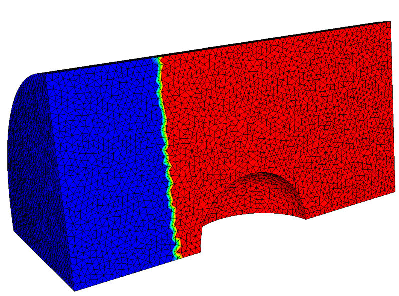
5. Adaptation for unsteady compressible flows

We consider the simulation of unsteady inviscid compressible flows in a time dependent frame of reference. In particular, we couple the `Fmg` library we developed to the `Flowmesh` solver [21, 28, 36], based on a node-centered second order, total variation-diminishing finite volume scheme for the Euler equations, written in an Arbitrary-Lagrangian-Eulerian (ALE) form [13]

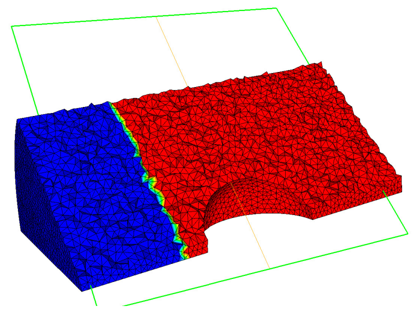
$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathbf{u}\,\mathrm{d}\Omega + \oint_{\partial\Omega(t)} \hat{\mathbf{n}}\cdot(\mathbb{F}(\mathbf{u}) - \mathbf{v}\mathbf{u})\,\mathrm{d}\Gamma = \mathbf{0} \tag{25}$$

where $\mathbf{u}$ is the array of the conservative solution, $\mathbb{F}(\mathbf{u})$ its flux, ρ denotes the mass density, $\rho\mathbf{U}$ the momentum, and ρe^t the total energy density. The moving domain velocity is represented by the vector field $\mathbf{v}$

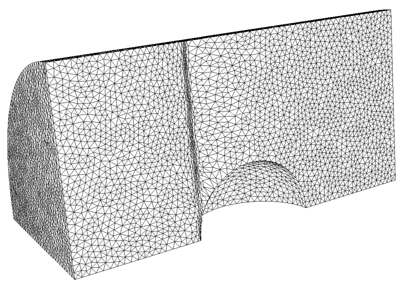
$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho\mathbf{U} \\ \rho e^t \end{pmatrix}^T, \qquad \mathbb{F}(\mathbf{u}) = \begin{pmatrix} \rho\mathbf{U} \\ \rho\mathbf{U}\otimes\mathbf{U} + P\mathbb{I} \\ \rho e^t\mathbf{U} + P\mathbf{U} \end{pmatrix}^T \tag{26}$$



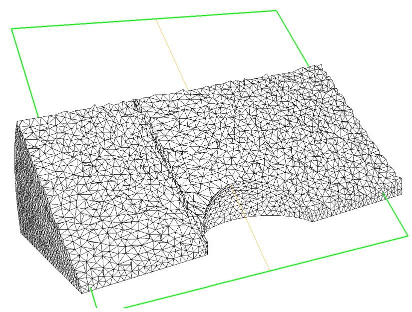
(a) Solution on input mesh boundary.



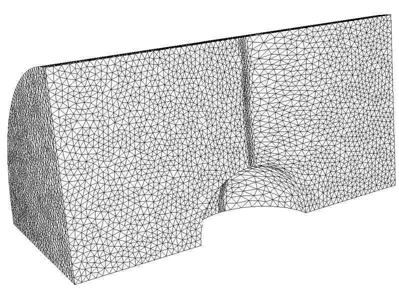
(b) Solution on input mesh volumic cut.



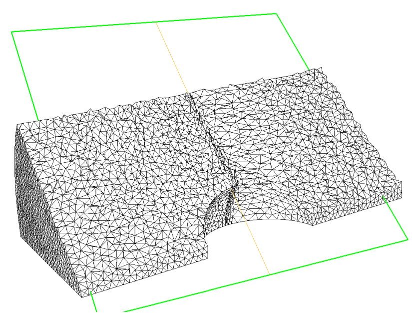
(c) Output mesh boundary at $t = 1.0$.



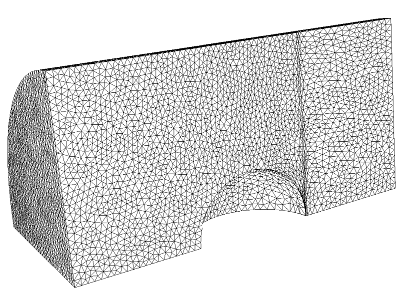
(d) Output mesh volumic cut at $t = 1.0$.



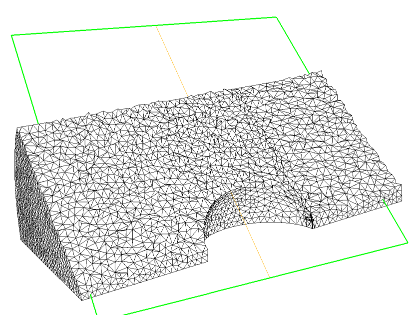
(e) Output mesh boundary at $t = 3.5$.



(f) Output mesh volumic cut at $t = 3.5$.

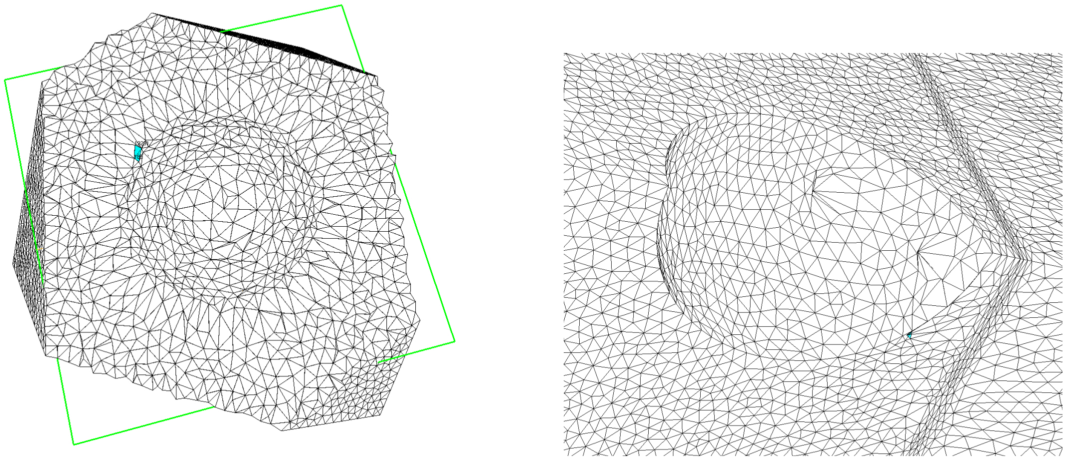


(g) Output mesh boundary at $t = 6.0$.



(h) Output mesh volumic cut at $t = 6.0$.

Figure 6: Moving front test case, meshes from $t = 0.0$ to $t = 6.0$.



(a) Cube $h = 0.1$ fifth iteration without limiteri with $(\tau, \gamma_\tau) = (10000, 0.1)$. One inverted element (cyan).

(b) Moving front test case without limiter at $t = 6.5$. Two blocked points (distorted balls on the surface of the sphere) and one inverted element near the lower circle (cyan).

Figure 7: Examples of folded meshes with no limiter applied.

The pressure P is computed using the ideal gas equation of state for ideal gases

$$P = (\gamma - 1)\left(\rho e^t - \frac{1}{2}\rho|\mathbf{U}|^2\right)$$

Within the code, a local conservative solution transfer procedure at each time step is guaranteed by the ALE formulation.

Unsteady mesh adaptation is performed according to the scheme shown in section 3.5. At each time step, the flow solution is predicted on the previous computational mesh, then the computational mesh is adapted, and finally the flow solution is recomputed on the adapted mesh. To this end, `Flowmesh` makes use of a conservative ALE-remap exactly matching the volumes swept by cell faces during mesh displacements and nodal volumes, and automatically fulfilling a Discrete Geometric Conservation Law (DGCL) [22, 18, 53]. The code also includes the support of topological mesh modifications like edge split, edge collapse, barycentric node insertion, and Delauney node insertion, not used in this work.

To apply mesh adaptation at each time step, a low order computation of the solution at the next time step on the current mesh is used to provide a monitor function to the mesh PDEs.

5.1. Case 1: two-dimensional forward facing step

As a preliminary validation, we reproduce the results shown for the same method without a posteriori relaxation in [10] for the two-dimensional forward facing step [17, 48, 52]. Our initial mesh is a Delauney triangulation made of 10946 elements, 5474 nodes, with an average edge length $h = 0.0025$. Note that this unstructured mesh has

	Base mesh		Refined mesh	
	# nodes	# elements	# nodes	# elements
Step 2D	5474	10946	21639	43276
Step 3D	47445	277655	555026	3217351
Shock-sphere	35379	209142	488963	2872845

Table 5: Number of nodes and elements for the simplicial meshes employed for the unsteady compressible flow cases.

	Base (nonadapted)	Base (adapted)	Refined (nonadapted)
Step 2D	31m 32s	44m 43s	2h 52m 21s
Step 3D	1h 39m 55s	2h 21m 28s	45h 37m 18s
Shock-sphere	40m 12s	1h 32m 48s	12h 5m 12s

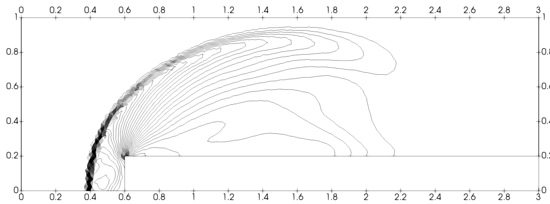
Table 6: Computational times comparison. The overhead due to solution prediction and adaptation is important, but negligible if compared with an uniform refinement strategy.

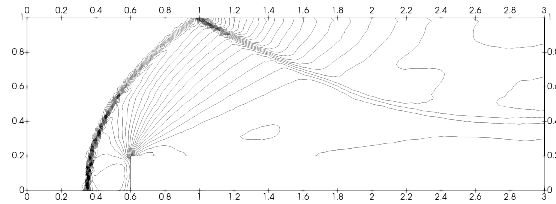
a higher edge size with respect to the one proposed in [52], which had an edge size $h = 0.00125$. The initial condition is a uniform Mach 3 flow towards the right of the domain.

All simulations are run on 4 cores of a Intel Xeon E5-2690 (2.6 GHz), mesh adaptation is serial. We perform mesh adaptation on the base $h = 0.0025$ mesh, and compare results with those obtained without adaptation on the refined $h = 0.00125$ mesh. Adaptation is performed on mass density, with $(\alpha, \gamma_\alpha) = (40, 0.1)$ and $(\beta, \gamma_\beta) = (10, 0.5)$. Mesh data are shown in table 5, while contour lines for mass density are compared in figures 8 and 10. Contour lines range and spacing for each time instant is the same as in [52]. The adapted meshes are shown in figures 9 and 11. Shock waves are resolved better on the coarse adapted mesh than on the refined nonadapted mesh, while resolution on rarefaction fans and contact discontinuities is comparable. Computational times are shown in table 6. While mesh adaptation produces a significant overhead if compared to the base nonadapted case, this overhead is negligible if compared to the refined nonadapted calculation.

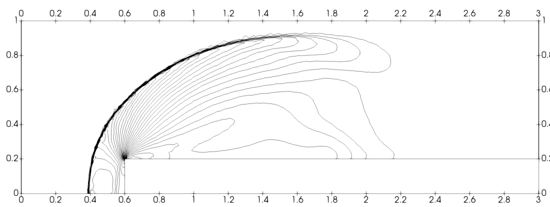
5.2. Case 2: three-dimensional forward facing step

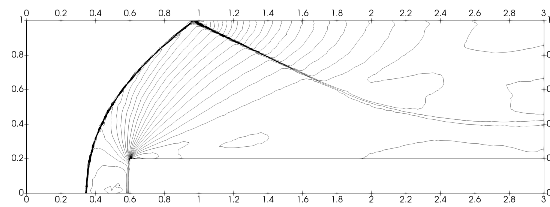
We propose a three-dimensional extension of the classical supersonic forward facing step. The impulsive start of a Mach 3 flow in a 3 length units long and 1 length unit wide/high wind tunnel, with a 0.2 length unit wide/high step located at 0.6 length units from the inlet (see figure 12a). Adaptation is performed on the mass density (figures **??** and **??**), with $(\alpha, \gamma_\alpha) = (40, 0.02)$ on a base mesh with an overall edge size $h = 0.04$ (slightly refined on the step front plane, $h = 0.02$). Results are compared with those obtained without adaptation on a refined mesh with uniform edge size $h = 0.015$. The number of elements and nodes in the meshes are shown in table 5. Contour lines for mass density on the same diagonal cut plane are shown in figures 13 and 15, for 50 equispaced lines between the values 0.715867 and 6.03154. To obtain a comparable resolution on shocks between the coarse adapted and the refined nonadapted meshes,



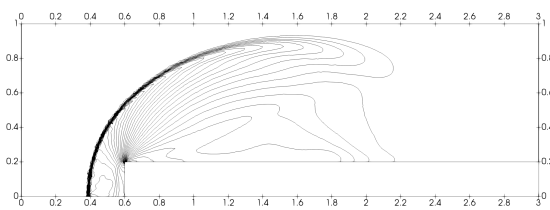


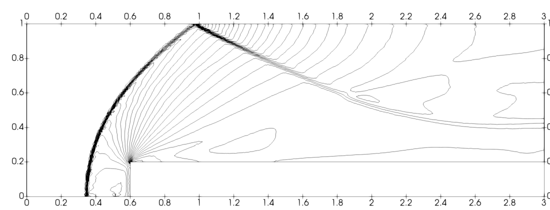
(a) Nonadapted $h = 0.0025$ mesh at $t = 0.5$. (b) Nonadapted $h = 0.0025$ mesh at $t = 1.0$.





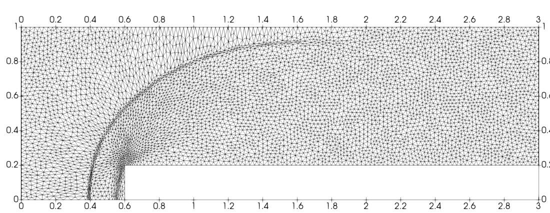
(c) Adapted $h = 0.0025$ mesh at $t = 0.5$. (d) Adapted $h = 0.0025$ mesh at $t = 1.0$.

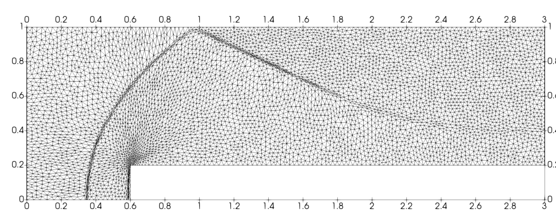




(e) Fine (nonadapted) $h = 0.00125$ mesh at $t = 0.5$. (f) Fine (nondapted) $h = 0.00125$ mesh at $t = 1.0$.

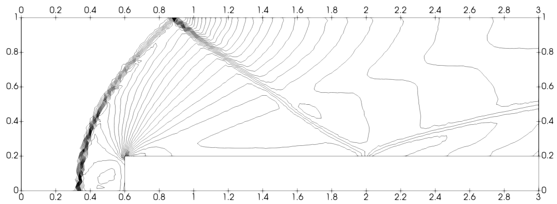
Figure 8: Two-dimensional forward facing step mass density contour lines at $t = 0.5$ and $t = 1.0$.



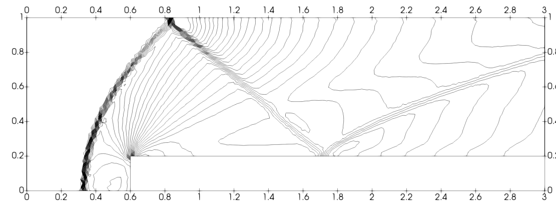


(a) Adapted $h = 0.0025$ mesh at $t = 0.5$. (b) Adapted $h = 0.0025$ mesh at $t = 1.0$.

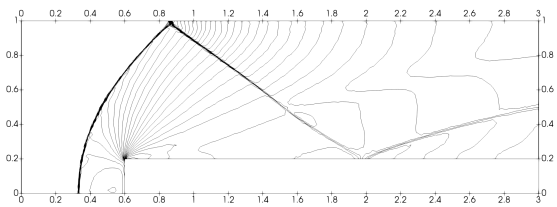
Figure 9: Two-dimensional forward facing step adapted meshes at $t = 0.5$ and $t = 1.0$.



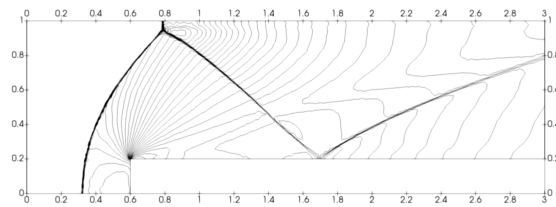
(a) Nonadapted $h = 0.0025$ mesh at $t = 1.5$.



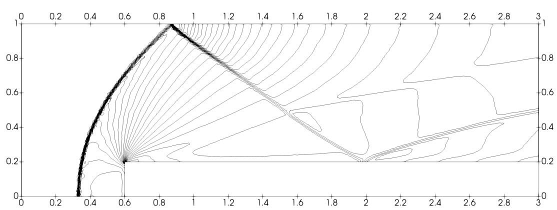
(b) Nonadapted $h = 0.0025$ mesh at $t = 2.0$.



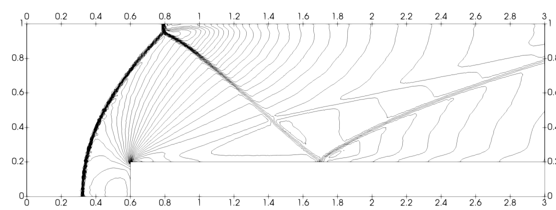
(c) Adapted $h = 0.0025$ mesh at $t = 1.5$.



(d) Adapted $h = 0.0025$ mesh at $t = 2.0$.

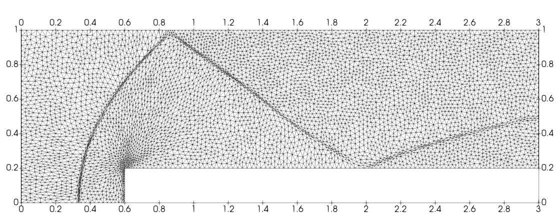


(e) Fine (nonadapted) $h = 0.00125$ mesh at $t = 1.5$.

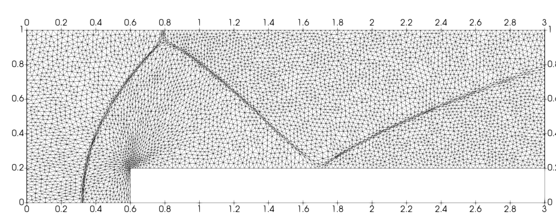


(f) Fine (nonadapted) $h = 0.00125$ mesh at $t = 2.0$.

Figure 10: Two-dimensional forward facing step mass density contour lines at $t = 1.5$ and $t = 2.0$.

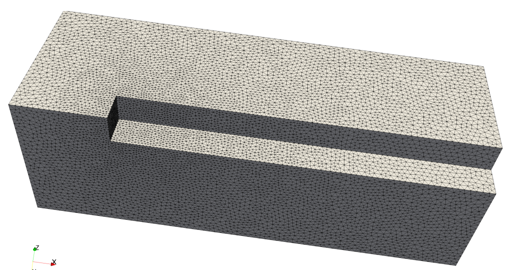


(a) Adapted $h = 0.0025$ mesh at $t = 1.5$.

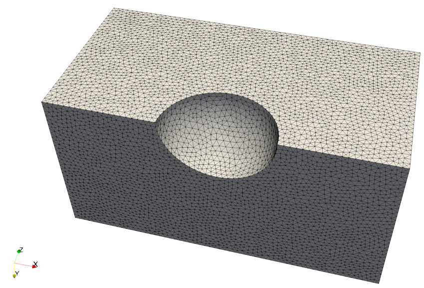


(b) Adapted $h = 0.0025$ mesh at $t = 2.0$.

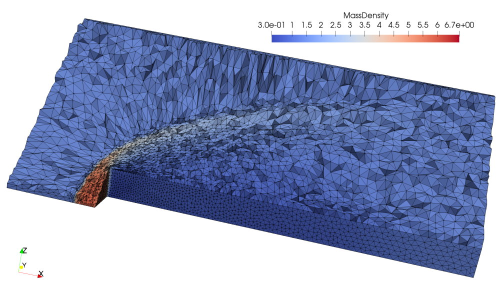
Figure 11: Two-dimensional forward facing step adapted meshes at $t = 1.5$ and $t = 2.0$.



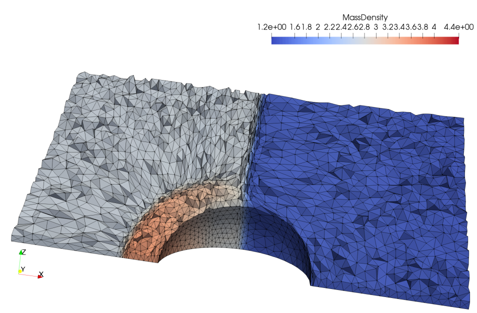
(a) Forward facing step test case.



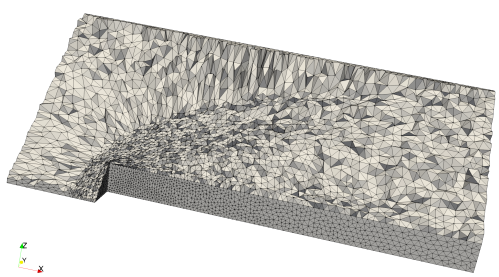
(b) Shock-sphere test case.



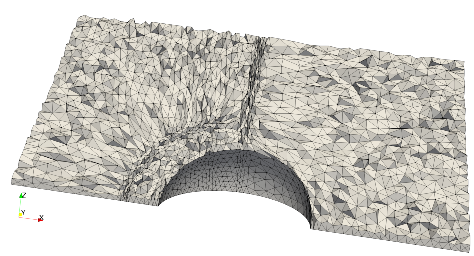
(c) Step, volumic cut and mass density at $t = 0.7$.



(d) Sphere, volumic cut and mass density at $t = 90$.



(e) Step, volumic cut at $t = 70$.



(f) Sphere, volumic cut at $t = 90$.

Figure 12: Initial meshes, adapted meshes and solution for the three-dimensional forward facing step and shock-sphere interaction cases.

we had to produce a refined mesh that is more than ten times bigger (in terms of nodes and elements) than the coarse one. Note that the diagonal cut is possibly the most demanding plane on which results can be compared, as the Laplacian model is uncoupled in multiple space directions, thus it tends to provide better results on cartesian planes, as shown in section 4.1. Adapted meshes are shown in figures 14 and 16. Computational times are shown in table 6. The benefits in terms of computational times in three dimensions are greater than in two dimensions. Anyway, while in two dimension we observed that mesh tangling was a rare occurrence with our Laplacian model, in three dimensions it was impossible to continue the time simulation without the a-posteriori limiter after the first few time steps, due to the strong deformation that quickly led to tangled elements at the step front and around its corners, but also at the shock reflection lines.

5.3. Case 3: shock–sphere interaction

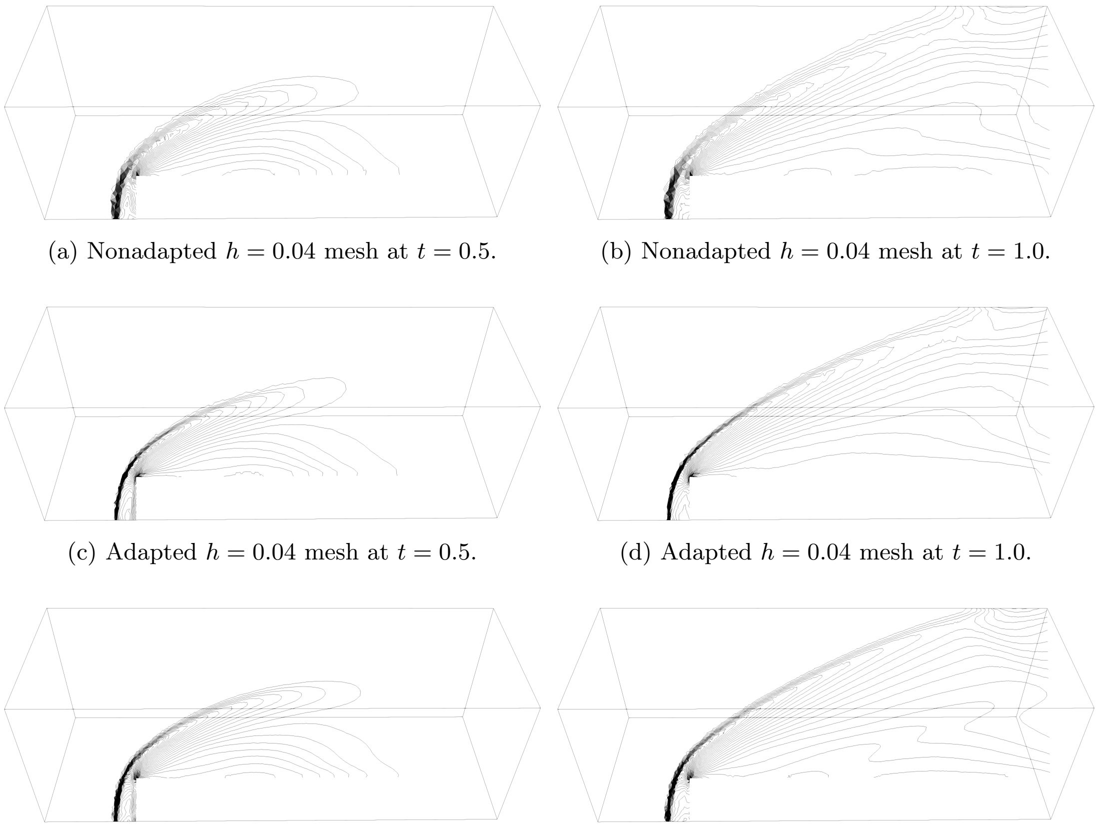
In order to test the capabilities of the method to handle simultaneously shock waves and curved boundary, we choose to simulate the interaction of a traveling shock wave on a sphere. Some configurations for the diffraction of shock waves over cylindrical and spherical obstacles have been studied experimentally for example in [7, 40]. An early application of unstructured mesh adaptation to two-dimensional shock-cylinder simulations can be found in [14], while structured grid adaptation on axisymmetric shock-sphere simulations can be found in [37].

The simulation is limited to a quarter of a cylindrical domain (as for the analytically moving shock of the previous section, see figure 12b). We choose a planar shock moving at $M_s = 1.5$. Adaptation is performed on the mass density (figures **??** and **??**), with $(\alpha, \gamma_\alpha) = (40, 0.1)$. Again, the aim is to compare the results obtained with mesh adaptation on a base mesh with edge size $h = 0.05$ with those obtained on a uniformly refined mesh with edge size $h = 0.02$. Mesh data are shown in table 5. Contour lines for the mass density solution on a radial plane are shown in figures 17 and 19, for 50 equispaced lines between the values 1.36081 and 4.00883. Resolution on shock waves with mesh adaptation is comparable with those obtained on a uniform mesh about ten times bigger in terms on number of nodes and elements. Adapted meshes are shown in figures 18 and 20. Computational times are shown in table 6.

In this case too it was impossible to complete the simulation over valid meshes without the action of the a-posteriori limiter near the corners and the curved surface.

6. Conclusions

The proposed algorithm for dynamic r-adaptation extends to three dimensions the method first proposed in [9, 10, 5] for two-dimensional flows. An iterative solver based on diagonal Jacobi iterations for the discretized mesh PDEs with natural boundary conditions allows a cheap, uncoupled solution in each space direction. A novel a-posteriori relaxation scheme allows to prevent mesh tangling through the construction of a sequence of valid meshes also over curved boundary surfaces and corners, which is the main concern of r-adaptation methods in multiple dimensions, and it is interleaved with

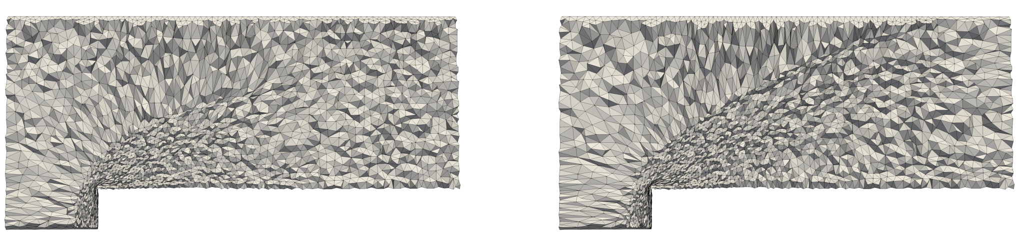


(a) Nonadapted $h = 0.04$ mesh at $t = 0.5$. (b) Nonadapted $h = 0.04$ mesh at $t = 1.0$.

(c) Adapted $h = 0.04$ mesh at $t = 0.5$. (d) Adapted $h = 0.04$ mesh at $t = 1.0$.

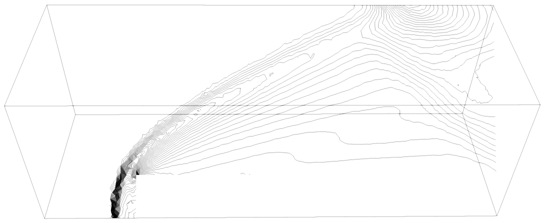
(e) Fine (nonadapted) $h = 0.015$ mesh at $t = 0.5$. (f) Fine (nonadapted) $h = 0.015$ mesh at $t = 1.0$.

Figure 13: Three-dimensional forward facing step mass density contour lines at $t = 0.5$ and $t = 1.0$.

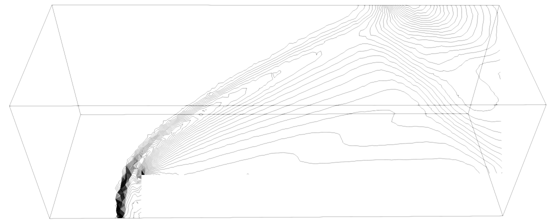


(a) Adapted $h = 0.04$ mesh at $t = 0.5$. (b) Adapted $h = 0.04$ mesh at $t = 1.0$.

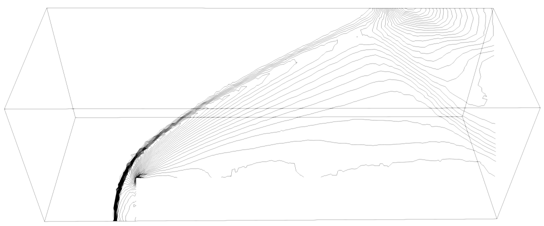
Figure 14: Three-dimensional forward facing step adapted meshes at $t = 0.5$ and $t = 1.0$.



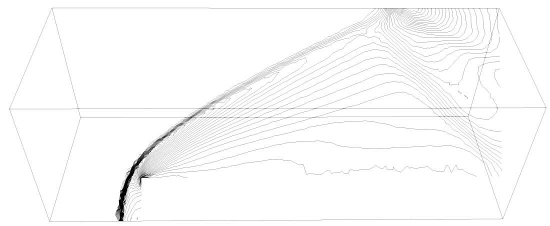
(a) Nonadapted $h = 0.04$ mesh at $t = 1.5$.



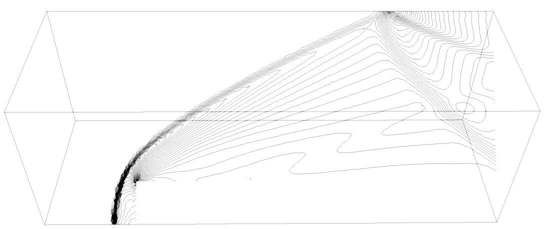
(b) Nonadapted $h = 0.04$ mesh at $t = 2.0$.



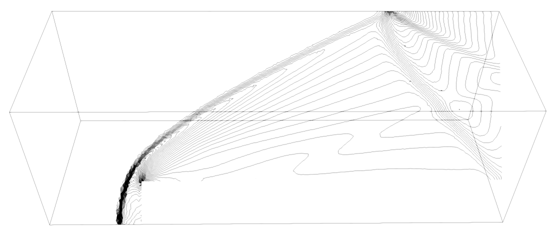
(c) Adapted $h = 0.04$ mesh at $t = 1.5$.



(d) Adapted $h = 0.04$ mesh at $t = 2.0$.

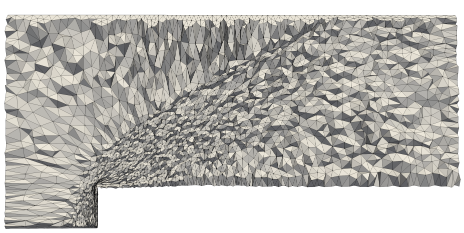


(e) Fine (nonadapted) $h = 0.015$ mesh at $t = 1.5$.

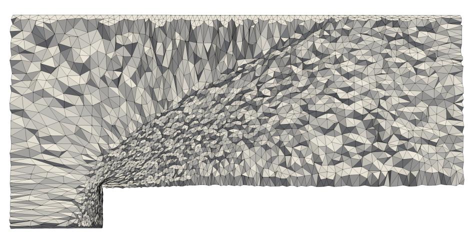


(f) Fine (nonadapted) $h = 0.015$ mesh at $t = 2.0$.

Figure 15: Three-dimensional forward facing step mass density contour lines at $t = 1.5$ and $t = 2.0$.

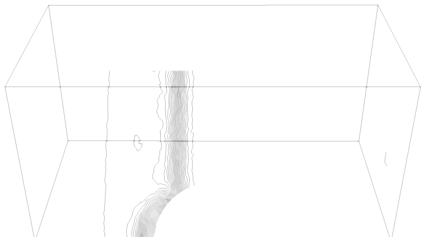


(a) Adapted $h = 0.04$ mesh at $t = 1.5$.

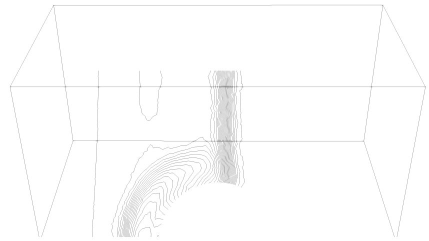


(b) Adapted $h = 0.04$ mesh at $t = 2.0$.

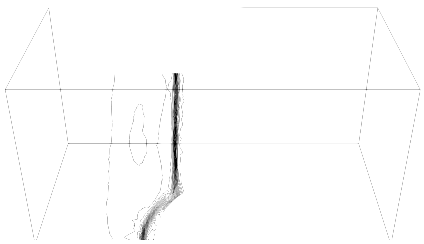
Figure 16: Three-dimensional forward facing step adapted meshes at $t = 1.5$ and $t = 2.0$.



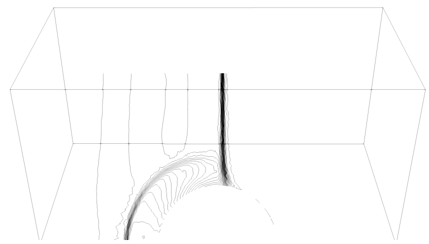
(a) Nonadapted $h = 0.05$ mesh at $t = 0.5$.



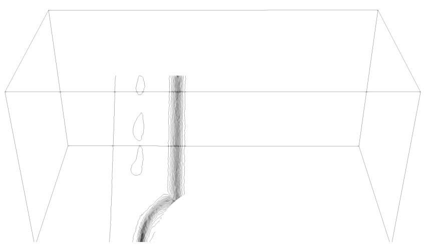
(b) Nonadapted $h = 0.05$ mesh at $t = 1.0$.



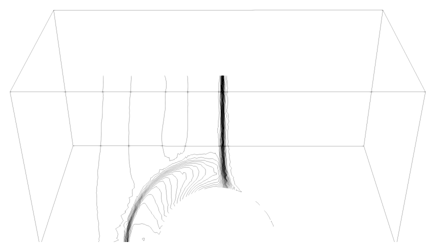
(c) Adapted $h = 0.05$ mesh at $t = 0.5$.



(d) Adapted $h = 0.05$ mesh at $t = 1.0$.

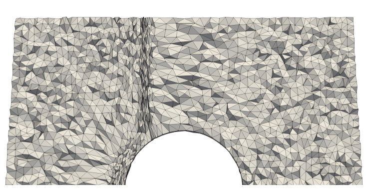


(e) Fine(nonadapted) $h = 0.02$ mesh at $t = 0.5$.

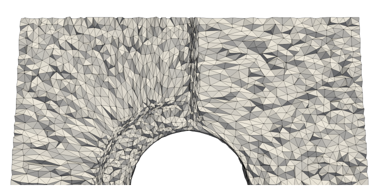


(f) Fine (nonadapted) $h = 0.02$ mesh at $t = 1.0$.

Figure 17: Shock-sphere interaction mass density contour lines at $t = 0.5$ and $t = 1.0$.

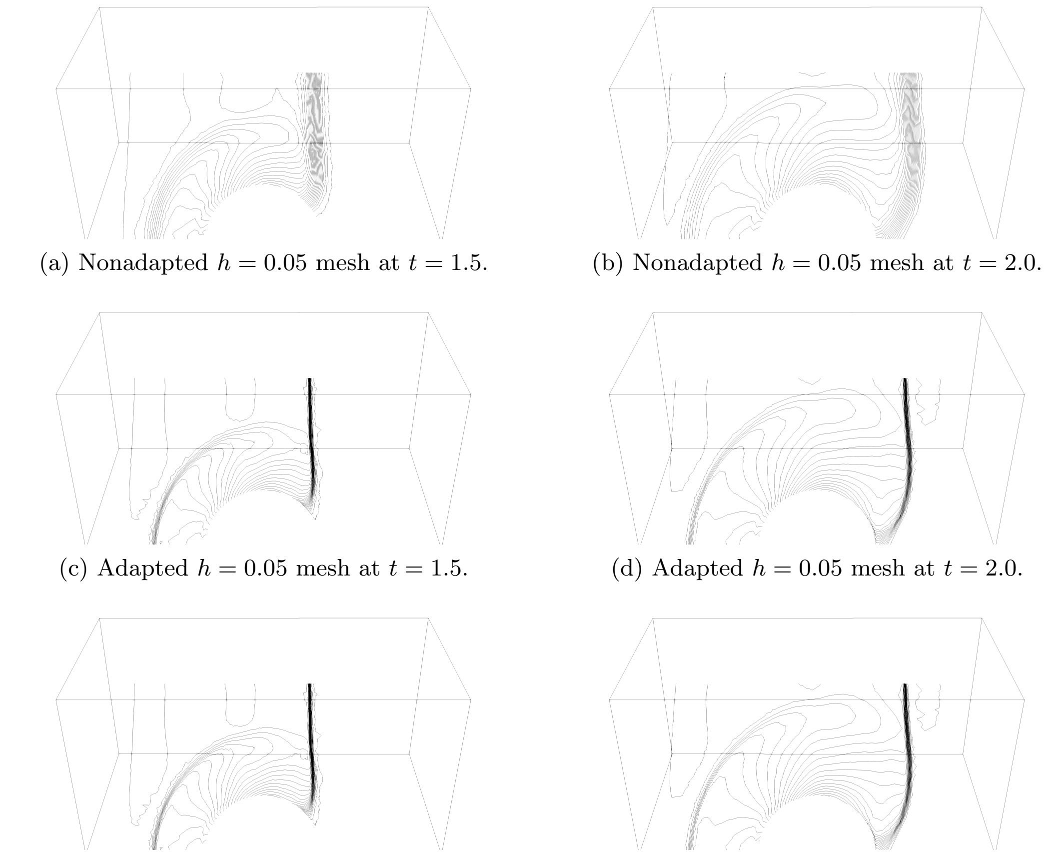


(a) Adapted $h = 0.05$ mesh at $t = 0.5$.



(b) Adapted $h = 0.05$ mesh at $t = 1.0$.

Figure 18: Shock-sphere interaction adapted meshes at $t = 0.5$ and $t = 1.0$.



(a) Nonadapted $h = 0.05$ mesh at $t = 1.5$. (b) Nonadapted $h = 0.05$ mesh at $t = 2.0$.

(c) Adapted $h = 0.05$ mesh at $t = 1.5$. (d) Adapted $h = 0.05$ mesh at $t = 2.0$.

(e) Fine (nonadapted) $h = 0.02$ mesh at $t = 1.5$. (f) Fine (nonadapted) $h = 0.02$ mesh at $t = 2.0$.

Figure 19: Shock-sphere interaction mass density contour lines at $t = 1.5$ and $t = 2.0$.

(a) Adapted $h = 0.05$ mesh at $t = 1.5$. (b) Adapted $h = 0.05$ mesh at $t = 2.0$.

Figure 20: Shock-sphere interaction adapted meshes at $t = 1.5$ and $t = 2.0$.

a projection step on the curved boundary parametric model. The iterative correction scheme allows to obtain valid meshes both in the volume and on the curved boundaries, and does not depend either on the specific choice of the mesh PDE model or the boundary geometry representation.

The reference domain formulation for mesh movement produces sufficiently adapted meshes in as few as ten Jacobi iterations per time step during an unsteady flow simulation. While the a-posteriori relaxation algorithm is akin to a forward substitution algorithm, and thus formally dependent from the node ordering, this doesn't appear to spoil the adaptation pattern in any of our tests. We show the successful generation of valid adapted mesh on three-dimensional cases with moving shock waves. While the computational time overhead with respect to the original unadapted mesh is non-negligeable, it is more than acceptable when compared to the simulation times needed to achieve the same accuracy on discontinuous flow features on uniformely refined mesh. The attractiveness of the method rests in fact in its applicability on moving shocks, where an off-line mesh refinement approach would require to refine the mesh in most of the computational domain, and its easy coupling with ALE solvers, enabling solution conservation on the adapted meshes.

Limitations of this r-adaptation method are the same of the original two-dimensional formulation, namely the Laplacian models excessively pulls nodes towards non-convex boundaries and the displacement uncoupling in the multiple space directions can create sensible adaptation patterns for excessively strong adaptation parameters. Also, the choice of the parameters of the monitor function appear to be application dependent. possibly leading to excessive mesh stretching for same values of the parameters. In these extreme situations, the effect of the novel a-posteriori relaxation scheme allows nonetheless to recover a valid mesh by blocking mesh displacement in critical zones, allowing to continue the mesh movement at successive time steps as the flow features evolve away from the blocked mesh elements. We would like to remark that our limiting procedure is targeted at preserving mesh validity throughout the adaptation procedure. This means that a nodal displacement can be blocked if the volume of an adjacent element falls below an user-defined threshold, but the mesh remains valid. Thus, the vertex positions of the blocked elements can be relaxed either by a subsequent application of r-adaptation at the next time step as the monitor function moves (as it is often the case in the simulation of traveling waves), either by the application of standard smoothing algorithms, which are fundamentally simpler than untangling methods.

While a linear finite element approximation is sufficient to model the nodal degrees of freedom of straight-sided meshes, generalizations of the a-posteriori limiting method to curved meshes can be envisaged by increasing the degree of the finite element basis. This would require the formulation of a volume positivity predicate for the curved tetrahedron, which is outside the scope of this work.

Future research lines include the parallelization of the current method, for which no specific problems are envisaged, and the study of r-adaptation as a tool to complement h-adaptation in time-dependent simulations to somewhat reduce the overhead of the adaptation strategy.

Acknowledgements

We would like to recognize the fundamental contribution our colleague and friend Cécile Dobrzynski brought to this work.

References

[1] Frédéric Alauzet and Pascal Frey, *Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques*, Research Report RR-4759, INRIA, 2003.

[2] Frédéric Alauzet, *A parallel matrix-free conservative solution interpolation on unstructured tetrahedral meshes*, Computer Methods in Applied Mechanics and Engineering **299** (2016), 116 – 142.

[3] Luca Arpaia and Mario Ricchiuto, *Mesh adaptation by continuous deformation. basics: accuracy, efficiency, well balancedness*, Research Report RR-8666, Inria Bordeaux Sud-Ouest ; INRIA, January 2015.

[4] Luca Arpaia and Mario Ricchiuto, *r-adaptation for shallow water flows: conservation, well balancedness, efficiency*, Computers & Fluids **160** (2018), 175 – 203.

[5] ______, *Well balanced residual distribution for the ALE spherical shallow water equations on moving adaptive meshes*, Journal of Computational Physics **405** (2020), 109173.

[6] J.U. Brackbill, *An adaptive grid with directional control*, Journal of Computational Physics **108** (1993), no. 1, 38 – 50.

[7] A. E. Bryson and R. W. F. Gross, *Diffraction of strong shocks by cones, cylinders, and spheres*, Journal of Fluid Mechanics **10** (1961), no. 1, 1–16.

[8] Chris J. Budd, Weizhang Huang, and Robert D. Russell, *Adaptivity with moving grids*, Acta Numerica **18** (2009), 111–241.

[9] Hector D. Ceniceros and Thomas Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, Journal of Computational Physics **172** (2001), no. 2, 609 – 639.

[10] Guoxian Chen, Huazhong Tang, and Pingwen Zhang, *Second-order accurate godunov scheme for multicomponent flows on moving triangular meshes*, Journal of Scientific Computing **34** (2008), no. 1, 64–86.

[11] C. Dapogny, C. Dobrzynski, and P. Frey, *Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems*, Journal of Computational Physics **262** (2014), 358 – 378.

[12] C Dapogny, C. Dobrzynski, P. Frey, and A. Froehly, *Mmg platform*.

[13] Jean Donea, Antonio Huerta, J.-Ph. Ponthot, and A. Rodríguez-Ferran, *Arbitrary lagrangian–eulerian methods*, John Wiley & Sons, Ltd, 2004.

[14] D. Drikakis, D. Ofengeim, E. Timofeev, and P. Voionovich, *Computation of non-stationary shockwave/cylinder interaction using adaptive-grid methods*, Journal of Fluids and Structures **11** (1997), no. 6, 665 – 692.

[15] Arkady S Dvinsky, *Adaptive grid generation from harmonic maps on riemannian manifolds*, Journal of Computational Physics **95** (1991), no. 2, 450 – 476.

[16] Richard P. Dwight, *Robust mesh deformation using the linear elasticity equations*, Computational Fluid Dynamics 2006 (Berlin, Heidelberg) (Herman Deconinck and E. Dick, eds.), Springer Berlin Heidelberg, 2009, pp. 401–406.

[17] Ashley F Emery, *An evaluation of several differencing methods for inviscid fluid flow problems*, Journal of Computational Physics **2** (1968), no. 3, 306 – 331.

[18] Charbel Farhat, Philippe Geuzaine, and Céline Grandmont, *The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids*, Journal of Computational Physics **174** (2001), no. 2, 669 – 694.

[19] P.E. Farrell and J.R. Maddison, *Conservative interpolation between volume meshes by local galerkin projection*, Computer Methods in Applied Mechanics and Engineering **200** (2011), no. 1, 89 – 100.

[20] Meire Fortunato and Per-Olof Persson, *High-order unstructured curved mesh generation using the winslow equations*, J. Comput. Phys. **307** (2016), no. C, 1–14.

[21] A. Guardone, D. Isola, and G. Quaranta, *Arbitrary lagrangian eulerian formulation for two-dimensional flows using dynamic meshes with edge swapping*, Journal of Computational Physics **230** (2011), no. 20, 7706 – 7722.

[22] Hervé Guillard and Charbel Farhat, *On the significance of the geometric conservation law for flow computations on moving meshes*, Computer Methods in Applied Mechanics and Engineering **190** (2000), no. 11, 1467 – 1482.

[23] Glen Hansen, Andrew Zardecki, Doran Greening, and Randy Bos, *A finite element method for unstructured grid smoothing*, Journal of Computational Physics **194** (2004), no. 2, 611 – 631.

[24] D. Hermes and P.-O. Persson, *High-order solution transfer between curved triangular meshes*, 2018.

[25] Weizhang Huang, *Variational mesh adaptation: Isotropy and equidistribution*, Journal of Computational Physics **174** (2001), no. 2, 903 – 924.

[26] Weizhang Huang and Lennard Kamenski, *On the mesh nonsingularity of the moving mesh pde method*, no. 87, 1887–1911.

[27] Weizhang Huang and Robert D. Russell, *Adaptive moving mesh methods*, Applied Mathematical Sciences **174** (2011), no. 174, Applied Mathematical Sciences.

[28] D. Isola, A. Guardone, and G. Quaranta, *Finite-volume solution of two-dimensional compressible flows over dynamic adaptive grids*, Journal of Computational Physics **285** (2015), 1 – 23.

[29] A.A. Johnson and T.E. Tezduyar, *Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces*, Computer Methods in Applied Mechanics and Engineering **119** (1994), no. 1, 73 – 94.

[30] P. Knupp and S. Steinberg, *Fundamentals of grid generation*, The Fundamentals of Grid Generation, Taylor & Francis, 1993.

[31] M. Kucharik and M. Shashkov, *Extension of efficient, swept-integration-based conservative remapping method for meshes with changing connectivity*, International Journal for Numerical Methods in Fluids **56** (2008), no. 8, 1359–1365.

[32] Milan Kucharik, Mikhail Shashkov, and Burton Wendroff, *An efficient linearity-and-bound-preserving remapping method*, Journal of Computational Physics **188** (2003), no. 2, 462–471.

[33] Ruo Li, Tao Tang, and Pingwen Zhang, *A moving mesh finite element algorithm for singular problems in two and three space dimensions*, Journal of Computational Physics **177** (2002), no. 2, 365 – 393.

[34] G. Liao, *Variational approach to grid generation*, Numerical Methods for Partial Differential Equations **8** (1992), no. 2, 143–147.

[35] Michael A. Park, Adrien Loseille, Joshua Krakos, Todd R. Michal, and Juan J. Alonso, *Unstructured grid adaptation: Status, potential impacts, and recommended investments towards cfd 2030*, AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, June 2016, pp. –.

[36] B. Re, C. Dobrzynski, and A. Guardone, *An interpolation-free ale scheme for unsteady inviscid flows computations with large boundary displacements over three-dimensional adaptive grids*, Journal of Computational Physics **340** (2017), 26 – 54.

[37] M Sun, T Saito, K Takayama, and H Tanno, *Unsteady drag on a sphere by shock wave loading*, Shock waves **14** (2005), no. 1-2, 3–9.

[38] Huazhong Tang and Tao Tang, *Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM Journal on Numerical Analysis **41** (2003), no. 2, 487–515.

[39] Tao Tang, *Moving mesh methods for computational fluid dynamics*, Contemporary mathematics **383** (2005), 141–174.

[40] H Tanno, K Itoh, T Saito, A Abe, and K Takayama, *Interaction of a shock with a sphere suspended in a vertical shock tube*, Shock Waves **13** (2003), no. 3, 191–200.

[41] PD Thomas and CK Lombard, *Geometric conservation law and its application to flow computations on moving grids*, AIAA journal **17** (1979), no. 10, 1030–1037.

[42] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin, *Numerical grid generation: foundations and applications*, North-Holland, 1985.

[43] Joe F Thompson, Frank C Thames, and C Wayne Mastin, *Tomcat — a code for numerical generation of boundary-fitted curvilinear coordinate systems on fields containing any number of arbitrary two-dimensional bodies*, Journal of Computational Physics **24** (1977), no. 3, 274 – 302.

[44] Joe F Thompson, Frank C Thames, and Charles Wayne Mastin, *Boundary-fitted curvilinear coordinate systems for solution of partial differential equations on fields containing any number of arbitrary two-dimensional bodies*, Tech. Report NASA-CR-2729, NASA, 1977.

[45] Thomas Toulorge, Christophe Geuzaine, Jean-François Remacle, and Jonathan Lambrechts, *Robust untangling of curvilinear meshes*, Journal of Computational Physics **254** (2013), 8 – 26.

[46] John G Trulio and Kenneth R Trigger, *Numerical solution of the one-dimensional lagrangian hydrodynamic equations*, Tech. report, California. Univ., Livermore, CA (United States). Lawrence Radiation Lab., 1961.

[47] Michael Turner, Joaquim Peiró, and David Moxey, *Curvilinear mesh generation using a variational framework*, Computer-Aided Design **103** (2018), 73 – 91, 25th International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[48] Bram van Leer, *Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method*, Journal of Computational Physics **32** (1979), no. 1, 101 – 136.

[49] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L Mitchell, *Curved pn triangles*, Proceedings of the 2001 symposium on Interactive 3D graphics, 2001, pp. 159–166.

[50] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell, *Curved pn triangles*, Proceedings of the 2001 Symposium on Interactive 3D Graphics (New York, NY, USA), I3D '01, Association for Computing Machinery, 2001, p. 159–166.

[51] Alan M Winslow, *Adaptive-mesh zoning by the equipotential method*, Tech. report, Lawrence Livermore National Lab., CA (USA), 1981.

[52] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, Journal of Computational Physics **54** (1984), 115–173.

[53] S. Étienne, A. Garon, and D. Pelletier, *Perspective on the geometric conservation law and finite element methods for ale simulations of incompressible flow*, Journal of Computational Physics **228** (2009), no. 7, 2313 – 2333.