Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 222 (2007) 592–623
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations

F. Alauzet [a,*], P.J. Frey [b], P.L. George [a], B. Mohammadi [c]

[a] INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France
[b] UPMC/Lab. J-L Lions, BP 187, 75252 Paris Cedex 05, France
[c] Université Montpellier II, Math. Dept., CC51, 34095 Montpellier Cedex 5, France

Received 3 November 2005; received in revised form 2 August 2006; accepted 11 August 2006
Available online 25 October 2006

Abstract

This paper deals with the adaptation of unstructured meshes in three dimensions for transient problems with an emphasis on CFD simulations. The classical mesh adaptation scheme appears inappropriate when dealing with such problems. Hence, another approach based on a new mesh adaptation algorithm and a metric intersection in time procedure, suitable for capturing and track such phenomena, is proposed. More precisely, the classical approach is generalized by inserting a new specific loop in the main adaptation loop in order to solve a transient fixed point problem for the mesh–solution couple. To perform the anisotropic metric intersection operation, we apply the simultaneous reduction of the corresponding quadratic form. Regarding the adaptation scheme, an anisotropic geometric error estimate based on a bound of the interpolation error is proposed. The resulting computational metric is then defined using the Hessian of the solution. The mesh adaptation stage (surface and volume) is based on the generation, by global remeshing, of a unit mesh with respect to the prescribed metric. A 2D model problem is used to illustrate the difficulties encountered. Then, 2D and 3D complexes and representative examples are presented to demonstrate the efficiency of this method.

Keywords: Mesh adaptation; Time-dependent problems; Anisotropic metrics; Geometric error estimate; CFD; Euler equations; Unit mesh

1. Introduction

Nowadays, reducing the complexity of numerical simulations is always a crucial stake despite the constant increase of computer power. Mesh adaptation is one among the various methods considered to reduce the simulation complexity. The aim is to control the numerical solution accuracy by modifying the domain discretization according to size and directional constraints. For time-dependent simulations, mesh adaptation is even more crucial as the solution is usually not known a priori and as physical phenomena progress arbitrarily in

* Corresponding author. Tel.: +33 1 39 63 57 93; fax: +33 1 39 63 58 82.
E-mail address: Frederic.Alauzet@inria.fr (F. Alauzet).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.08.012

the (whole) computational domain. Without adaptation this would imply the use of a uniformly fine mesh size everywhere in order to preserve the solution accuracy.

Mesh adaptation is a non-linear problem linked to the solution. Therefore, it seems intuitive to propose an iterative procedure to solve this problem. Indeed, for a steady simulation, an adaptive computation is carried out via a mesh adaptation loop inside which an algorithmic (or iterative) convergence of the couple mesh–solution is sought, in the sense that the solution is algorithmically (or iteratively) converging toward the steady state solution and the mesh is converging toward the adapted mesh associated to this converged steady state. In the following sections, we will refer to this scheme as the *classical mesh adaptation algorithm*. Nevertheless, such an approach is no longer valid for time-dependent simulations as the physical solution progresses in time. We will point out that, in general, the classical mesh adaptation algorithm reveals intrinsically inadequate when dealing with CFD transient simulations. Indeed, roughly speaking, the mesh is always "left behind" with respect to the solution, i.e., a phase shift in time between mesh and solution occurs. In order to reduce this shift, a possible corrective method could be to adapt the mesh frequently, but in this case, relatively to each mesh modification, an important source of error due to the interpolation (or transfer) of the solution from the old mesh on the new one is introduced. Consequently, the solution accuracy is strongly related to the number of adaptation. Moreover, the choice of the number of mesh adaptation is arbitrary. To overcome these problems, a new general mesh adaptation scheme is proposed, specifically intended for time-dependent problems. This approach can be perceived as a generalization of the steady mesh adaptation algorithm as a supplementary loop is introduced inside the classical mesh adaptation algorithm. This extra loop introduces an implicit coupling between the mesh and the solution by means of a transient fixed point. It allows us to solve the non-linear problem of mesh adaptation for unsteady simulations. Formally speaking, in this procedure, it is crucial: (i) to predict the progression of the physical phenomena in the domain and (ii) to refine (adapt) the mesh in all regions where these phenomena progress. With this completely automatic method, the spatial part and the temporal part of the (truncation) error are controlled.

To be useful, this algorithm must be evaluated on a real problem considered as representative of time-dependent simulations, for example, the progression of a priori unpredictable moving phenomena in a complex fixed geometry. Therefore, we consider here the problem of non-linear shock waves propagation in a geometry, which is a generalization of the Sod's shock tube Riemann problem. The 2D case of this example will allow us to discuss the relevant issues of the mesh adaptation for the unsteady simulations.

Over the last few years, a rather large number of papers have been published dealing with mesh adaptation for numerical simulations.[1] In the context of steady simulations in three dimensions, several works have been done in the case of unstructured isotropic mesh adaptation [1–4] and more recently concerning unstructured anisotropic mesh adaptation [5–8]. However, only a small number of papers have addressed time-dependent problems. Basically, three different approaches can be distinguished:

- adapt the mesh frequently in order to contain the solution progression within refined regions [9] and introduce a safety area around critical regions [10–12]. This method is based on coarsening/refinement techniques, without node displacement so as to reduce interpolation error;
- construct meshes using an unsteady mesh adaptation algorithm [13,14]. This method is based on local or global remeshing techniques and the error is estimated every n_1 flow solver time steps and the mesh possibly adapted. Nevertheless, the mesh is actually adapted every $n_2 > n_1$ time steps;
- and more recently, use a local adaptive remeshing algorithm so as to be able to construct an anisotropic mesh and adapt the mesh frequently in order to contain the solution evolution within refined regions [15,16]. However, we still have to be careful with interpolation error even if the solution interpolation is proceeded after each mesh modification.

All these approaches involve a large number of adaptations, although, they do not prevent the introduction of errors due to the interpolation (or transfer) of the solution from the old mesh on the new one for global remeshing algorithms. Moreover, the first and the third approaches do not control explicitly the error on

[1] On a simple google query around 1000 papers can be found.

the solution as they perform an arbitrary number of adaptations. Notice that the first one cannot be extended to anisotropic mesh adaptation.

In this context, a new mesh adaptation scheme based on global remeshing has been proposed to control the accuracy of the solution while reducing the number of mesh adaptations [17]. To this end, a preliminary approach based on the resolution of a transient fixed point problem has been described. This paper also focused on the adaptive meshing techniques and metric concern in the isotropic case. 2D results have been presented to validate the new scheme and a first experience in 3D on a non-curved geometry has been shown.

The present paper is concentrated on the application of mesh adaptation for transient CFD simulations and proposes an improved version of the transient fixed point-based mesh adaptation by combining the classical mesh adaptation and the transient fixed point scheme of [17]. This new method is motivated by a theoretical analysis in 1D on the advection equation. This is emphasized on realistic three-dimensional simulation with a complex curved geometry. Furthermore, this paper presents mesh adaptation techniques to apply this algorithm in the anisotropic case, even if only two-dimensional results are proposed. Our approach is independent of the type of problem at hand as it is based on an anisotropic geometric error estimate [5].

To solve the non-linear problem of mesh adaptation for unsteady simulation, a novel algorithm combining the classical mesh adaptation and the transient fixed point scheme of [17] is proposed. This new algorithm allows a notable gain in efficiency to be obtained. Moreover, the resulting scheme has the advantage to be able to deal with steady and unsteady simulations, in other words this scheme is a generalization of the classical mesh adaptation algorithm.

More precisely, this automatic method is based on an implicit iterative algorithm coupling with the mesh adaptation scheme in order to predict the solution evolution in the computational domain and to suitably mesh all the regions of the solution's progression. To predict the solution's progression, the idea is to solve at each iteration of the adaptation loop a transient fixed point problem for the mesh–solution couple. Knowing then the solution evolution throughout a short period of time, the mesh is adapted in all the regions where phenomena progress so as to preserve the solution accuracy. To this end, a metric intersection in time procedure is introduced in the metric construction. In other words, the time variable is implicitly introduced in the error estimate in order to control the error throughout the computation. As this method gives an answer to the prediction of physical phenomena, it controls the regularity of the structures moving in the domain.

In the following sections, we present and discuss the proposed approach. In Section 2, we recall the classical mesh adaptation algorithm and review the main stages of the adaptation procedure in the anisotropic case. In Section 3, motivated by a theoretical analysis on the 1D advection equation, the new mesh adaptation scheme is proposed to overcome the drawbacks of the classical approach. The classical and the new mesh adaptation scheme are thoroughly analyzed on the 2D model example, Section 4. Finally, in Sections 5 and 6, realistic 2D and 3D application examples are presented to emphasize the efficiency of the proposed scheme.

2. Mesh adaptation

In this section, we will recall the classical mesh adaptation scheme and the various stages involved. More details can be found in [5,18]. Usually, mesh adaptation provides a way of controlling the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. More precisely, the goal is to equally distribute the approximation (interpolation) error in all directions over the computational mesh.

2.1. The classical mesh adaptation scheme

For stationary problems, the classical mesh adaptation scheme aims at finding a fixed point for the mesh–solution couple. In other words, the goal is to algorithmically converge towards the stationary solution of the problem and similarly towards the corresponding invariant adapted mesh throughout an iterative algorithm on a sequence of consecutively adapted meshes. This iterative scheme is illustrated in Fig. 1, where i indicates the adaptation iteration index and where $\mathscr{H}$, $\mathscr{S}$, $\mathscr{S}^0$ and $\mathscr{M}$ denote the mesh, the solution, the initial solution at each iteration and the metric, respectively.

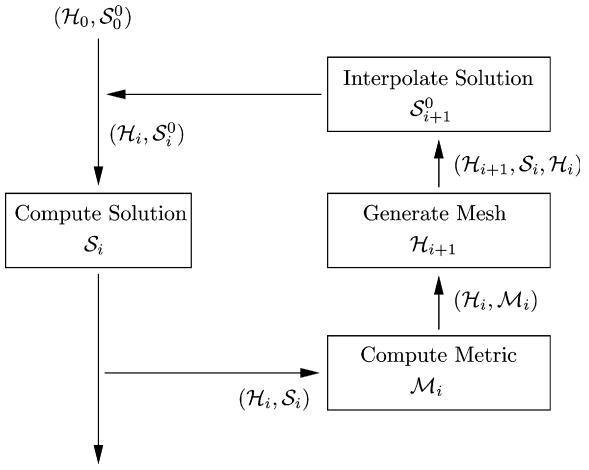

Fig. 1. Classical mesh adaptation scheme.

At each stage, a numerical solution is computed on the current mesh and has to be analyzed by means of an error estimate. The mesh adaptation is based on the edge length computation with respect to a discrete anisotropic metric specified at the mesh vertices. This metric is defined via a geometric error estimate that translates the solution variations into elements sizes and directions. Next, a unit (adapted) mesh is generated with respect to this metric. Finally, the solution is interpolated linearly on the new mesh. This procedure is repeated until the algorithmic convergence of the solution and of the mesh is achieved.

2.2. Flow modeling

Here we consider a typical CFD simulation for flow problems modeled by the Euler equations. Assuming that the gas is perfect, non-viscous and that there is no thermal diffusion, the Euler equations with gravity for mass, momentum and energy conservation read:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0,$$
$$\frac{\partial (\rho \vec{U})}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) + \nabla p = \rho g,$$
$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((\rho E + p)\vec{U}) = \rho \vec{g} \cdot \vec{U},$$

where ρ denotes the density, $\vec{U}$ the velocity vector, $E = T + \frac{\|\vec{U}\|^2}{2}$ the total energy and $p = (\gamma - 1)\rho T$ the pressure with $\gamma = 1.4$ the ratio of specific heats, T the temperature and $\vec{g}$ the gravity vector.

The Euler system is solved by means of a mixed-element-volume scheme on unstructured tetrahedral meshes [19]. The scheme is vertex-centered and uses a particular edge-based formulation with upwind elements. This flow solver employes a HLLC approximate Riemann solver to compute numerical fluxes [20]. High-order scheme is derived according to a MUSCL (monotone upwind schemes for conservation laws) type method using downstream and upstream tetrahedra. This approach is compatible with vertex-centered and edge-based formulations, allowing rather easy and, importantly, inexpensive high-order extensions of monotone upwind schemes. This low diffusion scheme is third-order on structured meshes. The flux integration based on the edges and their corresponding upwind elements (crossed by the edge) is a key-feature in order to preserve the positivity of the density for vertex-centered formulation. The MUSCL type method is combined with a generalization of the Superbee limiter with three entries to guarantee the TVD (total variation diminishing) property to the scheme [21].

An explicit time stepping algorithm is used by means of a 4-stage, 3-order strong-stability-preserving (SSP) Runge–Kutta scheme that allows us to use a *CFL* coefficient up to 2 [22]. In practice, we consider a *CFL* equal to 1.8.

2.3. Metric computation

We will focus here on the construction of the anisotropic metric tensor. The latter is defined at the mesh vertices using a geometric error estimate. A least square approach is presented to compute the Hessian of the solution and, for CFD simulations, we will describe an error estimate normalization. Finally, the intersection operation for anisotropic metrics is explained.

2.3.1. A geometric error estimate

As for the elliptic problems, we shall assume here that controlling the interpolation error allows us to control the approximation error. Hence, we deliberately based our anisotropic geometric error estimate on the interpolation error. The error estimate aims at defining a discrete metric field that prescribes size and stretching requirements for the mesh adaptation procedure. Consequently, in an adapted mesh the interpolation error is equally distributed in all directions. More precisely, for each mesh element K, the anisotropic error interpolation bound involves the second derivatives of the variable u:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant c_d \max_{x\in K} \max_{e\in E_K} \langle \vec{e}, |H_u(x)|\vec{e}\rangle = \varepsilon_K, \tag{1}$$

where c_d is a constant related to the dimension, E_K is the set of edges of K and $|H_u| = \mathscr{R}|\Lambda|\mathscr{R}^{-1}$ is the absolute value of the Hessian of the variable u ($\mathscr{R}$ being the matrix of eigenvectors and $|\Lambda| = \mathrm{diag}(|\lambda_i|)$ being the absolute value of the matrix of eigenvalues).

Remark 2.1. The solution can be perceived as a Cartesian surface embedded in $\mathbb{R}^{d+1}$, the error estimate measuring then the gap between a piecewise linear approximation and the underlying surface. This leads to the name of *geometric* error estimate, indicating also that it is problem-independent.

2.3.2. Metric definition

The natural dot product of $\mathbb{R}^n$ is denoted by $\langle,\rangle$. A *metric tensor* (or simply a *metric*) $\mathscr{M}$ in $\mathbb{R}^n$ is a $n\times n$ symmetric definite positive matrix. This means that $\mathscr{M}$ verifies:

(i) if $\langle \vec{u}, \mathscr{M}\vec{u}\rangle = 0$ then $\vec{u} = 0$,
(ii) $\forall \vec{u} \in \mathbb{R}^n$, we have $\langle \vec{u}, \mathscr{M}\vec{u}\rangle \geqslant 0$.

From this definition, the *scalar product* of two vectors in $\mathbb{R}^n$ is defined according to a metric $\mathscr{M}$ as:

$$\langle \vec{u}, \vec{v}\rangle_{\mathscr{M}} = \langle \vec{u}, \mathscr{M}\vec{v}\rangle = {}^t\vec{u}\mathscr{M}\vec{v} \in \mathbb{R}.$$

With this notion, it is easy to define the associated *norm* of a vector in $\mathbb{R}^n$:

$$\|\vec{u}\|_{\mathscr{M}} = \sqrt{\langle \vec{u}, \vec{u}\rangle_{\mathscr{M}}} = \sqrt{{}^t\vec{u}\mathscr{M}\vec{u}},$$

which actually measures the length of the vector $\vec{u}$ in the metric $\mathscr{M}$.

The notion of metric is equivalent to the notion of quadratic form and a metric $\mathscr{M}$ could be geometrically represented by its associated unit ball, an ellipsoid, defined by $\mathscr{E}_{\mathscr{M}} = \{M | \sqrt{{}^t\overrightarrow{OM}\mathscr{M}\overrightarrow{OM}} = 1\}$ where O is the centre of the ellipsoid.

2.3.3. Metric construction

A discrete metric approximation which uses the mesh vertices as support is considered. Let $h_{\min}$ and $h_{\max}$ be the minimal and the maximal mesh element size, respectively, and let ε be the desired interpolation error. Then, according to Relation (1), we define at each mesh vertex the anisotropic metric tensor $\mathscr{M}$ as:

$$\mathscr{M} = \mathscr{R}\tilde{\Lambda}\mathscr{R}^{-1}, \quad \text{where } \tilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_i) \text{ and } \tilde{\lambda}_i = \min\left(\max\left(\frac{c|\lambda_i|}{\varepsilon}, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right).$$

Introducing a minimal and a maximal element size is a practical way to avoid unrealistic metrics. It also allows us to control the time-stepping in the flow solver. In other words, in view of equally distributing the interpolation error over the mesh, we have modified the scalar product that underlies the notion of distance used in mesh generation algorithms (where the local metric $\mathscr{M}$ replaces the usual Euclidean metric).

2.3.4. Error estimate in CFD

Physical phenomena can involve large scale variations (e.g. multi-scale phenomena, recirculation, and weak and strong shocks). It is thus difficult to capture the weakest phenomena via mesh adaptation, and even harder to do it when, for instance in CFD, strong shocks are located in the flow because they hide weak phenomena. Capturing such weak phenomena is crucial for obtaining an accurate solution by taking into account all phenomena interactions in the main flow area.

A local error estimation can overcome this problem [23]. Relation (1) is normalized using the local absolute value of the variable u:

$$\left\| \frac{u - \Pi_h u}{|u|_\epsilon} \right\|_{\infty,K} \leqslant c \max_{x \in K} \max_{\vec{e} \in E_K} \left\langle \vec{e}, \frac{|H_u(x)|}{|u(x)|_\epsilon} \vec{e} \right\rangle, \tag{2}$$

where $|u|_\epsilon = \max(|u|, \epsilon \|u\|_{\infty,\Omega})$ with $\epsilon \ll 1$ a constant. The term $\epsilon \|u\|_{\infty,\Omega}$ introduces a cut off to avoid dividing by zero.

However, in the context of anisotropic mesh adaptation for compressible flow, capturing weak phenomena by means of Relation (2) leads to mesh isotropically strong shocks. This is due to the discretization of the solution that introduces virtual oscillations in the direction parallel to the shock. These oscillations have a magnitude of the same order as weak phenomena. We propose to filter these oscillations with the local gradient of the solution in order to preserve the anisotropy. To this end, we suggest the following error estimate:

$$\left\| \frac{u - \Pi_h u}{\gamma |u|_\epsilon + (1-\gamma)\bar{h} \|\nabla u\|_2} \right\|_{\infty,K} \leqslant c \max_{x \in K} \max_{\vec{e} \in E_K} \left\langle \vec{e}, \frac{|H_u(x)|}{\gamma |u(x)|_\epsilon + (1-\gamma)\bar{h} \|\nabla u(x)\|_2} \vec{e} \right\rangle, \tag{3}$$

where $\bar{h}$ is the diameter (i.e., the length of its largest edge) of element K and γ is a parameter belongs to $[0,1]$ that will be considered close to zero if strong shocks are involved in the flow. Notice that γ will be chosen equal to 1 in the case of isotropic mesh adaptation.

2.3.5. Metric intersection

Previously, the metric construction for a scalar variable has been shown. However, it is often necessary to take several variables into account in the metric construction, so as to capture different physical phenomena. Indeed, some phenomena are only represented by specific variables. When several variables are considered, several metrics are specified at each mesh vertex. All these metric tensors must be reduced to a single one for mesh generation purposes. To this end, a metric intersection procedure is used. Nevertheless, these variables could have a different meaning or a different physical nature. It then becomes necessary to have dimensionless variables, as for Inequality (2).

Formally speaking, let $\mathscr{M}_1$ and $\mathscr{M}_2$ be two metric tensors given at a vertex P. The metric tensor $\mathscr{M}_{1\cap 2}$ corresponding to the intersection of $\mathscr{M}_1$ and $\mathscr{M}_2$ must be such that the interpolation error for each variable is bounded by the given tolerance value. To this end, we use the simultaneous reduction of the quadratic forms associated with the two metrics to obtain the intersected metric.

More precisely, the idea is to find a basis (e_1, e_2, e_3) such that $\mathscr{M}_1$ and $\mathscr{M}_2$ are congruent to a diagonal matrix in this basis, and then to deduce the intersected metric. To do so, the matrix $\mathscr{N} = \mathscr{M}_1^{-1}\mathscr{M}_2$ is introduced. $\mathscr{N}$ is diagonalizable in $\mathbb{R}$ and the sought basis is given by the normalized eigenvectors of $\mathscr{N}$ denoted by e_1, e_2 and e_3. The terms of the diagonal matrices that are associated to the metrics $\mathscr{M}_1$ and $\mathscr{M}_2$ in this basis are obtained with the Rayleigh formula:

$$\lambda_i = {}^t e_i \mathscr{M}_1 e_i \quad \text{and} \quad \mu_i = {}^t e_i \mathscr{M}_2 e_i, \quad \text{for } i = 1,3.$$

Let $\mathscr{P} = (e_1 e_2 e_3)$ be the matrix whose columns are formed by the eigenvectors $\{e_i\}_{i=1,3}$ of $\mathscr{N}$, $\mathscr{P}$ is invertible as (e_1, e_2, e_3) is a basis of $\mathbb{R}^3$. Therefore, we have:

$$\mathscr{M}_1 = {}^t\mathscr{P}^{-1}\begin{pmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{pmatrix}\mathscr{P}^{-1} \quad \text{and} \quad \mathscr{M}_2 = {}^t\mathscr{P}^{-1}\begin{pmatrix}\mu_1 & 0 & 0\\ 0 & \mu_2 & 0\\ 0 & 0 & \mu_3\end{pmatrix}\mathscr{P}^{-1}.$$

The resulting intersected metric $\mathscr{M}_1 \cap \mathscr{M}_2$ is given by:

$$\mathscr{M}_{1\cap 2} = \mathscr{M}_1 \cap \mathscr{M}_2 = {}^t\mathscr{P}^{-1}\begin{pmatrix}\max(\lambda_1,\mu_1) & 0 & 0\\ 0 & \max(\lambda_2,\mu_2) & 0\\ 0 & 0 & \max(\lambda_3,\mu_3)\end{pmatrix}\mathscr{P}^{-1}.$$

Geometrically speaking, let $\mathbf{M_3}$ be the set of the 3×3 metric tensors, then the simultaneous reduction gives the largest ellipsoid included in the geometrical intersection of the two ellipsoids associated to the metrics $\mathscr{M}_1$ and $\mathscr{M}_2$ (cf. Fig. 2):

$$\mathscr{E}_{\mathscr{M}_{1\cap 2}} = \sup_{\mathscr{M}_i \in \mathbf{M}_d} \mathscr{E}_{\mathscr{M}_i} \subset \mathscr{E}_{\mathscr{M}_1} \cap \mathscr{E}_{\mathscr{M}_2}.$$

2.3.6. Evaluation of the Hessian matrix

The bound of the interpolation error, Relation (1), involves the Hessian matrix of the exact solution u of the problem, and therefore, it appears in the construction of the metric. Indeed, constructing a good (reasonable) metric requires computing the Hessian matrix accurately. But, practically, only the discrete solution of the problem u_h is available and the exact one u is not known. Moreover, in our case, u_h is only piecewise linear.

In order to evaluate the Hessian matrix, the idea is to reconstruct a high order solution u^* from u_h that is two times derivable and to approximate the approximation error by the gap between the discrete and reconstructed solutions:

$$\|u - u_h\| \approx \|u^* - u_h\|.$$

By analogy with Relation (1), we have the following bound:

$$\|u^* - \Pi_h u^*\|_{\infty,K} \leqslant c \max_{x\in K} \max_{e\in E_K} \langle \vec{e}, |H_{u^*}(x)|\vec{e}\rangle, \tag{4}$$

where the right-hand side term is known. Notice that in practice, it is not necessary to reconstruct explicitly u^*, it is enough to evaluate the Hessian of u^*. Here, we propose an approach based on a Taylor expansion and the solving of a linear system by means of a least-squares approximation.

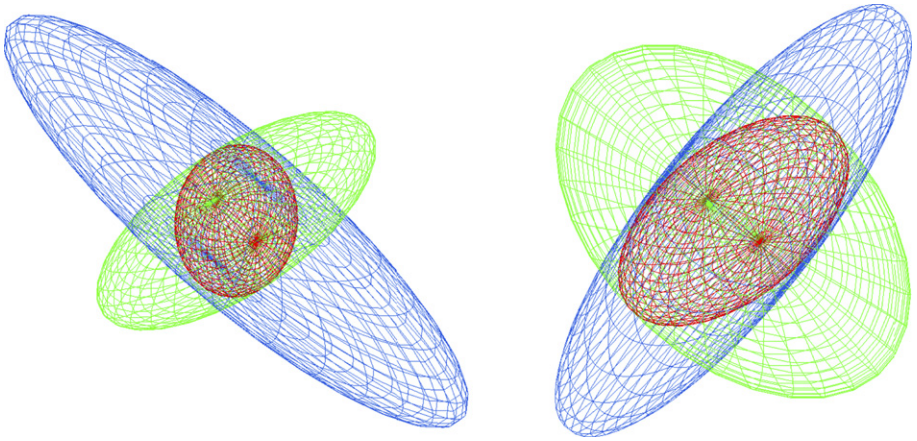

Fig. 2. Views illustrating the metric intersection procedure with the simultaneous reduction in three dimensions. In red, the resulting metric of the intersection of the blue and green metrics. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Let P be a mesh vertex and let $\mathscr{B}(P)$ be the ball of P (i.e., all mesh vertices connected to the vertex P). By considering a Taylor expansion of u^* at a vertex $P_i \in \mathscr{B}(P)$ and truncated at the order 2, we can write the following relation:

$$u_i^* = u^* + \overrightarrow{PP_i} \cdot \nabla u^*(P) + \frac{1}{2}\langle {}^t\overrightarrow{PP_i}, H_{u^*}(P)\overrightarrow{PP_i}\rangle \Longleftrightarrow \frac{1}{2}\langle {}^t\overrightarrow{PP_i}, H_{u^*}(P)\overrightarrow{PP_i}\rangle = u_i^* - u^* - \overrightarrow{PP_i} \cdot \nabla u^*(P)$$

with the notations $u^* = u^*(P)$ and $u_i^* = u^*(P_i)$. This relation can be developed as follows:

$$\frac{1}{2}\left(ax_i^2 + 2bx_iy_i + 2cx_iz_i + dy_i^2 + 2ey_iz_i + fz_i^2\right) = u_i^* - u^* - (\alpha x_i + \beta y_i + \gamma z_i), \tag{5}$$

using the notations:

$$\overrightarrow{PP_i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \quad \nabla u^*(P) = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \quad \text{and} \quad H_{u^*}(P) = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}.$$

This leads to a usually over-determined system[2] of the form:

$$AX = B, \quad \text{with } {}^tX = (\,a \quad b \quad c \quad d \quad e \quad f\,),$$

where A is a $n \times 6$ matrix ($n = \mathrm{Card}(\mathscr{B}(P))$) function of (x_i, y_i, z_i) and B is a vector of dimension n given by the right-hand side of the Relation (5), and function of $(\alpha, \beta, \gamma, x_i, y_i, z_i, u, u_i)$. This system is solved using a least-square approximation, i.e., it consists in minimizing the distance between the vectors AX and B of $\mathbb{R}^n$ by minimizing the square of the Euclidean norm of their difference. The problem is then to:

$$\text{Find } X \in \mathbb{R}^6 \text{ such that } \|AX - B\|^2 = \inf_{Y \in \mathbb{R}^6} \|AY - B\|^2.$$

It can be shown that the solution of this problem is the solution of the linear 6×6 system of normal equations:

$${}^tAAX = {}^tAB.$$

The latter is then solved using a standard Gauss method.

Remark 2.2. If, in some peculiar cases, the system is under-determined (i.e., $\mathrm{Card}(\mathscr{B}(P)) < 6$), additional vertices connected to the vertices of $\mathscr{B}(P)$ can be taken into account.

2.4. Mesh adaptation

In our approach, the adaptation of the current mesh is based on the specification of a discrete anisotropic metric tensor at each vertex. For these purposes, the standard Euclidean scalar product is modified according to a proper metric tensor field $\mathscr{M}$. The aim is then to generate a mesh such that all edges have a length of (or close to) one in the prescribed metric and such that all elements are almost regular. Such a mesh is called a *unit mesh*. Let P be a vertex and let $\mathscr{M}(P)$ be the metric at P, the length of the edge PX with respect to $\mathscr{M}(P)$ is defined as:

$$l_{\mathscr{M}(P)}(PX) = \langle \overrightarrow{PX}, \overrightarrow{PX}\rangle^{\frac{1}{2}}_{\mathscr{M}(P)} = \sqrt{{}^t\overrightarrow{PX}\,\mathscr{M}(P)\overrightarrow{PX}}.$$

As the metric is not uniform over the domain, we need to consider the metrics at the edge endpoints as well as all intermediate metrics along the edge. To achieve this, we assume that an edge PX has a local parametrization $PX = P + t\overrightarrow{PX}$ and we introduce its average length as:

$$l_{\mathscr{M}}(\overrightarrow{PX}) = \int_0^1 \sqrt{{}^t\overrightarrow{PX}\,\mathscr{M}(P + t\overrightarrow{PX})\overrightarrow{PX}}\,\mathrm{d}t. \tag{6}$$

[2] The system is overdetermined as six coefficients must be computed and the vertex P is usually connected to more than six vertices P_i in three dimensions.

If we remember that the metric $\mathcal{M}$ has been defined such that $l_{\mathcal{M}}(\overrightarrow{PX}) = 1$, the desired adapted mesh is then a *unit mesh* in the metric $\mathcal{M}$.

Here, we consider the generation of adapted meshes in three dimensions as a two-step process. At first the surface mesh is adapted using local modifications [24], then an adapted volume mesh is created using a constrained Delaunay algorithm [25]. Notice that, during the point insertion phase of the volume mesh generation, most of the vertices of the previous mesh are reused for cpu concerns. This could reduce interpolation errors but in practice kept vertices are mostly in non-critical area.

2.4.1. Mesh gradation control

As the mesh is intended for finite element or finite volume computations, the mesh size variation is also a major concern. To control the mesh size variation or mesh gradation, a correction of the metric field that governs the mesh generation is performed [26]. The metric field is modified such as the size variation between two neighboring elements prescribed by the metric is bounded by a given threshold value h_{grad}. This process allows the mesh variation to be smoothed and controlled, resulting in an improved quality of the mesh.

2.4.2. Surface mesh adaptation

Given a discrete surface (a piecewise linear approximation of the domain boundaries) and a discrete metric field, the aim is to generate an adapted mesh with respect to this metric and the geometry. To do so, the approach we use consists in modifying iteratively the initial surface mesh so as to complete a unit mesh for both constraint.

As we assume that no CAD modeler is available, an internal G^1 (tangent plane continuity) continuous geometric support is constructed, using a local parametrization at the mesh vertices. Then, a geometric metric tensor $\mathcal{G}$ is defined at the mesh vertices representing the local principal curvatures and directions [24], $\mathcal{G}$ is used to control the geometrical approximation.

Actually, this geometric metric $\mathcal{G}$ has to be intersected with the computational metric $\mathcal{M}$ (Section 2.3) in order to adapt the surface mesh to the solution and the local curvatures of the geometry. In turn, this metric $\mathcal{G} \cap \mathcal{M}$ must be modified to account for the desired mesh gradation. The resulting metric $\tilde{\mathcal{M}}$ is used to govern all surface mesh modifications.

The ingredients to comply with these requirements typically include mesh enrichment, mesh coarsening and local mesh optimization procedures. The local mesh modification operators involved are: edge flipping, edge collapsing, edge splitting, node repositioning and degree relaxation.

The surface mesh modification algorithm is pretty straightforward, edge lengths are computed with respect to the metric $\tilde{\mathcal{M}}$ and edges that are considered small are collapsed while edges that are considered long are split into unit length segments. Edge flips and node repositioning operations are performed to improve the overall size and shape mesh quality [24].

2.4.3. Volume mesh adaptation

Once the surface mesh has been adapted, a unit volume mesh is generated with respect to the modified metric $\tilde{\mathcal{M}}$. In our case, an empty mesh (with no internal vertices) is first built by means of a constrained Delaunay procedure due to the boundary enforcement. Then, based on an edge length analysis, internal nodes are added into the current mesh (most of them coming from the background mesh, at the previous iteration) using the *Delaunay kernel*. This procedure has been thoroughly detailed in Ref. [25].

2.5. Solution interpolation

Solution interpolation or solution transfer is also a key point in the mesh adaptation algorithm. The aim is to recover the solution field after generating a new adapted mesh. As we have a discrete solution field, we need an interpolation scheme to transfer this information from the current mesh to the newly adapted mesh.

During the interpolation stage, two problems have to be taken into account. Firstly, the new vertices are located in the background mesh by identifying the elements containing them. This can be solved by moving inside the oriented mesh by using its topology, thanks to a barycentric coordinates-based algorithm [18]. Secondly, once the localization has been solved, an interpolation scheme is used to extract the information from

the solution field. In our case, as the solution is considered piecewise linear by elements (because the solution is defined only at the mesh vertices) we utilize a classical P^1 interpolation scheme.

3. Mesh adaptation based on a transient fixed point

In this section, before presenting our new mesh adaptation algorithm designed for transient problem, let us carry out an error analysis on the simple linear transport equation in one dimension. Then, we will expose the problematic related to the classical mesh adaptation scheme and present our new approach.

3.1. Controlling the time error

We analyze the error in time obtained with two finite volume schemes with an explicit discretization in time on the transport equation (a first order hyperbolic problem) in one dimension:

$$u_t + cu_x = 0, \tag{7}$$

with $c > 0$. Let x_j for $j = 1, \ldots, N$ be a uniform spatial discretization of the one-dimensional domain and let t_n for $n = 0, \ldots, T$ be a uniform time discretization. For each vertex x_j at time t_n, we have $u_j^n \approx u(x_j, t_n)$.

Remark 3.1. In one dimension and uniform meshes, finite volume scheme are equivalent to finite difference scheme.

3.1.1. A first order scheme

Let us consider the well known explicit first order upwind method:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0. \tag{8}$$

Using Taylor expansion, the local truncation error τ_j^n is given by:

$$\tau_j^n = c\frac{\Delta x}{2}u_{xx}(x_j, t_n) - \frac{\Delta t}{2}u_{tt}(x_j, t_n) + \mathcal{O}(\Delta x^2, \Delta t^2).$$

An estimation of the spatial and the time error is provided by the first term and the second term, respectively (indices are omitted for clarity):

$$\varepsilon(\Delta x) = c\frac{\Delta x}{2}u_{xx} \quad \text{and} \quad \varepsilon(\Delta t) = -\frac{\Delta t}{2}u_{tt}.$$

From Eq. (7), we deduce $u_{tt} = c^2 u_{xx}$ therefore

$$\varepsilon(\Delta t) = -c^2\frac{\Delta t}{2}u_{xx}.$$

Now, we trivially deduce that the time error is bounded by the spatial error under the CFL condition. Indeed, the CFL condition is $v = c\frac{\Delta t}{\Delta x} \leqslant 1$ implies that:

$$|\varepsilon(\Delta t)| \leqslant |c\frac{\Delta x}{2}u_{xx}| = |\varepsilon(\Delta x)|.$$

However, as presented in Section 2.2, we use for our numerical simulation a third-order solver. In the following, the same analysis will be done for this scheme.

3.1.2. Finite volume discretization

For the finite volume approach, a cell C_j is defined as the interval $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ where $x_{j-\frac{1}{2}} = \frac{x_j + x_{j-1}}{2}$ and $x_{j+\frac{1}{2}} = \frac{x_{j+1} + x_j}{2}$.

Let $U = \{u_j\}$ be the unknown vector which has for components the approximation of the function $u(x)$ at each vertex x_j of the mesh. Then, the vector $\Psi(U)$ that is the spatial approximation of $(f(u))_x = c\ u_x$ is constructed, it reads:

$$\Psi_j(U) = \frac{\Phi_{j+\frac{1}{2}} - \Phi_{j-\frac{1}{2}}}{\Delta x}, \tag{9}$$

where Φ is a numerical flux function given by:

$$\Phi_{j-\frac{1}{2}} = \Phi(u_{j-\frac{1}{2}^-}, u_{j-\frac{1}{2}^+}) \quad \text{and} \quad \Phi_{j+\frac{1}{2}} = \Phi(u_{j+\frac{1}{2}^-}, u_{j+\frac{1}{2}^+}),$$

where $u_{j\pm\frac{1}{2}^\pm}$ are the values of u at the boundaries of the cell C_j. Moreover, we impose to Φ to be consistent with the flux $f(u)$: $\forall u$, $\Phi(u,u) = f(u)$. Scheme (9) is called *upwind in the sense of Harten, Lax and Van Leer* [27] if the numerical flux function Φ verifies:

$$\Phi(u,v) = \frac{1}{2}(f(u) + f(v) - d(u,v)),$$

with $d(u,v) = |f'(\frac{u+v}{2})|(v-u) + o(|u-v|)$.

We notice that the expression of Φ is composed with a centered term and the term $d(u,v)$ that contains the numerical internal viscosity of the scheme. To control this viscosity, a parameter δ is introduced, thus the flux function Φ^δ writes:

$$\Phi^\delta(u,v) = \frac{1}{2}(f(u) + f(v) - \delta d(u,v)).$$

In the linear case where $f(u) = cu$ with $c > 0$, $d(u,v)$ is defined by : $d(u,v) = c(u-v)$ and the numerical flux function is reduced to:

$$\Phi^\delta(u,v) = \frac{c}{2}[(1+\delta)u + (1-\delta)v].$$

For $\delta = 1$, we have: $\Phi^1(u,v) = c\,u$. It results:

$$\Psi_j^\delta(U) = \frac{c}{2\Delta x}\left((1+\delta)u_{j+\frac{1}{2}^-} + (1-\delta)u_{j+\frac{1}{2}^+} - (1+\delta)u_{j-\frac{1}{2}^-} - (1-\delta)u_{j-\frac{1}{2}^+}\right). \tag{10}$$

3.1.3. Explicit time discretization

A temporal integration of the same order as the spatial discretization is required as we deal with time dependent simulation. To this end, we propose to use to high-order explicit multi-step Runge–Kutta method that has been proposed in [28]. Such methods are qualified as strong-stability-preserving (SSP) time discretization methods which have a non-linear stability property that makes them particularly suitable for the integration of hyperbolic conservation laws where discontinuous behavior is present.

The optimal order 3 SSP Runge–Kutta scheme with 3-steps is given by:

$$\begin{aligned}
U^{(1)} &= U^n + \Delta t L(U^n),\\
U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}U^{(1)} + \frac{1}{4}\Delta t L(U^{(1)}),\\
U^{n+1} &= \frac{1}{3}U^n + \frac{2}{3}U^{(2)} + \frac{2}{3}\Delta t L(U^{(2)}).
\end{aligned}$$

The discretization of the advection equation (7) leads to a system of ordinary differential equations:

$$U_t + \Psi(U) = 0,$$

which is solved with the previous SSP Runge–Kutta scheme utilizing $L = -\Psi$. In the linear case where $f(u) = cu$, Ψ could be linearly expressed in function of u. Consequently, we introduce a linear operator λ such that: $\Psi = -\lambda u$. Therefore, we deduce for the order 3 SSP Runge–Kutta:

$$\frac{u^{n+1} - u^n}{\Delta t} = \lambda u^n + \lambda^2 \frac{\Delta t}{2} u^n + \lambda^3 \frac{\Delta t^2}{6} u^n. \tag{11}$$

3.1.4. Low diffusion scheme

Now, let us define the interpolation of the values of $u_{j\pm\frac{1}{2}^\pm}$ of $u(x)$ at the boundaries of the volume cells C_j. The values $u_{j\pm\frac{1}{2}^\pm}$ are constructed by means of first order Taylor expansion of $u(x)$, we obtain:

$$u_{j+\frac{1}{2}^-} = u_j + \frac{1}{2}[(1-\beta)(u_{j+1}-u_j) + \beta(u_j - u_{j-1})],$$
$$u_{j+\frac{1}{2}^+} = u_{j+1} - \frac{1}{2}[(1-\beta)(u_{j+1}-u_j) + \beta(u_{j+2} - u_{j+1})],$$
$$u_{j-\frac{1}{2}^-} = u_{j-1} + \frac{1}{2}[(1-\beta)(u_j - u_{j-1}) + \beta(u_{j-1} - u_{j-2})],$$
$$u_{j-\frac{1}{2}^+} = u_j - \frac{1}{2}[(1-\beta)(u_j - u_{j-1}) + \beta(u_{j+1} - u_j)].$$

Substituting these values in Eq. (10), we get:

$$\Psi_j^\delta(U) = \frac{c}{4\Delta x}[(1+\delta)\beta u_{j-2} - 2(1+\beta+2\delta\beta)u_{j-1} + 6\delta\beta u_j + 2(1+\beta-2\delta\beta)u_{j+1} - (1-\delta)\beta u_{j+2}]. \tag{12}$$

Now, we consider Taylor expansion of the function u about x_j, from the previous relation we have:

$$\Psi^\delta(U) = c\left(u_x + (1-3\beta)\frac{\Delta x^2}{6}u_{xxx} + \delta\beta\frac{\Delta x^3}{4}u_{xxxx}\right) + \mathcal{O}(\Delta x^4), = -\lambda u$$

Therefore, the linear operator λ is defined by:

$$\lambda = -c\left(\frac{\partial}{\partial x} + (1-3\beta)\frac{\Delta x^2}{6}\frac{\partial^3}{\partial x^3} + \delta\beta\frac{\Delta x^3}{4}\frac{\partial^4}{\partial x^4}\right) + \mathcal{O}(\Delta x^4), \tag{13}$$

and thus,

$$\lambda^2 = c^2\left(\frac{\partial^2}{\partial x^2} + (1-3\beta)\frac{\Delta x^2}{3}\frac{\partial^4}{\partial x^4}\right) + \mathcal{O}(\Delta x^3), \tag{14}$$

$$\lambda^3 = -c^3\frac{\partial^3}{\partial x^3} + \mathcal{O}(\Delta x^2). \tag{15}$$

The expression of λ determines the spatial approximation of the convective flux $(f(u))_x$.

3.1.5. A third order scheme

From the previous analysis, by fixing the parameters δ and β to 1 and 1/3, respectively, we get an upwind third-order scheme in space and in time. Indeed, by using Taylor expansion of u^{n+1} around t_n in the left-hand side term of Relation (11) and by substituting the values of λ in the right-hand side term with equalities (13)–(15), we obtain:

$$u_t + cu_x = c^2\frac{\Delta t}{2}u_{xx} - c^3\frac{\Delta x^2}{6}u_{xxx} - c\frac{\Delta x^3}{12}u_{xxxx} - \frac{\Delta t}{2}u_{tt} - \frac{\Delta t^2}{6}u_{ttt} - \frac{\Delta t^3}{24}u_{tttt} + \mathcal{O}(\Delta t^4, \Delta x^4).$$

Using the advection equation, the previous relation is simplified to:

$$u_t + cu_x = -c\frac{\Delta x^3}{12}u_{xxxx} - \frac{\Delta t^3}{24}u_{tttt} + \mathcal{O}(\Delta t^4, \Delta x^4).$$

An estimation of the spatial and the time error is provided by the first term and the second term of the right-hand side, respectively (indices are omitted for clarity):

$$\varepsilon(\Delta x) = -c\frac{\Delta x^3}{12}u_{xxxx} \quad \text{and} \quad \varepsilon(\Delta t) = -\frac{\Delta t^3}{24}u_{tttt}.$$

From Eq. (7), we deduce $u_{tttt} = c^4 u_{xxxx}$ therefore

$$\varepsilon(\Delta t) = -c^4\frac{\Delta t^3}{24}u_{xxxx}.$$

Finally, we deduce that the time error is bounded by the spatial error under the CFL condition. Indeed, the CFL condition is $v = c\frac{\Delta t}{\Delta x} \leqslant 1$ implies that:

$$|\varepsilon(\Delta t)| \leqslant |c\frac{\Delta x^3}{24}u_{xxxx}| \leqslant |\varepsilon(\Delta x)|.$$

In conclusion, this simple mono-dimensional study on the advection equation shows that the time error is controlled by the spatial error under the CFL condition. Therefore, if the spatial error is bounded by a threshold value at each time step throughout a mesh adaptation process, then the time error is also bounded by this threshold.

To analyze mesh adaptation schemes, let us introduce the notion of characteristic time frame for a mesh.

Definition 3.1. *The characteristic time frame* τ of a mesh can be defined as the largest period of time throughout which the spatial error remains bounded (or equally distributed) by a given value.

Notice that τ can be expressed as a function of the desired interpolation error ε, the mesh gradation h_{grad}, the minimal mesh size $h_{\min}$ and $\vec{U}(t)$ the speed of physical phenomenon at time $t : \tau = f(\varepsilon, h_{\text{grad}}, h_{\min}, \vec{U}(t))$.

3.2. Classical mesh adaptation algorithm

The classical mesh adaptation algorithm (cf. Section 2.1) consists in finding a fixed point for the mesh–solution couple. It seems obvious that this algorithm is intrinsically inappropriate to study time-dependent phenomena because of their arbitrary progression in the computational domain. Indeed, in such approach the mesh is always "left behind" with respect to the solution. In other words, this algorithm introduces small perturbations in the resolution of an unsteady problem because of the shift between the solution and the metric. This implies that we are not guaranteeing to control the spatial error. Consequently, the accuracy of the numerical solution is closely related to the matching between the characteristic time frame τ, and the frequency the mesh is adapted. Another problem is that the adequate number of adaptations is not known a priori, it is problem-dependent. Therefore, an arbitrary choice is made by the user.

More precisely, if an insufficient number of mesh adaptations is performed, then at each mesh adaptation iteration the solution time increment is Δt which is greater than τ. Then, the spatial error is no more controlled. Practically, the solution is diffused.

On the other hand, if $\Delta t = \mathcal{O}(\tau)$ (in general, 5–20 time steps of the flow solver are sufficient), we could expect the solution to be more accurate. Nevertheless in this case, relatively to each mesh modification, a source of error due to the interpolation (or transfer) of the solution from the old mesh on the new one is introduced. Therefore, for "long time" computation this affects the solution's accuracy.

These two cases will be discussed in Section 4.

3.3. Unsteady mesh adaptation algorithm

In time-dependent simulations, we are faced with the problem of the random progression of physical phenomena in the domain. From the error analysis in Section 3.1, we saw that for explicit algorithm under CFL condition the error in space control the error in time, consequently to control the solution accuracy we have to control the error in space throughout a given time frame of computation. Moreover, we have to propose an automatic approach complying with the problem, regardless of the number of desired adaptations. In other words, in this approach the choice of the number of mesh adaptations does not depend on the considered simulation and has no or just a slight impact on the accuracy of the numerical solution. Finally, the number of mesh adaptations has to be reduced to minimize the errors due to the solution interpolation (or transfer) stage. To this end, we propose a new transient fixed point mesh adaptation algorithm.

This transient fixed point mesh adaptation algorithm combines the classical mesh adaptation scheme and the fixed point coupling scheme of [17], that allows a notable gain of efficiency to be obtained. Therefore, it is a generalization of the classical mesh adaptation method. In particular, this improved algorithm is able to deals with steady and unsteady simulations.

This scheme predicts the solution evolution in the computational domain and suitably meshes all the regions of the solution's progression by means of an iterative implicit algorithm. This answer to the physical phenomena prediction allows us to control the spatial error during the resolution. More formally, the number

of adaptations has to be reduced while the solution accuracy must be preserved, in an attempt to avoid refining an excessively large part of the domain. In other words, the aim is to control and increase the characteristic time frame of the mesh (cf. Definition 3.1). This leads into the introduction of the time in the metric construction.

3.3.1. Unsteady adaptation scheme

In order to predict the progression of physical phenomena, a new mesh adaptation algorithm is proposed in which the principle is to solve a transient fixed point problem for the mesh–solution couple at each iteration of the mesh adaptation loop. This iterative algorithm consists of two steps: the main adaptation loop and an internal loop in which the transient fixed point problem is solved. At each iteration of the main adaptation loop, we consider a time period $[t, t + \Delta t]$ in which the solution evolves. Throughout this period, we try to algorithmically converge to the solution at $t + \Delta t$ and (in a certain way) to the associated adapted mesh. In other words, from the solution at time t, we compute the solution to time $t + \Delta t$, and the computation is iterated via the internal loop until the desired accuracy is obtained for the solution at $t + \Delta t$, i.e., two consecutive solutions of the internal loop are close to each other (see Relation (16)). Similarly, we algorithmically converge towards the corresponding invariant mesh adapted to this period $[t, t + \Delta t]$ throughout a sequence of consecutively adapted meshes. Hence, the solution behavior is predicted in all the regions of the domain where the solution evolves. Therefore, a metric which takes into account the solution progression could be defined by means of an intersection procedure in time (cf. Section 3.3.2). Then, a new mesh is generated according to this metric field. Finally, the initial solution of this period is interpolated and the computation is resumed. This scheme is illustrated in Fig. 3, where i and j indicate the main and the internal loop index, respectively, and where $\mathscr{H}$, $\mathscr{S}$, $\mathscr{S}^0$ and $\mathscr{M}$ are the mesh, the solution, the initial solution at each iteration and the metric, respectively.

Notice that this novel algorithm is obtained by combining the classical mesh adaptation and the transient fixed point scheme of [17]. Indeed, if you choose 0 internal iteration then no mesh adaptation is performed with the algorithm of [17], whereas with the novel scheme you obtain the classical mesh adaptation algorithm. Thus, it can be applied to steady and unsteady simulations. In summary, this novel algorithm can be perceived as a generalization of the classical approach. And more technically, between two periods of time, the adapted mesh used with the new scheme to make the first computation has a reduced size as it is adapted only to the solution at time t implying a gain in efficiency.

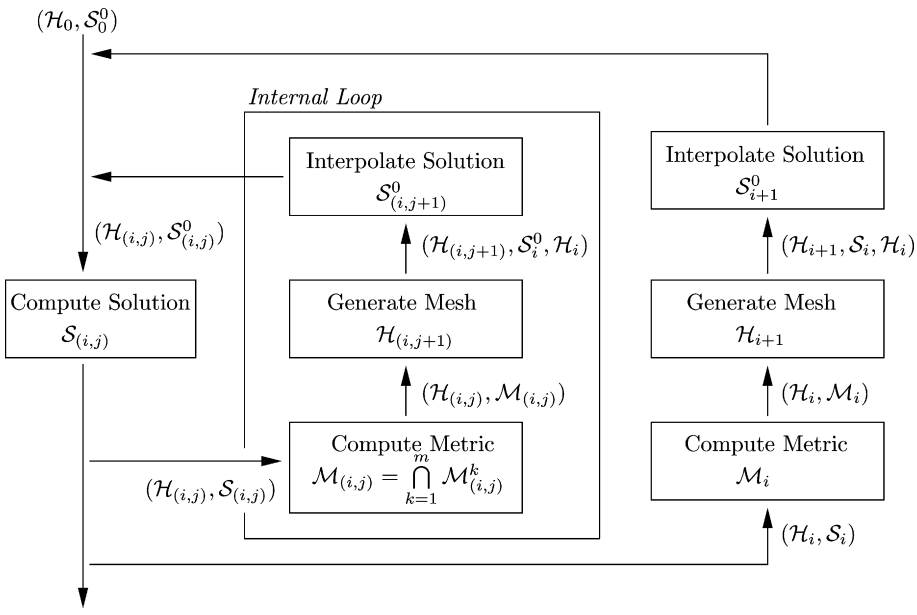

Fig. 3. Modified mesh adaptation scheme for time-dependent simulations.

At each internal iteration, the final solution of the period (i.e., at $t + \Delta t$) of the previous and the current iteration, denoted $\mathscr{S}_{(i,j)}$ and $\mathscr{S}_{(i,j+1)}$ respectively, are compared to analyze the algorithmic convergence of the solution in the internal loop. To this end, we consider the L^1 norm of the difference of the consecutive solutions for CFD simulation. Indeed, the L^1 norm gives the mean behavior of the solution without focussing too much on the discontinuities of the flow. Moreover, we know from the TVD theory that the good space for the convergence of numerical schemes for scalar hyperbolic conservation laws is L^1_{loc}, thus the space L^1 is adequate.

If the difference is lesser than the given threshold value, the algorithmic convergence is considered reached, the external main adaptation loop is resumed. Otherwise the computation is restarted. Let ϵ be the given threshold value. The transient fixed point problem is solved until:

$$\frac{\|\mathscr{S}_{(i,j+1)} - \mathscr{S}_{(i,j)}\|_{L^1(\Omega)}}{\|\mathscr{S}_{(i,j+1)}\|_{L^1(\Omega)}} \leqslant \epsilon, \tag{16}$$

where Ω is the computational domain. The difference between two fields is computed by transferring (interpolating) fields on the same mesh, i.e., here the difference is computed on $\mathscr{H}_{(i,j)}$ and on $\mathscr{H}_{(i,j+1)}$, then the mean of the two obtained values is utilized.

3.3.2. Metric intersection in time

Although the new mesh adaptation scheme is able to predict the solution behavior, we still have to specify the procedure to suitably mesh all the regions where the solution evolves. To this end, we introduce a metric intersection procedure in time.

As a physical phenomenon progresses in a well-defined region during a given time, a mesh with appropriate element sizes is needed in order to achieve an accurate solution in such region. The metric field must reflect this information, i.e., the time must be considered in the metric construction. For these purposes, all the intermediate solution metrics, throughout the time period $[t, t + \Delta t]$, have to be taken into account to mesh suitably all this region so as to control the error of the solution throughout this time period. Indeed, it is well known that a contact discontinuity is not compressive like a shock wave. Therefore, if a contact discontinuity is getting thicker on a coarser mesh area then it will be impossible to retrieve a finer one for the rest of the computation with a MUSCL type scheme.

In order to solve this problem, for each variable used in the metric construction, an intersection of the relative metrics (related to the successive solutions in time) is introduced. Formally speaking, the metric tensor, at the ith time period $[t, t + \Delta t]$ (corresponding to the ith main adaptation iteration) and at the jth internal loop iteration, is given by:

$$\mathscr{M}_{(i,j)} = \bigcap_{\theta \in [t, t+\Delta t]} \mathscr{M}_{(i,j)}(\theta), \tag{17}$$

where $\cap$ is the metric intersection operation defined in Section 2.3, $\mathscr{M}_{(i,j)}(\theta)$ is the intermediate metric given by the numerical solution at time $\theta \in [t, t + \Delta t]$ and $\mathscr{M}_{(i,j)}$ is the resulting metric utilized to adapt the mesh. Practically, it is sufficient to consider only a certain number of metrics provided by intermediate solutions, uniformly distributed between the initial and the final solutions of the time period, to mesh the regions at hand. Therefore, the metric tensor, at the ith time period and at the jth internal loop iteration, reads:

$$\mathscr{M}_{(i,j)} = \bigcap_{k=1}^{m} \mathscr{M}^k_{(i,j)},$$

where $\mathscr{M}^k_{(i,j)}$ is the kth intermediate metric given by the numerical solution.

Remark 3.2. This metric definition ensures that for all the solutions throughout the time period the interpolation error for each element remains bounded by the prescribed threshold value ε, until $h_{\min}$ is not reached. Nevertheless, for each solution the interpolation error is no longer equally distributed. The adapted meshes are not "optimal" for one solution at a given time, but we can consider that they are "pseudo-optimal" for the computation along this period.

4. Application to the model problem

In this section, we will apply and study the behavior of the classical mesh adaptation and the transient fixed point mesh adaptation to a model problem of non-linear waves propagation in a complex geometry. To this end, we will describe the computation of the reference solution for the model problem and define how to compute the gap to this solution. Then, we will analyze the results obtained with the classical approach and emphasize the problematic related to this approach (cf. Section 3.2). Secondly, the new mesh adaptation scheme will be applied and analyzed on this problem. Finally, the impact of anisotropic mesh adaptation will be discussed.

4.1. Non-linear waves propagation as a model problem

As a model problem of unsteady simulations, we consider here the two-dimensional simulation of non-linear shock waves propagation in a complex geometry (a city plaza) modeled by the Euler equations for compressible flows, Section 2.2. This simulation can be perceived as a multi-dimensional generalization of the Sod Riemann problem [29] in a geometry. We deliberately choose, as model problem, the example already proposed in [17] to point out the limitations of the classical mesh adaptation scheme and the improvements of the new approach. More precisely, an initial Heavyside perturbation (high pressure and density region) is introduced in a uniform field (the ambient air). The main feature of such problem is related to the random character of the propagation and thus to the difficulty of predicting the phenomenon behavior and the shocks' interactions.

The computational domain size is $150 \times 90\ \mathrm{m}^2$ and the relevant parameters are $\rho = 0.125$, $p = 0.1$ and $u = v = 0$, ρ, p and $\vec{U} = (u, v)$ representing the density, the pressure and the velocity vector, respectively. In the high pressure region (delimited by the square $[65, 67.5] \times [0, 2.5]$), the initial conditions are: $\rho = 1$, $p = 1$ and $u = v = 0$.

To analyze the mesh adaptation schemes, we will compare the obtained adapted solutions with a reference solution. As this model problem has neither simple nor known analytical solution, the reference solution is computed on a highly refined constant size mesh. To this end, the relative error between two solutions u and v is given by the following formula:

$$f_{\mathrm{err}}(u, v) = \frac{2\|u - v\|_{\Omega}}{\|u\|_{\Omega} + \|v\|_{\Omega}}, \tag{18}$$

where $\|\cdot\|_{\Omega}$ is a suitable norm (here the L^1 norm) in the domain Ω.

Definition 4.1. An unstructured uniform mesh of size h (i.e., all edges having a constant size h) is a *reference* mesh if $f_{\mathrm{err}}(u_{h/2}, u_h) \leqslant \varepsilon$, where $u_{h/2}$ and u_h denote the solutions obtained on the mesh of size $h/2$ and h, respectively, and $\varepsilon \ll 1$ is a constant.

In other words, on the reference mesh the solution is mesh independent. The numerical solution obtained on a such mesh is called *reference solution* to tolerance ε.

4.1.1. Computing the reference solution

For a mesh of constant size h equal to 30 cm, a relative error $f_{\mathrm{err}}(u_{h/2}, u_h) \leqslant 0.0012$ in $L^1(\Omega)$ norm is obtained. Therefore, we can consider this solution as a reference solution. Nevertheless, for practical reasons, we consider here, as reference mesh, the mesh of constant size h_{ref} equal to 15 cm (1,412,782 vertices). The error obtained with several uniform meshes are reported in Table 1.

Notice that the convergence is not evident in L^2 norm. This is illustrated by the fact that the error in L^2 norm committed on the mesh with a size of $2h_{\mathrm{ref}}$ is lower than the one with a size of $4/3h_{\mathrm{ref}}$. This is due to the higher weight of the singularities in the L^2 norm which are mesh dependent in the region where the explosion start.

From the above results, we deduce that the physical phenomena progression will be correctly predicted and accurately captured if the numerical solution u is "close to" the reference solution u_{ref} in the sense that it verifies:

$$\frac{\|u_{\text{ref}} - u\|_{L^1(\Omega)}}{\|u_{\text{ref}}\|_{L^1(\Omega)}} \leqslant 0.0012 \quad \text{and} \quad \frac{\|u_{\text{ref}} - u\|_{L^2(\Omega)}}{\|u_{\text{ref}}\|_{L^2(\Omega)}} \leqslant 0.008. \tag{19}$$

4.2. Numerical study

The goal is to compute the solution at the physical time $t_{\max} = 0.2$ s for the model problem. In each adaptive computation, the density variable has been chosen as criterion for adapting the mesh with the following parameters:

$$\varepsilon = 0.001, \quad h_{\min} = 0.15 \text{ m}, \quad h_{\max} = 20 \text{ m} \quad \text{and} \quad h_{\text{grad}} = 4,$$

where ε is the prescribed error threshold, $h_{\min}$ and $h_{\max}$ are the minimal and the maximal element size, respectively, and h_{grad} is the mesh gradation. The initial mesh, adapted to the initial conditions, contains 1317 vertices and 2467 triangles. For each simulation, tables report:

- the number n_{adap} of mesh adaptations,
- the number of vertices *np* and the number of triangles *ne* of the final mesh,
- the overall cpu time *t cpu*, each simulation has been carried out in serial on a Macintosh G5 with 2.5 GHz PowerPC processor and 1 GB of memory,
- the error in L^1 norm and L^2 norm with respect to the reference solution for the density.

The aim of this section is to illustrate the limitation of the classical mesh adaptation approach and to validate the new mesh adaptation approach to the model problem of non-linear waves propagation.

4.2.1. Limitation of the classical mesh adaptation

The classical mesh adaptation algorithm has been applied to the model problem with 20, 40, 60, 100, 200, 300 and 400 mesh adaptations. The final meshes statistics and the error relative to the final solutions (density) on Ω are reported in Table 2. The results analysis classified them in three categories: the time increment Δt between two adaptations is larger than or almost or lower than the mesh characteristic time frame τ. In the following, each case is analyzed in detail.

Firstly, let us analyze the case where 20 mesh adaptations are performed corresponding to a time increment $\Delta t = 0.01$ s between two adaptations. The final adapted mesh obtained has a size two orders of magnitude

Table 1
Statistics related to the uniform meshes of sizes $8/3h_{\text{ref}}$, $4/3h_{\text{ref}}$ and $2h_{\text{ref}}$ and final solution (density) error on Ω

Mesh	Vertices	Triangles	$\|\cdot\|_{L^1(\Omega)}$	$\|\cdot\|_{L^2(\Omega)}$
$8/3h_{\text{ref}}$	192,993	383,931	0.017	0.0096
$2h_{\text{ref}}$	353,799	704,719	0.0011	0.0072
$4/3h_{\text{ref}}$	767,851	1,531,612	0.008	0.0077

Table 2
Statistics of the final adapted meshes for the classical approach with 20, 40, 60, 100, 200, 300 and 400 mesh adaptations and the errors relative to the final solutions (density) on Ω

n_{adap}	*np*	*ne*	*t cpu*	$\|\cdot\|_{L^1(\Omega)}$	$\|\cdot\|_{L^2(\Omega)}$
20	11,801	23,155	2 min	0.0037	0.0101
40	15,853	31,211	3 min 30 s	0.0029	0.0091
60	18,561	36,651	5 min	0.0027	0.0088
100	21,621	42,769	8 min	0.0026	0.0085
200	23,266	46,079	13 min	0.0026	0.0080
300	22,259	44,069	18 min	0.0031	0.0095
400	21,148	41,855	19 min	0.0032	0.0099
$\mathscr{H}_{\text{ref}}$	1,412,782	2,819,770	5 h	–	–

smaller than the reference mesh, and consequently so does the cpu time. Nevertheless, the solution is not fully satisfactory. Indeed the L^1 error is three times higher than the previously error given threshold value, Relation (19). This drawback is mainly related to the insufficient number of mesh adaptations. Fig. 5 shows the evolution of the reference solution and the solution after 20 mesh adaptations. We can observe that the adapted solution is diffused because the mesh is left behind the solution. This problem is emphasized in Fig. 6, where the solution (density) profile is given along two arbitrary lines. Notice that the mesh resolution decreases as the number of adaptations increases (Fig. 4).

Secondly, we consider the case where the mesh is frequently adapted in order to avoid the problem of the latency of the mesh with respect to the solution: 200 adaptations are performed corresponding to a time increment $\Delta t = 0.001$ s (approximatively every 5 flow solver time steps) between two mesh adaptations. The final adapted mesh size is about 60 times smaller than the reference mesh and the computational time has been divided by 25. Here, as the shock waves are confined in the refined regions of the adapted mesh, the number of adaptations can be considered as sufficient. Despite the fact that the time increment Δt is almost or smaller than the mesh characteristic time frame τ, the solution obtained is not fully satisfactory. Indeed, the L^1 error is 2 times higher than the given error threshold value. Fig. 5 shows the evolution of the reference solution and the

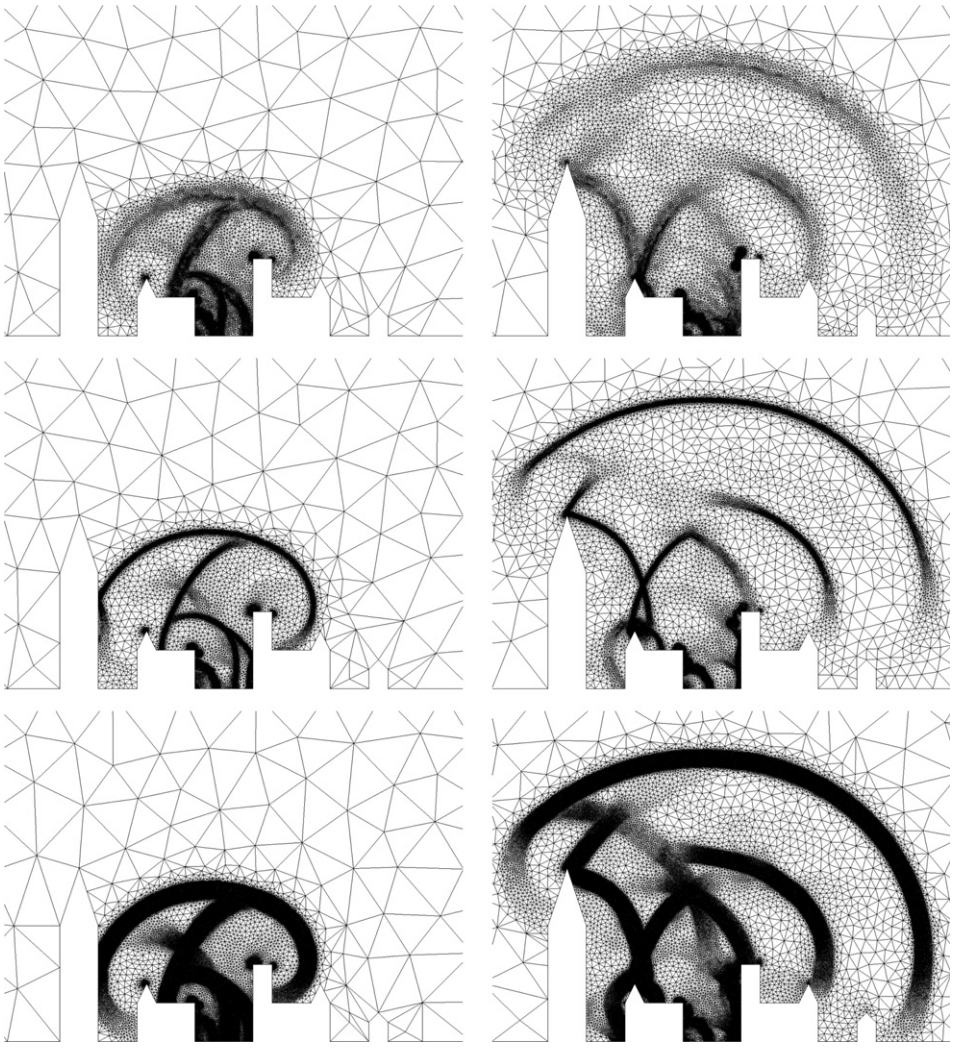

Fig. 4. The progression of adapted meshes at different time 0.1 and 0.2 s. From top to bottom, the adapted meshes with 20 classical mesh adaptations, 200 classical mesh adaptations and 20 transient fixed point mesh adaptations.

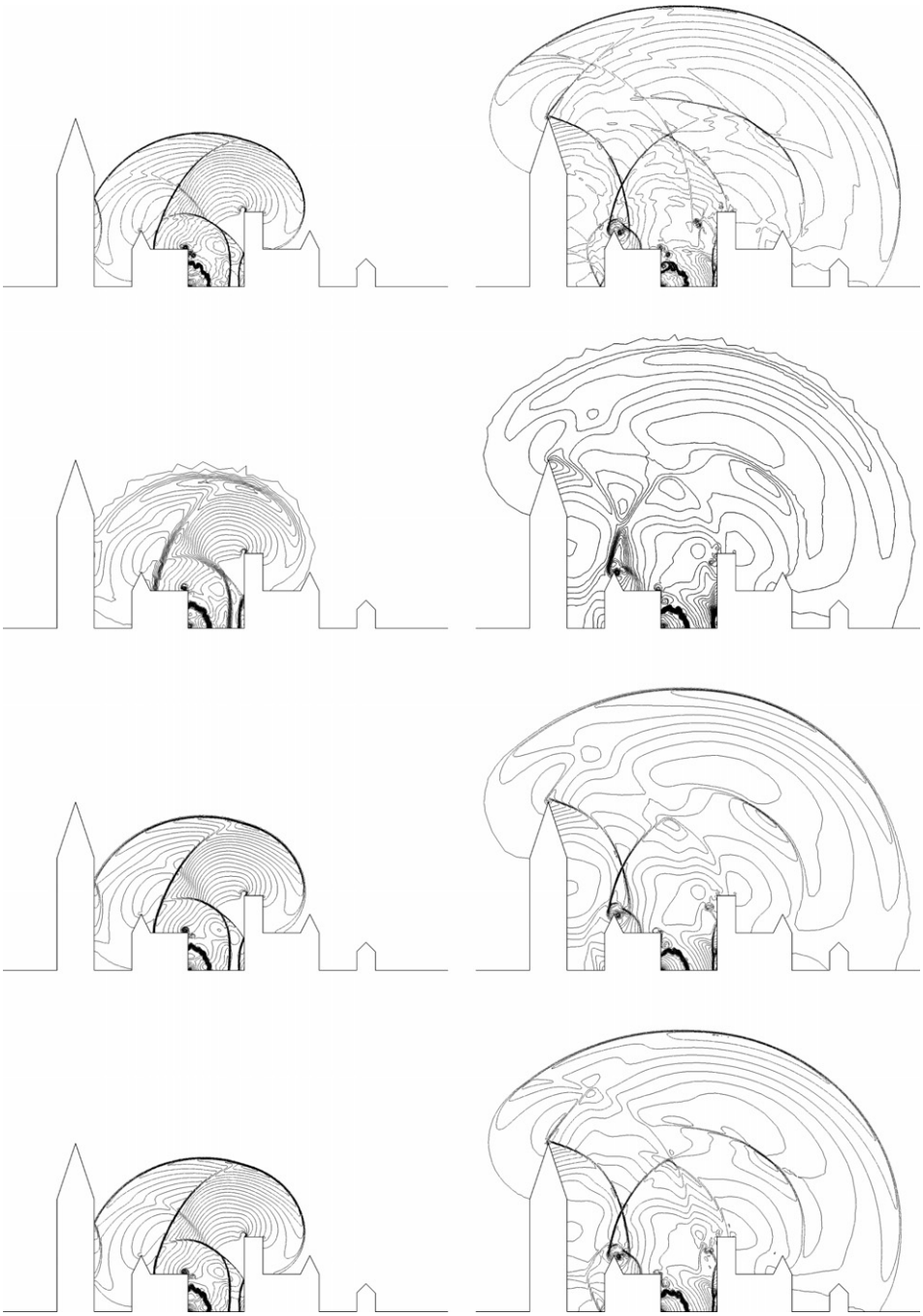

Fig. 5. The progression of density iso-lines at different time 0.1 and 0.2 s. From top to bottom, the solutions with the reference mesh, 20 classical mesh adaptations, 200 classical mesh adaptations and 20 transient fixed point mesh adaptations.

solution after 200 mesh adaptations. The amplitude of strong phenomena is preserved, such as shock waves, but weak phenomena have been lost. Notice that the adapted meshes resolution is preserved along the adaptations (cf. Fig. 4).

Finally, a very large number of adaptations is performed: 300 and 400 adaptations are done corresponding to approximatively every 3 and 2.5 flow solver time steps between two mesh adaptations, respectively. Now,

we notice that in this case the L^1 error increases with the number of mesh adaptations, Table 2. This error is due to the large number of solution interpolations (or transfers) in the adaptation scheme. Indeed, mesh solution interpolation is an important source of error especially if the mesh connectivity is dramatically changed from one mesh to the other that diffuses the solution. Such error becomes dominant as the number of interpolations increases, thus resulting in an increase of the error. Moreover, these simulations show that the cpu time increases with the number of adaptations confirming that a very large number of adaptations penalizes the global cpu time of the simulation.

As expected, the previous cases study allow us to conclude that the classical mesh adaptation scheme is not really suitable for time-dependent simulations for hyperbolic systems of conservation laws. Indeed, the accuracy of the numerical solution cannot be explicitly controlled, whatever the adopted strategy.

4.2.2. Transient fixed point mesh adaptation

The new adaptive approach has been applied to the model problem with 10, 20 and 30 mesh adaptations, i.e., the simulation's time frame is decomposed into 10, 20 and 30 periods, corresponding to a period step Δt equal to 0.02, 0.01 and 0.0066 s, respectively. 13, 7 and 5 uniformly distributed intermediate solutions (including the initial and the final one) are considered for the metric intersection in time, respectively. In the internal loop, a maximum of five iterations are performed and the threshold value is set to $\epsilon = 0.0002$. The final meshes statistics and the error relative to the final solutions (density) on Ω are reported in Table 3. In the following, we discuss in detail the results of the simulation with 20 adaptations.

With 20 mesh adaptations, the final adapted mesh contains 87,471 vertices that is 16 times less than the reference mesh size. Consequently, the cpu time has been divided by a factor of 9. From a practical point of view, in the internal loop, three iterations are sufficient to achieve the solution's algorithmic convergence in the period, i.e., the desired accuracy of the solution at $t + \Delta t$. Table 3 reports the L^1 and L^2 errors that are almost the prescribed error threshold values, Relation (19). Therefore, the resulting solution is close to the reference solution. The comparison between the reference solution and the adapted solution is presented in Fig. 5. The intensity of the solution has been preserved and we have captured more details in the flow than

Table 3
Statistics of the final adapted meshes for the unsteady approach with 10, 20 and 30 mesh adaptations and the errors relative to the final solutions (density) on Ω

n_{adap}	np	ne	t cpu	$\|\cdot\|_{L^1(\Omega)}$	$\|\cdot\|_{L^2(\Omega)}$
10	140,364	279,897	49 min	0.0011	0.0071
20	87,134	173,612	26 min	0.0012	0.0075
30	69,670	138,748	26 min	0.0014	0.0070
$\mathscr{H}_{ref}$	1,412,782	2,819,770	5 h	–	–

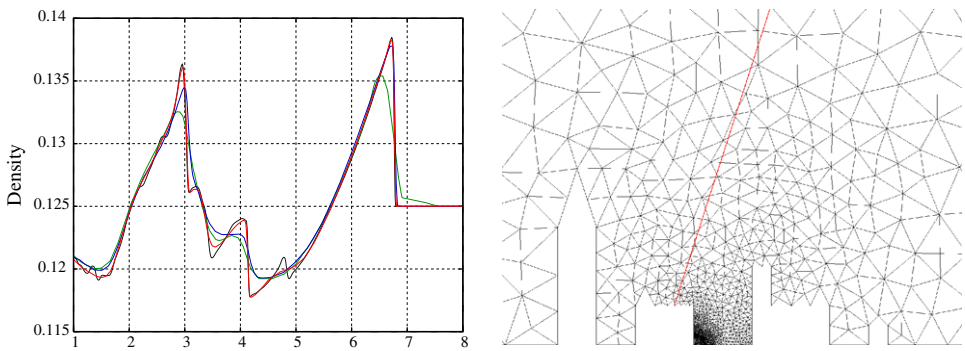

Fig. 6. Left, density values of the solution at time $t_{max} = 0.2$ s along the lines of equation $y = 3.07692x - 17.4615$. The reference solution (black) and the adapted solution with 20 classical mesh adaptations (green), 200 classical mesh adaptations (blue) and 20 transient fixed point mesh adaptations (red) are compared. Right, the initial mesh with the extraction line (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

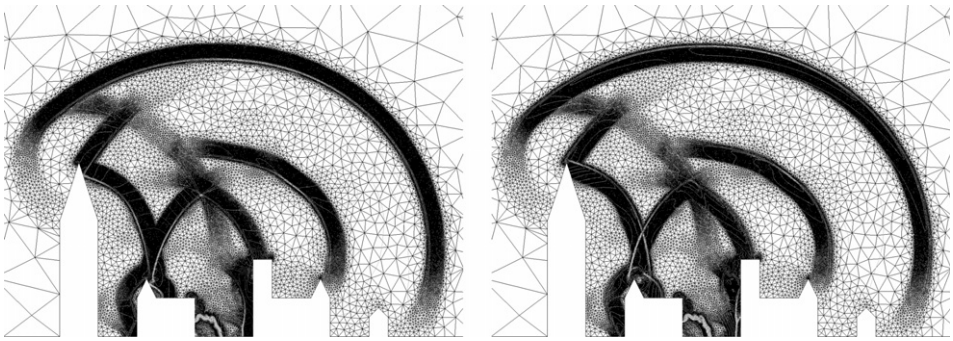

Fig. 7. Final adapted mesh for the unsteady approach with 20 mesh adaptations and the corresponding solutions (density). Left, the initial solution of the last period at time $t = 0.1875$ s and right, the final solution of the last period at time $t_{\max} = 0.2$ s.

the simulations with classical mesh adaptation. This is emphasized in Fig. 6, where the solution (density) profile is plotted along arbitrary lines.

The impact of the metric intersection in time is shown in Fig. 7 for the initial (left) and the final (right) solutions at time $t = 0.1875$ and $t_{\max} = 0.2$ s of the last period, respectively. Notice that all the regions where the phenomena progress have been uniformly refined to preserve the accuracy of the solution and that the phenomena remain confined in these refined regions, in other words the spatial error has been controlled along the period. Consequently, with this approach the characteristic time frame of the mesh is automatically set to the order of the period step: $\tau \approx \Delta t$.

The results obtained with 10 and 30 adaptations bring us to the same conclusion. As expected, solutions with less interpolation steps are slightly more accurate since a smaller error due to the projection stage has been introduced. Nevertheless, a larger mesh size is obtained as the mesh characteristic time frame has been increased that may impact the overall cpu time. The increase of cpu time for the simulation with 10 adaptations is mainly due to the algorithmic convergence of the solution inside the internal loop. In this case, four iterations are required to achieve the solution's algorithmic convergence for 50% of the adaptations.

To conclude with this new approach the space and time error have been controlled whatever the number of chosen mesh adaptation. Moreover, reducing the number of interpolations (here only 10–30) allowed us to control the dissipation introduced by the projection stage. In this approach, the time has been implicitly introduced in the error estimate (by means of metric intersection in time) by controlling the mesh characteristic time frame.

4.2.3. Impact of anisotropic mesh adaptation

In the following, the impact of the anisotropic mesh adaptation is illustrated on the model problem. For anisotropic simulations, we modify two adaptation parameters, ε and γ are set to 0.07 and 0.004, respectively, to have the same complexity like in the isotropic case by utilizing the CFD error estimate provided by Relation (3). The other parameters remain unchanged.

The three previous transient fixed mesh adaptation simulations and the classical mesh adaptation with 200 adaptations have been carried out in the anisotropic case. The final meshes statistics and the error relative to the final solutions (density) on Ω are reported in Table 4. This results led us to the same conclusion for the transient fixed point mesh adaptation. The errors are almost the same for the L^1 norm and lower for the L^2 norm. Compared to the isotropic case, the final adapted meshes size has been reduced by a factor close to 3, consequently the cpu time has been diminished.

For the classical mesh adaptation scheme, even with 200 adaptations we were not able to follow the phenomenon progression implying a larger error that in the isotropic case. Indeed, the mesh size grows faster in the anisotropic case thus reducing the mesh characteristic time frame.

To conclude, with the anisotropic unsteady mesh adaptation, we have obtained a solution close to the reference solution, in the sense given by Relation (19). Notice that for the case with 20 adaptations a reduction of the mesh size by a factor of 46 and the run time by 17 have been obtained.

Fig. 9 shows the progression of the adapted solution (top) and mesh (bottom) with 20 anisotropic mesh adaptations based on a transient fixed point. We notice that the physical phenomena intensity has been pre-

Table 4
Statistics of the final anisotropic adapted meshes for the unsteady approach with 10, 20 and 30 adaptations and the errors relative to the final solutions (density) on Ω

n_{adap}	np	ne	t cpu	$\|\cdot\|_{L^1(\Omega)}$	$\|\cdot\|_{L^2(\Omega)}$
10	52,435	104,083	26 min	0.0011	0.0063
20	31,419	62,178	18 min	0.0013	0.0057
30	25,343	50,079	18 min	0.0015	0.0063
200	9,159	17,894	13 min	0.0037	0.0097
$\mathscr{H}_{ref}$	1,412,782	2,819,770	5 h	–	–

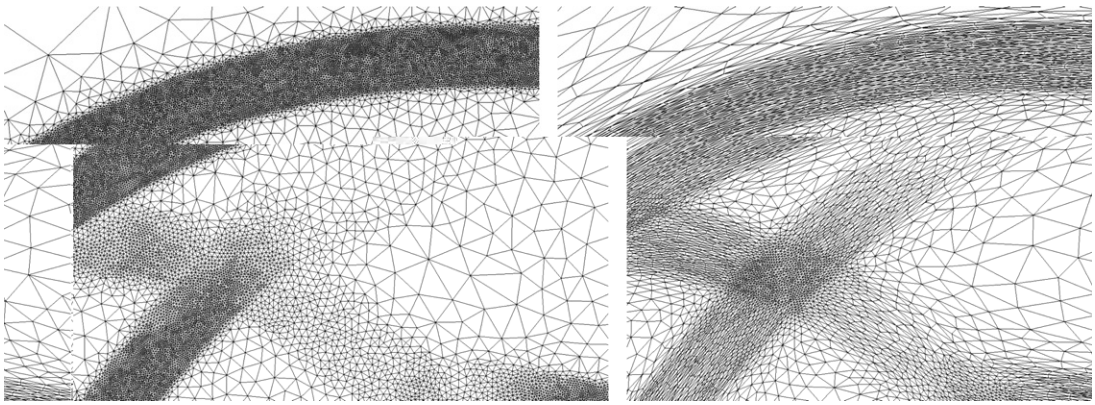

Fig. 8. The impact of the anisotropy is illustrated on a zoom on the final anisotropic (left) and isotropic (right) adapted meshes with 20 unsteady mesh adaptation.

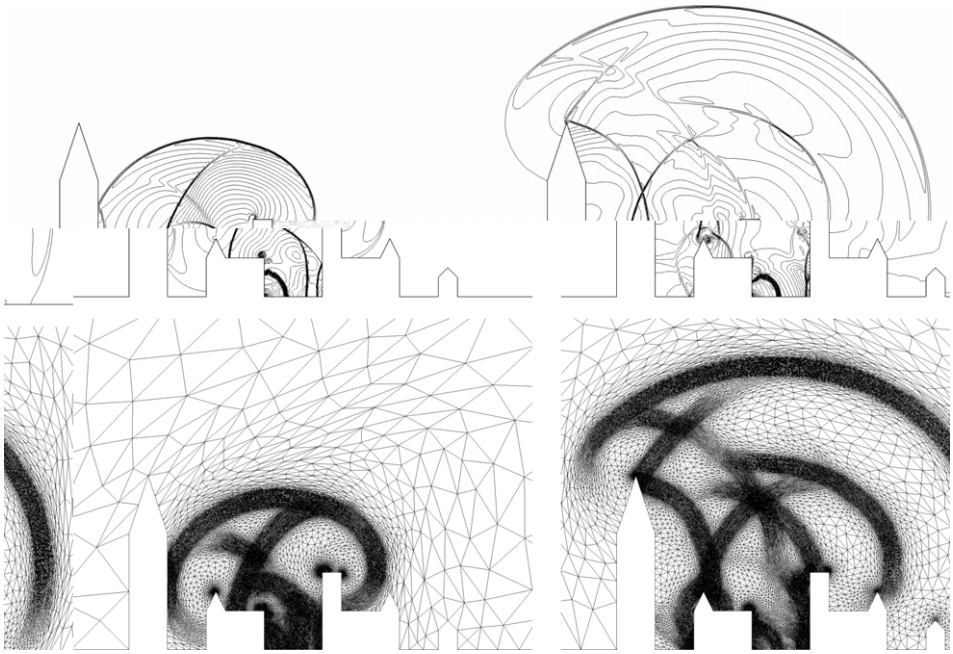

Fig. 9. The progression of density iso-lines for the adapted solution and the associated adapted mesh at different time 0.1 and 0.2 s, for the unsteady anisotropic mesh adaptation with 20 adaptations.

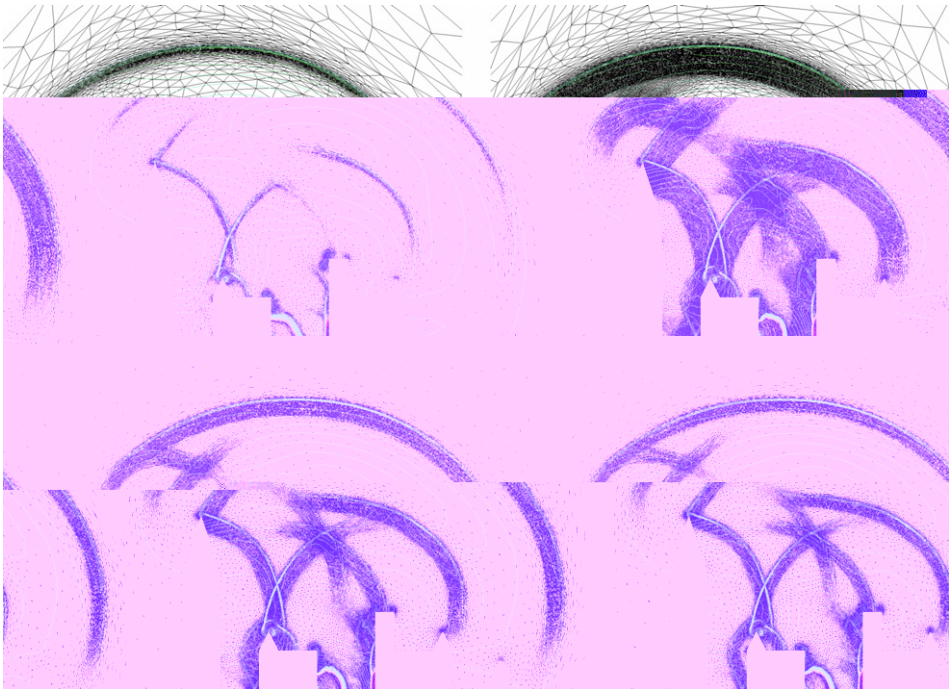

Fig. 10. Final anisotropic adapted meshes and the corresponding solutions (density) at time $t_{\max} = 0.2$ s for the classical mesh adaptation with 200 adaptations (top left) and the unsteady approach with 10 (top right), 20 (bottom left) and 30 (bottom right) mesh adaptations.

served and small details have been captured in the flow as in the isotropic case. Fig. 10 represents the final anisotropic adapted meshes with the corresponding solutions at time $t_{\max} = 0.2$ s for all simulations. This figure points out the control of the mesh characteristic time frame with the new approach, which is of the order of the period step: $\tau \approx \Delta t$ for each computation. Moreover, we observe that the mesh anisotropy in the adapted area has been preserved by the metric intersection in time procedure (cf. Fig. 8).

4.3. Conclusion

The transient fixed point mesh adaptation is able to predict the solution progression and to mesh suitably the regions of evolution by means of an implicit iterative algorithm. As the solution is anticipated for a period of time $[t, t + \Delta t]$ and the computation is reiterated, we can think that the efficiency of such approach will be greatly reduced. Nevertheless, we have demonstrated that this method is really efficient in terms of run time and complexity (the run time has been reduced by a factor of 9 and 17 for the isotropic and anisotropic cases, respectively) and also in terms of accuracy as it controls the space and the time error along the computation. This approach is more expensive than the classical one, but this is mandatory to solve the non-linear mesh adaptation problem and to obtain very accurate solutions.

Moreover, the scheme accuracy is independent of the chosen period step Δt, i.e., the physical time length between two adaptations, as the domain is automatically meshed in an optimal manner such that the mesh characteristic time frame is approximatively equal to the period step Δt.

The efficiency of the method will be emphasized in the next section on Rayleigh–Taylor instabilities simulations and a realistic three-dimensional simulation of a blast in a town.

5. Application to Rayleigh–Taylor instabilities

The instability of an interface separating miscible fluids of different densities subject to gravity is usually known as a Rayleigh–Taylor instability (RTI) [30]. Here, we consider two inviscid fluids in unstable equilibrium in a rectangular cavity of size $[0, 0.25] \times [0, 1]$. The flow is modeled by the Euler equations with gravita-

tional fields of $(0,-1)$ and $(0,0,-1)$ in two and three dimensions. The upper half of the cavity is filled with a fluid whose density is two times heavier than that of the fluid filling the lower part (unit density). Both fluids are submitted to gravity force. The initial pressure corresponds to the hydrostatic equilibrium $\frac{\partial p}{\partial z} = -\rho g$. To initiate the instability, an initial perturbation of the velocity field is introduced. The light fluid rises into the heavy one as bubbles and, the heavy fluid falls into the light one as spikes, leaving behind an irregular region of mixed fluids.

In such simulations, the growth rate of this instability and the rate of mixing between the two fluids is sensitive to the numerical or physical mass diffusion. Therefore, a high resolution of the contact discontinuity is particularly crucial [31]. Mesh adaptation techniques seem to be appropriate as it is well known that they reduce the numerical diffusion by utilizing a high spatial resolution (a dense mesh) in the regions where the instabilities are created, i.e., at the interface between the two fluids.

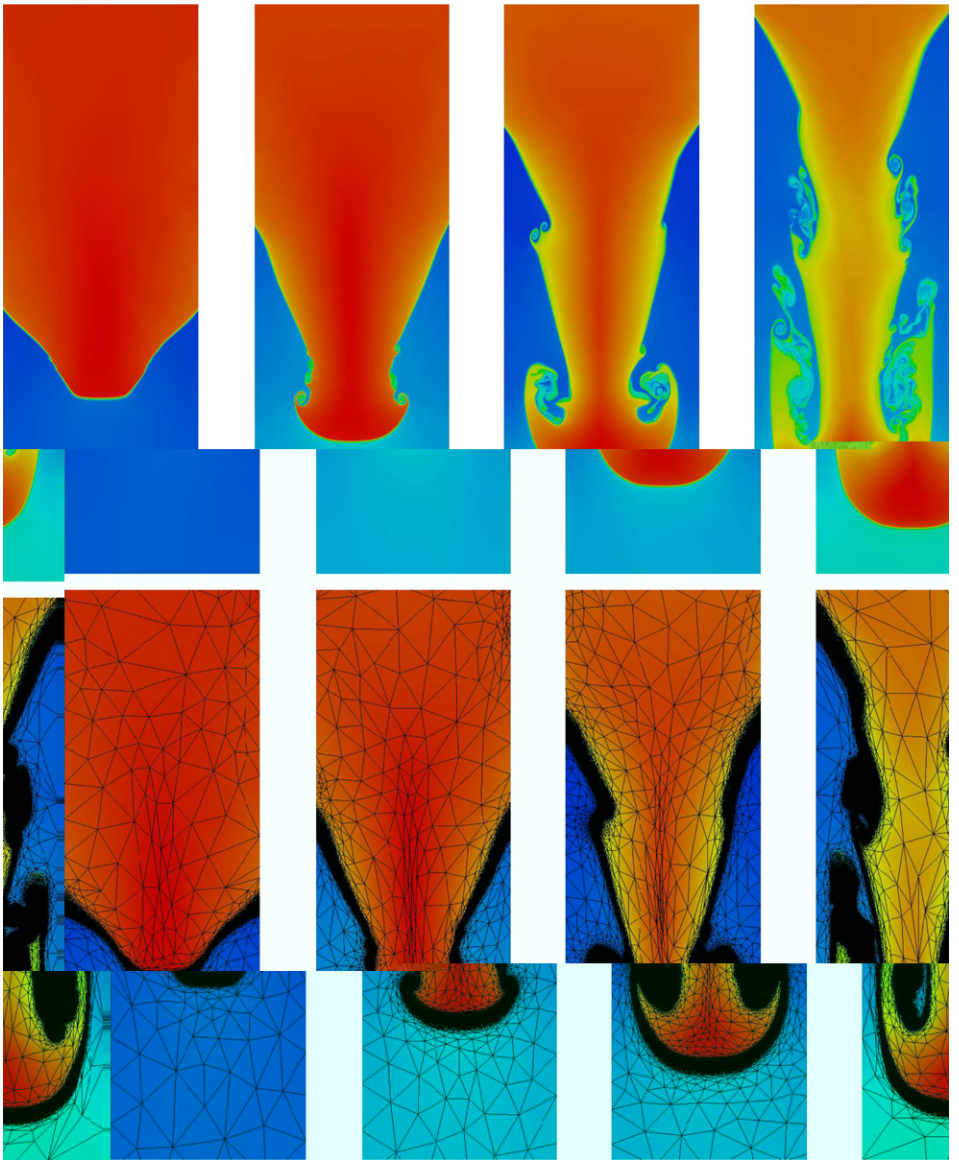

Fig. 11. 2D RTI development: density iso-values and corresponding meshes at different non-dimensional time $t = 0.4$, $t = 0.9$, $t = 1.3$ and $t = 1.7$ (left to right).

The proposed RTI examples aim at demonstrating that the unsteady mesh adaptation approach is suitable in dealing with such problem. But the accuracy or the validity of the obtained results, which are highly dependent on the resolution method, is beyond the scope of this paper.

5.1. A two-dimensional example

The solution has been computed at the non-dimensional time $t_{\max} = 1.7$. The time frame of the simulation is decomposed into 16 periods, i.e., we have performed 16 mesh adaptations, each involving four internal iter-

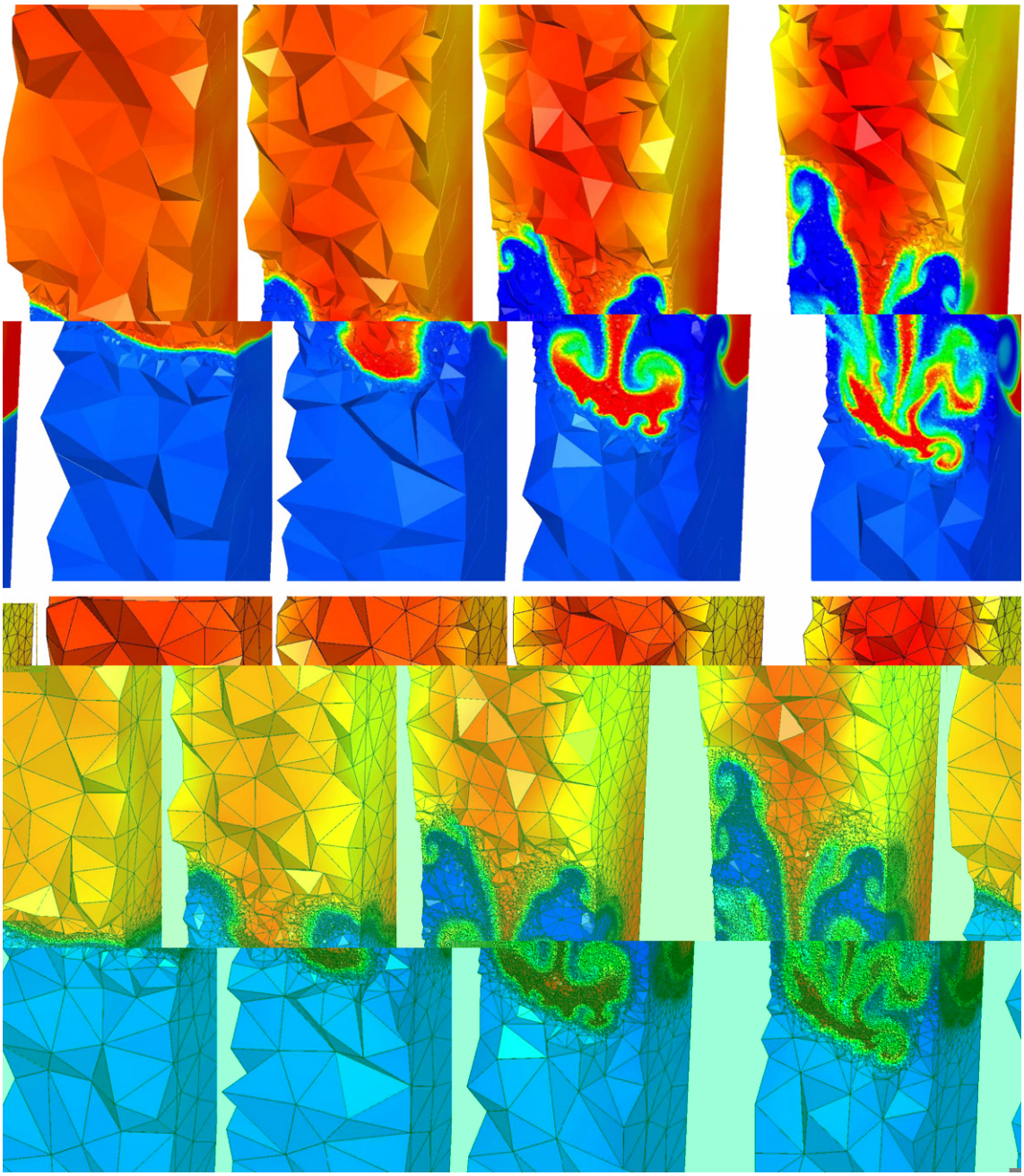

Fig. 12. 3D RTI development: density iso-values and cut through the corresponding meshes at different non-dimensional time $t = 0.2$, $t = 0.8$, $t = 1.4$ and $t = 1.8$ (left to right).

ations to solve the transient fixed point problem. The anisotropic metric is defined based on the density variations in order to accurately capture the contact discontinuity between the two fluids. Inside the internal loop, nine intermediate solutions are used for the metric intersection in time, the adaptation parameters being:

$$\varepsilon = 0.15, \quad \gamma = 0.004, \quad h_{\min} = 3 \times 10^{-4}, \quad h_{\max} = 0.1 \quad \text{and} \quad h_{\text{grad}} = 3,$$

where ε is the prescribed error, $h_{\min}$ and $h_{\max}$ are the minimal and the maximal size allowed, respectively, and h_{grad} is the mesh size gradation. This simulation has been performed on a PC workstation, the final anisotropic adapted mesh contains 240,126 vertices and 479,914 triangles.

With the mesh adaptation procedure, a high spatial resolution is obtained and preserved at the interface between the two fluids (see Fig. 11). Moreover, the new mesh adaptation algorithm allows automatically the spatial resolution to be preserved at the interface of the two fluids throughout the progression of the phenomenon in order to accurately resolve contact discontinuities.

In this simulation, it can be observed that anisotropic meshes directly impact the development of the Rayleigh–Taylor instabilities. Such meshes preserve the mushroom shape of the Rayleigh–Taylor finger and all the mixing regions are located behind it. On the other hand, with isotropic meshes these features are more chaotic and a lot of perturbations appear on the top of the finger. Hence, less instabilities are obtained with anisotropic meshes despite the fact that a higher spatial resolution could be achieved. Moreover, in contrast to the isotropic meshes, anisotropic meshes tend to preserve the symmetry of the phenomenon even if the mesh is not symmetric. Such results would lead to assume that isotropic meshes introduce numerical perturbations which create their own virtual instabilities, a somewhat similar problem to that of the round-off errors that could introduce their own perturbations.

5.2. A three-dimensional example

The solution has been computed at the non-dimensional time $t_{\max} = 1.8$. In this case, the time frame of the simulation is decomposed into nine periods, i.e., we have performed nine mesh adaptations, and at each period we have done four internal iterations to solve the transient fixed point problem. The isotropic metric field is defined based on the density variations. Inside the internal loop, nine intermediate solutions are used for the metric intersection in time and the adaptation parameters have been set to:

$$\varepsilon = 0.008, \quad h_{\min} = 3 \times 10^{-3}, \quad h_{\max} = 0.1 \quad \text{and} \quad h_{\text{grad}} = 3.$$

This three-dimensional simulation has been performed on a PC workstation. The final adapted mesh contains 769,719 vertices and 4,536,589 tetrahedra.

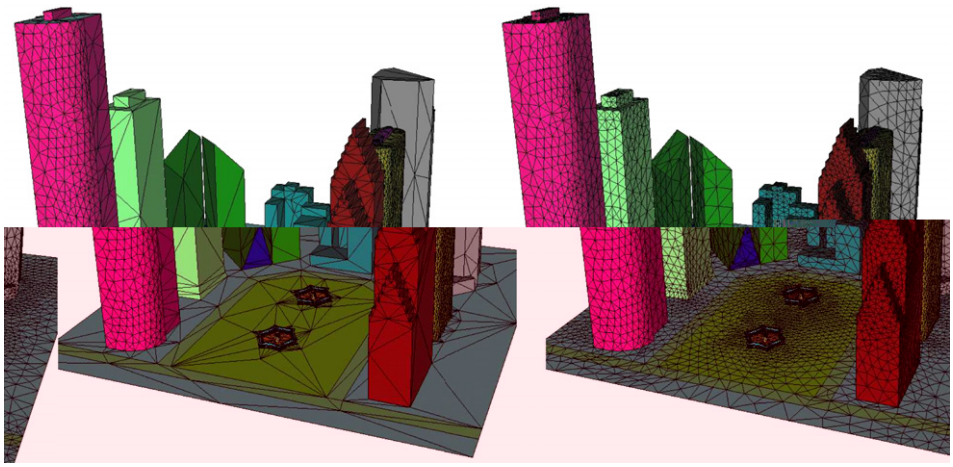

Fig. 13. Geometric surface mesh (left) and initial computational surface mesh (right) of the city plaza.

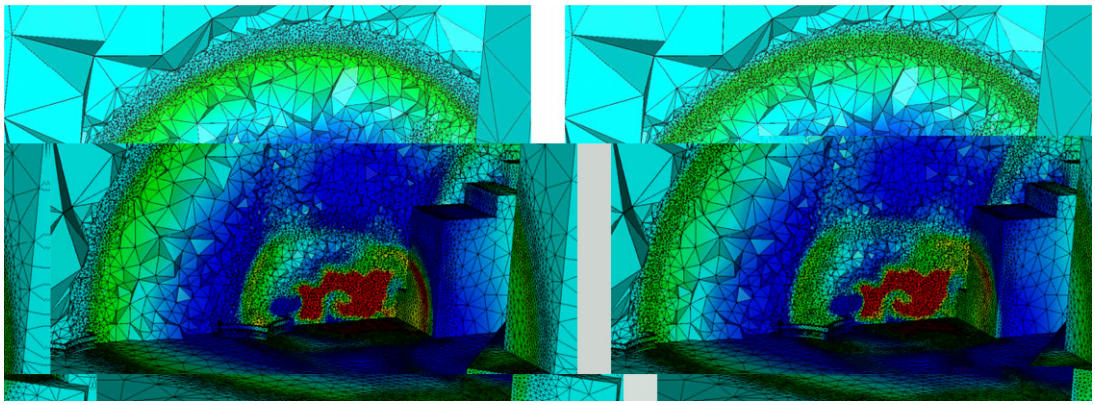

Fig. 14. Impact of the metric intersection in time in the volume and on the surface. Left, solution at the beginning of the period and, right, solution at the end of the period. The shocks waves progress only in the uniformly refined regions.

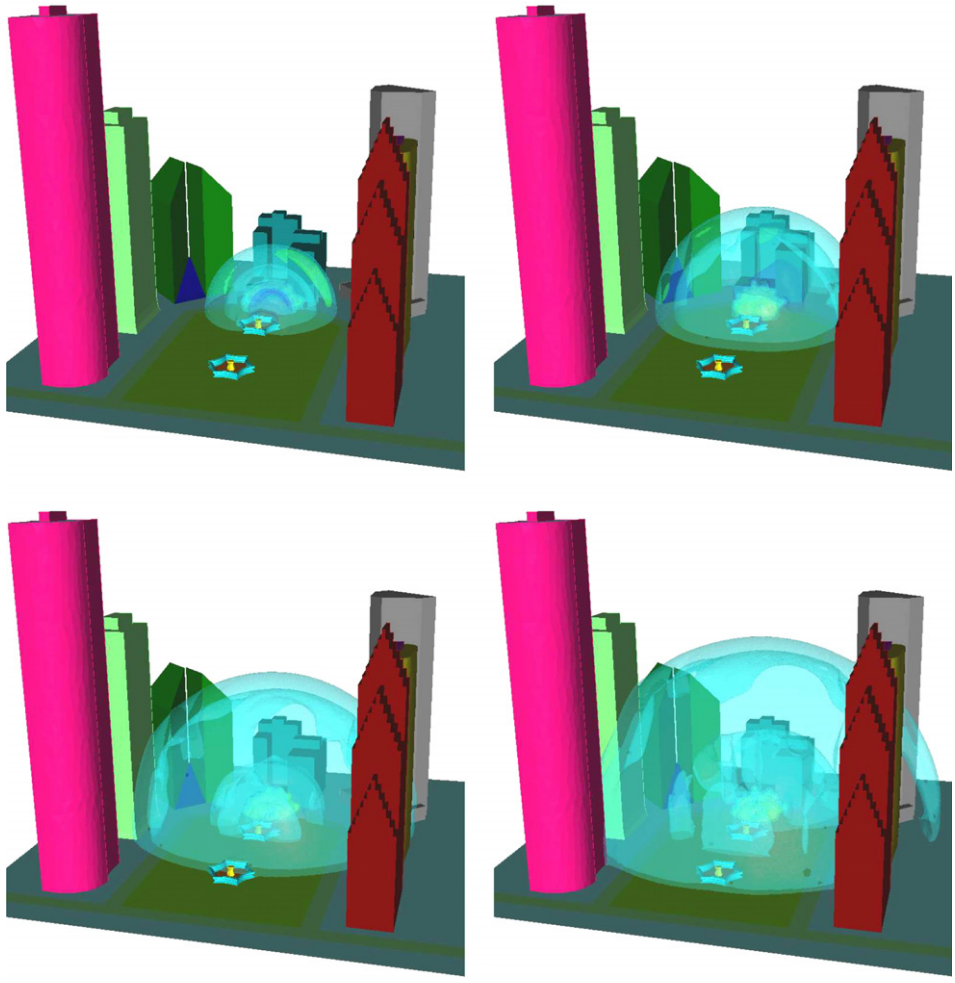

Fig. 15. The progression of the density's iso-surfaces in the domain at different time 0.027, 0.05, 0.077 and 0.1 s (form left to right and from top to bottom).

Similar to the two-dimensional case, a high spatial resolution at the interface between the two fluids is obtained with the new mesh adaptation procedure (see Fig. 12). The spatial resolution is preserved at the interface of the two fluids all throughout the progression of the phenomenon with only nine mesh adaptations. Mesh adaptation makes it possible to obtain a sufficient resolution for the development of instabilities in three dimensions. However, contrary to the previous example, the use of isotropic mesh adaptation has given us a more chaotic development of the Rayleigh–Taylor instabilities, even if coarser meshes (larger element sizes) have been used. In Fig. 12, one can see that several instabilities appear on the top of the Rayleigh–Taylor finger and that the mushroom shape of the phenomena has not been preserved.

6. 3D non-linear waves propagation

In this section, the transient fixed point based mesh adaptation scheme is applied on a realistic three-dimensional case. The problem concerns non-linear waves propagation simulating the development of a blast in a complex geometry (see Fig. 13). Here, the atmosphere has the same values as in 2D. The initial high density

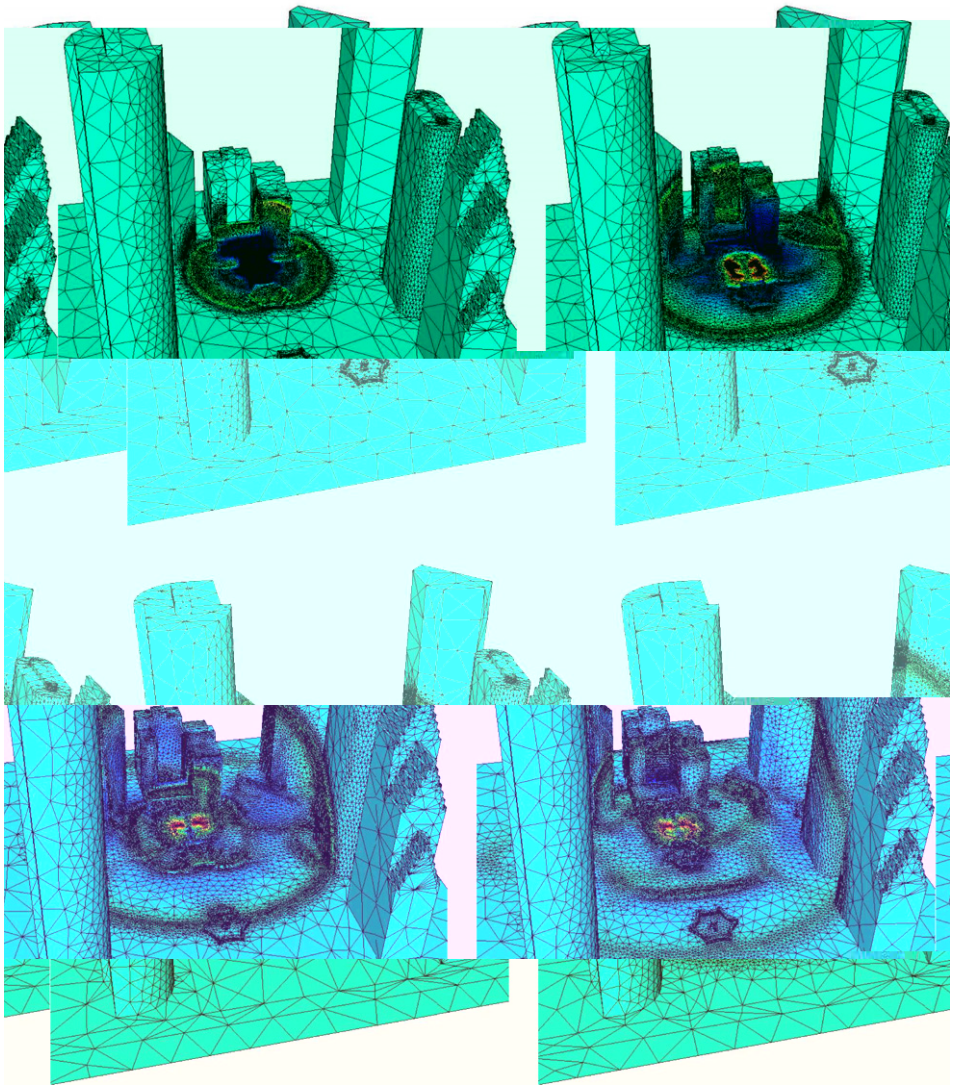

Fig. 16. Isotropic adapted surface meshes and the corresponding iso-density distributions at different time 0.027, 0.05, 0.077 and 0.1 s.

region is a hemisphere with a radius of 2.5 m inside which the density, pressure and speed vector are initialized to $\rho = 1$, $p = 1$ and $u = v = w = 0$, respectively.

In this example, the objective is to compute the solution at physical time 0.1 s. The simulation is split into 30 periods, i.e., 30 mesh adaptations are performed, corresponding to a period step of $\Delta t = 0.0033$ s. In each main loop iteration, three internal iterations are performed to solve the transient fixed point problem. The isotropic metric is defined based on density variations.

We used five intermediate solutions, including the initial and final one, for the metric intersection in time. The adaptation parameters are:

$$\varepsilon = 0.003, \quad h_{\min} = 0.3 \text{ m}, \quad h_{\max} = 10 \text{ m} \quad \text{and} \quad h_{\text{grad}} = 3,$$

where ε is the prescribed error threshold, $h_{\min}$ and $h_{\max}$ are the minimal and the maximal element size allowed, respectively, and h_{grad} is the mesh size gradation. Notice that in this example a mesh of uniform size $h_{\min} = 0.3$ m will contain approximately 1.8×10^7 vertices. An initial coarse mesh with 40,230 vertices has been

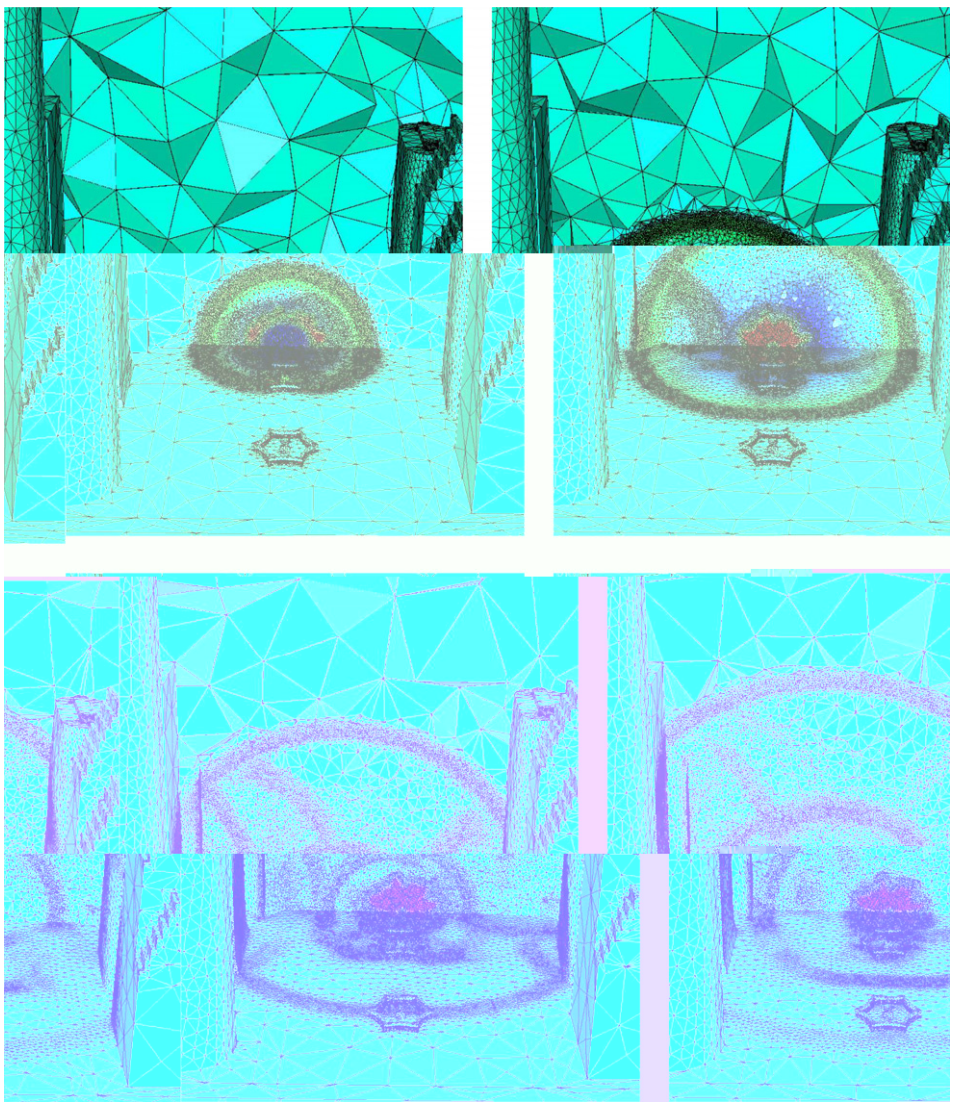

Fig. 17. Isotropic adapted volume meshes in a cut plane and the corresponding iso-density distributions at different time 0.027, 0.05, 0.077 and 0.1 s.

Table 5
Statistics of the adapted meshes at different iterations

Iteration	Physical time	Vertices	Tetrahedra	Triangles
8	0.027	280,525	1,630,619	40,736
15	0.05	603,644	3,541,268	63,526
23	0.077	739,854	4,326,861	78,816
30	0.1	743,735	4,328,741	87,322

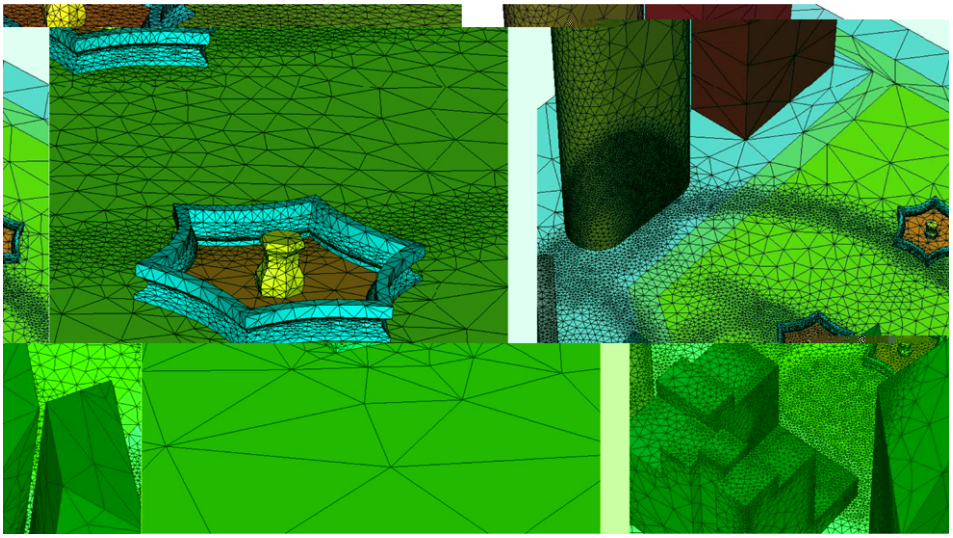

Fig. 18. Zoom on the adapted surface mesh pointing out the intersection between the computational metric $\mathscr{M}$ and the geometrical metric $\mathscr{G}$.

generated (see Fig. 13, right). Fig. 15 shows the shock wave propagation (the density isosurface) in the city plaza at various physical times. It clearly illustrates the unpredictable behavior of physical phenomena, with notably a large number of rebounds on the buildings.

Figs. 16 and 17 show the isotropic adapted surface and volume mesh with the corresponding iso-density distributions at various physical times, respectively. The corresponding meshes sizes are reported in Table 5. In these figures, the mesh density clearly indicates the region where the shock waves progress. Mesh refinement has been prescribed by the metric intersection in time, which allows us to obtain a uniform mesh density in order to preserve the solution accuracy along the computation.

The metric intersection in time is illustrated in Fig. 14. Notice that the adapted area is "optimal" in the sense that the mesh characteristic time τ is approximately equal to the period step Δt between two adaptations.

Fig. 18 exemplifies the intersection between the computational metric $\mathscr{M}$, which is used to control the solution error, and the geometrical metric $\mathscr{G}$, which is utilized to control geometric approximation of the domain. The mesh is clearly refined on curved boundary (e.g. fountain, building) and in the regions where shock waves progress.

This simulation has been performed on a Macintosh G5 with 2.5 GHz PowerPC processor and 1 Gb of RAM. At the 23rd adaptation (period) for the last internal loop iteration, the generation of the final surface mesh (78,816 triangles) and the final volume mesh (739,854 vertices) took 10 and 216 s, respectively. It required 20 s to compute one metric thus it took 1 min to compute all metrics and the metric intersection and it required 1 min for the interpolation stage. On the other hand, the number of time steps of the explicit flow solver is related to the flow characteristics and the mesh size. Consequently, at this iteration the solver has made 90 time steps with the 4-stage, 3-order SSP Runge–Kutta scheme with a CFL equal to 1.8 to increment the solution in time by Δt and the cpu time required to compute the Euler solution took approximately 1 h. It took two and a half days to carry out the whole simulation on a single processor machine.

7. Conclusion

In this paper, we have identified a class of CFD time-dependent problems for which we have demonstrated that the classical mesh adaptation algorithm is intrinsically unable to deal with. A new mesh adaptation method based on a transient fixed point algorithm has been proposed to solve such problems in the case of global remeshing. This method uses a generalization of the classical mesh adaptation algorithm in order to predict the evolution of the solution and metric intersection in time procedure to adapt the mesh. With this method, the time has been implicitly introduced in the error estimate to control the error throughout the simulation. This new approach preserves the solution accuracy and reduces the number of adaptations to control the error due to the interpolation stage In other words, it allows us to control the characteristic time frame of the mesh.

The two-dimensional results on the model problem have confirmed the validity and the efficiency of this approach with respect to the accuracy of the solution and the reduction of the complexity. It has been successfully applied on Rayleigh–Taylor instabilities and in three dimensions on a real simulation on a complex geometry where the solution is a priori unpredictable. All the numerical results presented have pointed out the progress in efficiency and in accuracy that have been accomplished since [17] due to the improvement described in this paper.

Notice that the whole adaptation scheme has been carried out on a workstation in a reasonable amount of time. The classical bottlenecks of such 3D simulations have been avoided without requiring the use of the parallel computing. Parallel computing can of course be used to reduce further the overall cpu time.

All the results obtained so far have emphasized the importance of the interpolation stage for unsteady simulations. Therefore, the main challenges that we are facing are related to:

- finding interpolation schemes with the same order of accuracy as the flow solver to reduce the error due to this stage;
- reducing the problem complexity, at the same error level, by using anisotropic mesh adaptation in three dimensions;
- analyzing the convergence order of the adaptive scheme.

All works are in progress.

References

[1] R. Biswas, R. Strawn, A new procedure for dynamic adaption of three-dimensional unstructured grids, Appl. Numer. Math. 13 (1994) 437–452.
[2] Y. Kallinderis, P. Vijayan, Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes, AIAA J. 31 (8) (1993) 1440–1447.
[3] D. Mavriplis, Adaptive meshing techniques for viscous flow calculations on mixed-element unstructured meshes, AIAA paper 97-0857.
[4] J. Peraire, J. Peiro, K. Morgan, Adaptive remeshing for three-dimensional compressible flow computations, J. Comput. Phys. 103 (1992) 269–285.
[5] P. Frey, F. Alauzet, Anisotropic mesh adaptation for CFD computations, Comput. Methods Appl. Mech. Eng. 194 (48-49) (2005) 5068–5082.
[6] C. Gruau, T. Coupez, 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric, Comput. Methods Appl. Mech. Eng. 194 (48-49) (2005) 4951–4976.
[7] X. Li, M. Shephard, M. Beall, 3D anisotropic mesh adaptation by mesh modification, Comput. Methods Appl. Mech. Eng. 194 (48–49) (2005) 4915–4950.
[8] A. Tam, D. Ait-Ali-Yahia, M. Robichaud, M. Moore, V. Kozel, W. Habashi, Anisotropic mesh adaptation for 3D flows on structured and unstructured grids, Comput. Methods Appl. Mech. Eng. 189 (2000) 1205–1230.
[9] R. Rausch, J. Batina, H. Yang, Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations, AIAA J. 30 (1992) 1243–1251.
[10] R. Löhner, Three-dimensional fluid–structure interaction using a finite element solver and adaptive remeshing, Comput. Syst. Eng. 1 (2-4) (1990) 257–272.
[11] R. Löhner, J. Baum, Adaptive *h*-refinement on 3D unstructured grids for transient problems, Int. J. Numer. Meth. Fluids 14 (1992) 1407–1419.

[12] W. Speares, M. Berzins, A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems, Int. J. Numer. Meth. Fluids 25 (1997) 81–104.
[13] P. de Sampaio, P. Lyra, K. Morgan, N. Weatherill, Petrov–Galerkin solutions of the incompressible Navier–Stokes equations in primitive variables with adaptive remeshing, Comput. Methods Appl. Mech. Eng. 106 (1993) 143–178.
[14] J. Wu, J. Zhu, J. Szmelter, O. Zienkiewicz, Error estimation and adaptivity in Navier–Stokes incompressible flows, Comput. Mech. 6 (1990) 259–270.
[15] C. Pain, A. Humpleby, C. de Oliveira, A. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Eng. 190 (2001) 3771–3796.
[16] J.-F. Remacle, X. Li, M. Shephard, J. Flaherty, Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods, Int. J. Numer. Meth. Eng. 62 (2005) 899–923.
[17] F. Alauzet, P.-L. George, B. Mohammadi, P. Frey, H. Borouchaki, Transient fixed point based unstructured mesh adaptation, Int. J. Numer. Meth. Fluids 43 (6-7) (2003) 729–745.
[18] P. Frey, P.-L. George, Mesh Generation. Application to Finite Elements, Hermès Science, Paris, Oxford, 2000.
[19] C. Debiez, A. Dervieux, Mixed-element-volume MUSCL methods with weak viscosity for steady and unsteady flow calculations, Comput. Fluids 29 (2000) 89–118.
[20] P. Batten, N. Clarke, C. Lambert, D. Causon, On the choice of wavespeeds for the HLLC riemann solver, SIAM J. Sci. Comput. 18 (6) (1997) 1553–1570.
[21] P.-H. Cournède, C. Debiez, A. Dervieux, A positive MUSCL scheme for triangulations, Research Report INRIA, RR-3465, 1998.
[22] R. Spiteri, S. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM J. Numer. Anal. 40 (2) (2002) 469–491.
[23] M. Castro-Diaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaptation for flow simulations, Int. J. Numer. Meth. Fluids 25 (1997) 475–491.
[24] P. Frey, About surface remeshing, in: Proceedings of the 9th International Meshing Rountable, New Orleans, LO, USA, 2000, pp. 123–136.
[25] P.-L. George, Tet meshing: construction, optimization and adaptation, in: Proceedings of the 8th International Meshing Rountable, South Lake Tao, CA, USA, 1999.
[26] H. Borouchaki, F. Hecht, P. Frey, Mesh gradation control, Int. J. Numer. Meth. Eng. 43 (6) (1998) 1143–1165.
[27] A. Harten, P. Lax, B.V. Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, SIAM Rev. 25 (1) (1983) 35–61.
[28] C. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[29] G. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1978) 1–31.
[30] D. Sharp, An overview of Rayleigh–Taylor instability, Physica D 12 (1984) 3–18.
[31] X. Li, B. Jin, J. Glimm, Numerical study for the three-dimensional Rayleigh–Taylor instability through the TVD/AC scheme and parallel computation, J. Comput. Phys. 126 (1996) 343–355.