

Transient fixed point-based unstructured mesh adaptation

F. Alauzet^{1,*,\dagger,\ddagger}, P. L. George¹, B. Mohammadi^{2,\S}, P. Frey¹ and H. Borouchaki¹

¹ INRIA, Gamma, Domaine de Voluceau, Rocquencourt B.P.105, Le Chesnay 78153, France

² Université de Montpellier II, Math CC51, Montpellier 34090, France

SUMMARY

This paper deals with adaptive simulation of time-dependent problems. Mesh adaptation for unsteady configurations is especially important as the phenomenon evolves in the whole computational domain and, in order to capture such a phenomenon accurately on adapted meshes, one has to predict the evolution path. Here, we propose a new adaptive algorithm for time-dependent simulation; the idea is to solve a transient fixed problem by introducing a new loop in the adaptation scheme. The metric used for 3D simulations is constructed with an *a posteriori* error estimate based on a discrete approximation of the Hessian of the solution. For time-dependent simulation, a metric intersection in time is introduced to refine all the areas where the phenomena evolve. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: unstructured mesh generation; mesh adaptation; metric; time-dependent simulation; CFD

1. INTRODUCTION

Mesh adaptation has proved useful for solving PDEs, as it allows to control the desired accuracy for the solution while reducing the overall cost of the computational time. This is particularly true for CFD problems. In the past, steady phenomena have been successfully addressed using mesh adaptation in two dimensions [1–3]. The idea of the mesh adaptation method for steady-state configurations is to converge toward a desired fixed point for the pair formed by the mesh and the solution. In this work, the aim is to extend this approach to three dimensions as well as to time-dependent simulations for Euler.

Mesh adaptation for unsteady configurations is especially important as the phenomena evolve in the whole computational domain, requiring a uniformly sized fine mesh everywhere without adaptation. The problem here, as compared to steady cases, is that there is not a fixed state to converge to. Thus, it is easy to see that if one wants to capture a time-dependent

* Correspondence to: F. Alauzet, INRIA-Gamma, Domaine de Voluceau-Rocquencourt, B.P. 105, Le Chesnay 78153, France.

† E-mail: Frederic.Alauzet@inria.fr

‡ Partially funded by Région Ile de France.

§ E-mail: mohamadi@math.univ-montp2.fr; URL: <http://www.math.univ-montp2.fr/>

phenomenon and to have a mesh suitable to compute the solution at a given time t , one has to predict the phenomenon evolution. And this, to avoid remeshing finely a part too large of the computational domain or to remesh too frequently which will be very costly for 3D complex geometries. In this context, few works on mesh adaptation have been done [4–6] based on refinement-coarsening techniques and not with unstructured mesh generation methods.

We propose a new adaptive algorithm, as the classical one is not suitable for such simulations. The idea is to solve a transient fixed point problem by introducing a new loop in the main adaptation loop. More precisely, at each internal iteration of the adaptation loop (from time t to time $t + \Delta t$), we start with an initial solution at t and the solution is computed until $t + \Delta t$. A metric associated with the computed solution is defined and is used to generate a new adapted mesh. Instead of continuing the computation like in the classical case, the computation is restarted with the same initial solution (at time t) but using now the new mesh. The iterative process is repeated until it converges toward a fixed point for $t + \Delta t$. When the convergence is reached, we advance to the next time step and the solution at $t + \Delta t$ becomes the new initial condition.

The metric is constructed with an *a posteriori* error estimate based on a discrete approximation of the Hessian of the solution. For time-dependent simulations, a new constraint is introduced in the metric definition, namely a metric intersection in time, which allows to refine all the area where the phenomenon evolved in the time frame and not just where it ends. The role of the error estimate is to indicate if the mesh density is locally appropriate. To match this density requirement, an adapted mesh needs to be generated. Therefore, in the isotropic case, a discrete scalar map is defined at the mesh vertices and is used to prescribe the local sizes of the mesh elements. At each stage of the adaptation scheme [7], the generation of an adapted mesh consists in two steps: (i) the construction of an adapted surface mesh [8] and (ii) the construction of a 3D adapted mesh [9]. The governed mesh creation (surface and volume) is based on the edge lengths computation with respect to the given metric (supplied at the vertices of the current mesh) and the objective is then to construct a *unit* mesh (in which all edges have a length close to 1) [10].

Finally, time-dependent simulations for a multi-dimensional Riemann problem are presented. The classical and the new version of the algorithm are compared. The impact of this new adaptation method on the complexity of the computation is analysed.

2. ADAPTIVE MESH GENERATION

In this section, we recall the notion of a regular mesh and we show that this is the key of our adaptive meshing problem.

A quality simplicial mesh or a *regular* mesh of domain Ω in \mathbb{R}^2 (resp. \mathbb{R}^3) is a mesh in which the elements are equilateral (resp. regular). As such a mesh does not necessarily exist, a regular mesh is simply the ‘best’ mesh that can be created.

2.1. Size-constrained regular meshes

We face here a general mesh generation problem which aims at constructing a mesh in which the element sizes is conformal to some given (isotropic or anisotropic) size specifications. A mesh conforms to a size map h if all of its edges have an ‘average’ length equal to the

average of the sizes specified along these edges. To this end, we assume that the data of the size map $h(X, \vec{d})$ ($\forall X, \vec{d}$) allows for the local definition of a metric (or a metric tensor) $\mathcal{M}(X)$ at X . The average length $l_m(e)$ of edge $e = PQ$ may be defined as the average of the lengths $L_{\mathcal{M}(X)}(e)$ (the length of e computed using $\mathcal{M}(X)$) when X moves along e :

$$l_m(e) = \frac{\|\int_X L_{\mathcal{M}(X)}(e) dX\|}{\|\int_X dX\|}$$

for $X = P + t \vec{PQ}$, we obtain:

$$l_m(PQ) = \int_0^1 L_{\mathcal{M}(P+t\vec{PQ})}(PQ) dt$$

The key is to find a metric $\mathcal{M}(X)$ (a symmetric positive definite matrix) which conforms at best to the given size map. The geometric locus of points Y which conform to metric $\mathcal{M}(X)$ at point X is in general an ellipsoid $\mathcal{E}(X)$ whose equation can be written as ${}^t\vec{XY} \mathcal{M}(X) \vec{XY} = 1$.

In the particular case, where the size map $h(X, \vec{d})$ only depends on X (isotropic case), metric $\mathcal{M}(X)$ reduces to

$$\mathcal{M}(X) = \frac{1}{h^2(X)} \mathcal{I}_d$$

where \mathcal{I}_d is the identity matrix in \mathbb{R}^d .

The size map $h(X, \vec{d})$ ($\forall X, \vec{d}$) is then seen as the metric map $\mathcal{M}(X)$ and a mesh conforming to this metric is a mesh where the edges have an average unit length. A mesh with this property is said to be a *unit mesh*. One could observe that this average edge length in a metric is nothing else than an edge length if we associate a Riemannian structure with the domain, this structure being defined by the metric map $\mathcal{M}(X)$. In this structure, the length $L_{\mathcal{M}}$ of edge PQ is given by

$$L_{\mathcal{M}}(PQ) = \int_0^1 \sqrt{{}^t\vec{PQ} \mathcal{M}(P+t\vec{PQ}) \vec{PQ}} dt$$

To summarize, a mesh is said to conform to a given size map $h(X, \vec{d})$ ($\forall X, \vec{d}$) if it is a unit mesh in the Riemannian structure associated with the underlying metric map.

In our case, the Riemannian structure is defined as a discrete structure and consists in defining the map by means of interpolation from the data of the map at the vertices of a current *background mesh*.

However, a unit mesh conforming to a metric map is not necessarily suitable for finite element purposes. It may be desirable to add another criterion regarding the element shapes. Indeed, the element shape quality widely depends on the size variation present in the metric map. To avoid this, it is only necessary to modify the metric map [11], in accordance with the desired size (while preserving certain properties included in the map).

2.2. Unit mesh construction

Throughout this section, we propose a general purpose method to construct a unit mesh of a domain Ω in \mathbb{R}^d , $d=2$ or 3 (the domain being defined by its boundary Σ), supplied with

a Riemannian metric \mathcal{M}_d . This method consists in meshing Ω in such a way that the edges in this mesh are of length one. If P is a vertex in the unit mesh of Ω and if PX is an edge sharing P , then one must have:

$$\int_0^1 \sqrt{{}^t\overline{PX} \mathcal{M}_d(P + t\overline{PX})\overline{PX}} dt = 1$$

The proposed method includes the optimization (remeshing) of the boundary mesh by means of unit elements and the construction of a unit mesh in Ω based on the above boundary discretization.

2.2.1. Unit remeshing of the boundary of Ω . In two dimensions, the boundary Σ of Ω is composed of curved segments, the problem is then to discretize these segments into unit length segments. For surface meshes, the problem is slightly more tedious. First, a geometric metric \mathcal{G} of the minimal radius of curvature in the isotropic case is computed at the vertices of the current mesh. Then, this metric is intersected with the given computational metric \mathcal{M} . The goal is now to complete a unit mesh with respect to this new metric.

To this end, the current mesh edges are analysed. More precisely, the length of each mesh edge is computed. The ideal edge length being one, the edges having a length larger than one are splitted into unit segments while the edge having a length too small are deleted [8], for instance, by means of collapses. After each mesh modification, the current mesh is optimized so as to improve the element shape quality. At completion, all mesh edges have a length close to one.

Geometric metric definition: The construction of the geometric metric involves the computation of the principal curvatures and principal directions at all mesh vertices. The principal curvatures at a point P of a C^2 surface can be computed numerically based on the given surface triangulation. To this end, the underlying surface geometry is locally approached by a quadric surface Σ defined by a least-squares fit of adjacent mesh vertices:

$$F(x, y, z) = ax^2 + bxy + cy^2 - z = 0$$

where a, b, c are three coefficients to be determined. We obtain a (usually over determined) linear system of m equations. Solving this system is equivalent of minimizing the following sum (a, b, c are solutions of an optimization problem):

$$\min \sum_i^m (ax_i^2 + bx_i y_i + cy_i^2 - z_i)^2 \quad (1)$$

that corresponds to minimizing the square of the norm of the distances to the quadric surface. Once quadric surfaces have been locally defined at mesh vertices, they are used to compute the local principal curvatures at the mesh vertices [8].

The geometric metric allows to bound the gap between a mesh edge and the surface by a given threshold value ε . A matrix of the form

$$\mathcal{G}_3(P)_{\rho_1, \rho_2} = {}^t\mathcal{D}(P) \begin{pmatrix} \frac{1}{\alpha^2 \rho_1^2(P)} & 0 & 0 \\ 0 & \frac{1}{\beta^2 \rho_2^2(P)} & 0 \\ 0 & 0 & \lambda \end{pmatrix} \mathcal{D}(P) \quad (2)$$

where $\mathcal{D}(P)$ corresponds to the principal directions at P , $\rho_1 = 1/\kappa_1$, $\rho_2 = 1/\kappa_2$ are the main radii of curvature, α and β are appropriate coefficients and $\lambda \in \mathbb{R}$, provides an *anisotropic* (curvature-based) control of the geometry.

Geometric support: A geometric support can be constructed internally. It represents the analytical definition of a surface and can be used to emulate the features of a geometric modelling system. The construction of the support involves the definition of a piecewise linear surface of order G^1 based on the geometric mesh previously extracted. Each triangle represents a patch, two adjacent patches sharing a common tangent plane to ensure the desired continuity property (except if the common edge is a ridge).

Given a mesh vertex, the geometric support is used to supply the location of the closest point onto the surface from the point. Moreover, at a G^1 continuous point, the surface normal and the principal radii of curvature can be returned by the support. For a point located along a ridge (where the tangent planes are not G^1 continuous), the tangent to the curve at the point is returned.

Surface mesh adaptation: This stage involves the construction of a unit mesh (with respect to the metric \mathcal{G}) using local geometric and topologic mesh modifications. Practically, the optimization procedure consists in analysing the current mesh edges in order to collapse the short edges and to split the large ones. An edge is considered as short (resp. long) if its length is lesser (resp. larger) than $1/\sqrt{2}$ (resp. $\sqrt{2}$). This procedure is based on edge splitting, edge collapsing and edge swapping operations. Moreover, several geometric measures are introduced to control the deviation between the mesh elements and the surface geometry as well as the element shape quality.

2.2.2. Unit mesh of Ω . The global scheme for unit mesh generation is classical: a coarse mesh (without internal points) of the domain is constructed using a classical Delaunay-based method, then this mesh is enriched by adding the field points before being optimized. The field points are defined in an iterative manner. At each iteration step, these points are created using an edge saturation method (i.e. the edges in the mesh are all of unit length). Then, they are inserted in the current mesh using the *constrained Delaunay kernel*, in a Riemannian context [9]. This process is repeated as long as the current mesh is modified. With this approach, the field points are necessarily well located with respect to the mesh entities already constructed. Similarly, the Delaunay kernel results in nearly optimal connexions[¶] from point to point.

Edge saturation: At each iteration step, the field points are defined in such a way as to subdivide the edges in the current mesh by means of unit length segments. The length of an edge in the current mesh is computed using the specified size field. In the case where the size field is given by an interpolation scheme from a background mesh, the edge is immersed in the background mesh in order to determine the size at each of its point. A field point is retained if it is not too close (i.e. at a distance less than 1) to an already existing point.

Point insertion: In a classical problem, the constrained Delaunay kernel can be written as $\mathcal{H} = \mathcal{H} - C(P) + B(P)$ where $C(P)$ is the cavity associated with point P and $B(P)$ is the re-meshing of $C(P)$ based on P (\mathcal{H} being the current mesh) [12]. An extension of this approach consists of redefining the cavity $C(P)$ in a Riemannian context [9]. To this end, we first introduce the *Delaunay measure* $\alpha_{\mathcal{M}_d}$ associated with the pair (P, K) , with respect to a

[¶] An optimal connexion is what we meet when regular elements exist.

metric \mathcal{M}_d :

$$\alpha_{\mathcal{M}_d}(P, K) = \left[\frac{d(O_K, P)}{r_K} \right]_{\mathcal{M}_d}$$

where O_K (resp. r_K) is the centre (resp. radius) of the circumsphere of K and $[*]_{\mathcal{M}_d}$ indicates that the quantity $*$ is measured in the Euclidean space characterized by the metric \mathcal{M}_d . The usual proximity criterion is expressed by $\alpha_{\mathcal{I}_d}(P, K) < 1$, where \mathcal{I}_d is the identity metric. Hence, region $C(P)$ is completed by adjacency starting from the element(s) containing P .

Mesh optimization: The proposed method results in a unit mesh of domain Ω . Nevertheless, the mesh quality can be improved by means of topological as well as geometrical modifications. The optimization scheme consists of iterative facet flips and node repositioning, based on the notion of edge quality and element quality.

- *Edge length quality:* The length quality Q_l of an edge AB in the Riemannian metric \mathcal{M}_d is defined as

$$Q_l(AB) = \begin{cases} L_{\mathcal{M}_d}(AB) & \text{if } L_{\mathcal{M}_d}(AB) \leq 1 \\ \frac{1}{L_{\mathcal{M}_d}(AB)} & \text{if } L_{\mathcal{M}_d}(AB) > 1 \end{cases}$$

- *Element shape quality:* In the classical Euclidean space, a popular measure for the shape quality of a mesh element K is [13]:

$$Q_f(K) = c \frac{V(K)}{\sum_{e(K)} l^2(e(K))}$$

where $V(K)$ denotes the volume of K , $e(K)$ being the edges in K and c is a scaling coefficient such that the quality of a regular element is valued by 1. In a Riemannian space, the quality of element K is defined by

$$Q_f(K) = \min_{1 \leq i \leq d+1} Q_f^i(K)$$

where $Q_f^i(K)$ is the element quality in the Euclidean space associated with the metric \mathcal{M}_d^i corresponding to the vertex number i in K .

To measure the quality $Q_f^i(K)$, it is only required to transform the Euclidean space related to the metric specified at vertex i of K in the usual Euclidean space and to consider the quality value of element K^i associated with K , in other words: $Q_f^i(K) = Q_f(K^i)$. It is easy to show that:

$$Q_f^i(K) = c \frac{\sqrt{\text{Det}(\mathcal{M}_d^i)} V(K)}{\sum_{e(K)} l_{\mathcal{M}_d^i}^2(e(K))}$$

- *Mesh quality:* The quality (in terms of edge lengths or element shapes) of a mesh \mathcal{H} is defined by

$$Q(\mathcal{H}) = \left(\frac{1}{|\mathcal{H}|} \sum_{E \in \mathcal{H}} Q(E), \min_{E \in \mathcal{H}} Q(E) \right)$$

where $Q(E)$ stands for $Q_l(AB)$ or $Q_f(K)$.

The facet flip operation only affects the mesh topology. This technique results in the removal of a facet of arbitrary dimensionality when it is possible. Flipping a facet f relies in the construction of a triangulation of the hull of the set of elements sharing f where f is no longer a mesh entity. This operation is performed only if the quality of the new triangulation is improved [9].

Repositioning a point P consists in moving P so as to enhance the quality of the worse elements sharing P . Two techniques can be advocated for node repositioning. One based on unit edge lengths, the other on optimal shape quality. Actually, both methods are applied.

3. METRIC DEFINITION

This section describes the definition of the (physical) metric \mathcal{M} introduced in the previous section.

3.1. General definition

Let Ω be an open bounded domain of \mathbb{R}^3 with a regular boundary Γ . Let $V \in \Omega$ be the space where the problem is solved and V_h be the sub-space of V where the associated discrete problem is solved. We denote by $u \in V$ the solution of the problem and by $u_h \in V_h$ the solution of the associated discrete problem. We also denote by $\Pi_h u$ the Lagrangian P^1 interpolate of the solution u over the discretization. The error made over the mesh is less than or equal to the interpolation error [14, 17]:

$$\|u - u_h\|_{1,\Omega} \leq C \|u - \Pi_h u\|_{1,\Omega} \quad (3)$$

Thus, if the interpolation error is controlled, the error made over the mesh is controlled. Consequently, the key idea is to modify the scalar product used in the mesh generator for distance, area and volume evaluations, in order to equi-distribute this interpolation error. Therefore, the aim is to define a local metric, which replaces the Euclidean metric. Using an automatic mesh generation method (described above) with this metric, equilateral triangles in two dimensions (resp. regular tetrahedra in three dimensions) are constructed.

For a P^1 Lagrange finite element discretization of the variable u , the interpolation error is bounded by [14]:

$$\mathcal{E} = \|u - \Pi_h u\|_{0,\Omega} \leq ch^2 \|D^2 u\|_{0,\Omega} \quad (4)$$

h being the element size, $D^2 u$ the Hessian matrix. So, the scalar product of the local metric is based on the evaluation of the Hessian of the variables of the problem. The Hessian matrix is symmetric, therefore:

$$D^2 u = \begin{pmatrix} \partial^2 u / \partial x^2 & \partial^2 u / \partial y \partial x & \partial^2 u / \partial z \partial x \\ \partial^2 u / \partial x \partial y & \partial^2 u / \partial y^2 & \partial^2 u / \partial z \partial y \\ \partial^2 u / \partial x \partial z & \partial^2 u / \partial y \partial z & \partial^2 u / \partial z^2 \end{pmatrix} = \mathcal{R} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathcal{R}^{-1}$$

where \mathcal{R} is the eigenvectors matrix of D^2u and λ_i its eigenvalues ($\in \mathbb{R}$). Using this information, the following metric tensor \mathcal{M} is introduced:

$$\mathcal{M} = \mathcal{R} \begin{pmatrix} \tilde{\lambda}_1 & 0 & 0 \\ 0 & \tilde{\lambda}_2 & 0 \\ 0 & 0 & \tilde{\lambda}_3 \end{pmatrix} \mathcal{R}^{-1} \quad (5)$$

where

$$\tilde{\lambda}_i = \min \left(\max \left(|\lambda_i|, \frac{c\mathcal{E}}{h_{\max}^2} \right), \frac{c\mathcal{E}}{h_{\min}^2} \right)$$

with h_{\min} and h_{\max} being the minimal and maximal lengths allowed on the mesh's edge and \mathcal{E} the interpolation error allowed in the mesh. Now, if a mesh generation procedure (described above) is used, to generate a mesh with edges close to the unit length in the metric $\mathcal{M}/(c\mathcal{E})$, the interpolation error \mathcal{E} is equi-distributed over the edges a_i of the mesh. More precisely, we have

$$\frac{1}{c\mathcal{E}} a_i^T \mathcal{M} a_i = 1 \quad (6)$$

Remark

If an isotropic mesh is generated, then the edge lengths are the same in all directions. So, an isotropic metric is obtained by taking:

$$\lambda = \min_{i=1,3} (\tilde{\lambda}_i)$$

and hence,

$$\mathcal{M} = \mathcal{R} \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix} \mathcal{R}^{-1} \quad (7)$$

Nevertheless, this definition is not sufficient for a suitable metric in the cases where systems, boundary layers and multi-scales phenomena are considered. Details on the elements used in the metric definition for such cases can be found in Reference [15].

3.2. Implementation of the metric evaluation

A key point in the metric concern the Hessian matrix, hence the second derivatives of the computed solution (which, actually, is used instead of u the unknown exact solution). But, here the solver is a P^1 finite element solver. Therefore, an approximation is used

$$\frac{\partial^2 u_h}{\partial x_i \partial x_j} (S_k) = \frac{- \int_{\Omega} \frac{\partial u_h}{\partial x_j} \frac{\partial \varphi}{\partial x_i}}{\int_{\Omega} \varphi} \quad (8)$$

with φ the finite element shape function with value 1 at the node S_k and 0 elsewhere. We have to specify the physical field u used in such a construction. The previous formula does not lead to a suitable metric definition along boundaries as the gradients are not correctly evaluated, more details can be found in Reference [15].

3.3. Metric intersection in time

For time-dependent problems, a new constraint in the mesh generation procedure is introduced. As in the time frame the phenomenon evolves in a defined area, if a suitable solution is wanted, all the area has to be refined. Thus, the metric associated to the computed solution has to be used to refine where the phenomenon will arrive, and the metric associated to the initial solution of the time frame (i.e. the final metric of the previous iteration) to refine where the phenomenon is at the beginning of the iteration. To do so, a metric intersection in time is introduced, for each variable, between the metrics defined by this variable at two consecutive iterations:

$$\mathcal{M}_i^j = \bar{\mathcal{M}}_i^0 \cap \hat{\mathcal{M}}_i^j \quad (9)$$

where

- $\bar{\mathcal{M}}_i^0$ is the initial metric interpolated on the current mesh \mathcal{H}_i^j , with $\mathcal{M}_i^0 = \mathcal{M}_{i-1}^{j_{\max}}$;
- $\hat{\mathcal{M}}_i^j$ is the metric given by the solution \mathcal{G}_i^j computed on the current mesh \mathcal{H}_i^j ;
- \mathcal{M}_i^j is the metric which is used to generate the new mesh \mathcal{H}_i^{j+1} .

But, if the time frame is too large, a coarsening will appear in the area between the two refinement. To avoid this problem, intermediate solutions have to be taken into account and each intermediate metric defined by each solution are intersected in time to define the final metric.

Thus, the time Δt between two adaptations becomes a new parameter. If a large value of Δt is set, the number of adaptations will be reduced, although the number of metric intersections will be larger as well as the mesh size. On the contrary, decreasing the value of Δt will result in more adaptations and less metric intersections, and the mesh size will be also smaller.

4. MESH ADAPTATION FOR UNSTEADY SIMULATIONS

In this section, mesh adaptation algorithms are presented. In the first part, we describe the classical mesh adaptation algorithm, the so-called ‘stationary fixed point-based adaptation algorithm’ which is suitable for steady simulations but not for unsteady simulations. In the second part, we present the new mesh adaptation algorithm specifically modified for unsteady configurations, the so-called ‘transient fixed point-based adaptation algorithm’.

The following notations are used: t the non-dimensional time and T_{\max} the maximal non-dimensional time. We denote, respectively, by \mathcal{H} , \mathcal{S} and \mathcal{M} the mesh, the solution and the metric.

4.1. Classical mesh adaptation algorithm

The idea of the classical mesh adaptation method for steady simulations consists in searching a fixed point for the couple (Mesh, Solution). In other words, we want to converge toward the stationary solution of the problem, as well as, to some extent, to converge toward its associated fixed mesh (i.e. the unit mesh with respect to the previously defined metric).

It is a ‘one loop’ algorithm where each iteration starts with an initial pair formed by the mesh and the solution computed at the previous iteration (or supplied at the first iteration). By means of the flow solver, after interpolation of the previous solution on the current mesh, the new solution is computed and a new mesh is created based on the metric and associated with this new solution. The iterative process is pursued until the convergence is achieved.

In time-dependent simulations, it is not always possible to converge toward a stationary solution. So, this algorithm is not adapted for unsteady simulation because the mesh used to compute the solution is associated with the previous solution in time and it is not suitable for the computation, the phenomenon goes out of the refined area. In other words, the mesh is late over the solution.

For this algorithm, we denote by N_{adap} the number of adaptations. Hence, at each iteration when the solution is computed we advance in time by $\Delta t = T_{\text{max}}/N_{\text{adap}}$. The classical mesh adaptation algorithm reads:

- Initially at $t=0$, we have: $\mathcal{H}_0, \bar{\mathcal{I}}_0, N_{\text{adap}}, T_{\text{max}}, \Delta t, i=0$,
- Adaptation loop:
 - While** ($i < N_{\text{adap}} \Leftrightarrow (t < T_{\text{max}})$) **Do**
 1. **If** $i=0$ **Then**
 - (a) compute the solution over this mesh, advance in time by Δt :
 $(\mathcal{H}_0, \bar{\mathcal{I}}_0) \rightarrow \mathcal{S}_0$,
 2. **Elseif** $i > 0$ **Then**
 - (a) compute the metric:
 $(\mathcal{H}_i, \mathcal{I}_i) \rightarrow M_i$,
 - (b) generate the new mesh using the metric:
 $(\mathcal{H}_i, M_i) \rightarrow \mathcal{H}_{i+1}$,
 - (c) interpolate the previous solution over the new mesh:
 $(\mathcal{H}_i, \mathcal{I}_i, \mathcal{H}_{i+1}) \rightarrow \bar{\mathcal{I}}_{i+1}$,
 - (d) compute the solution over this new mesh, advance in time by Δt :
 $(\mathcal{H}_{i+1}, \bar{\mathcal{I}}_{i+1}) \rightarrow \mathcal{S}_{i+1}$,
 - Endif**
 - $i = i + 1$

Done.

4.2. Mesh adaptation algorithm for unsteady simulations

The aim here is to obtain a mesh adapted in all the area where the solution evolves. Therefore, the idea is to predict the phenomenon.

Consequently, a transient fixed point problem is considered for the couple (mesh, solution). Inside the main adaptation loop, a new loop is introduced in which the transient fixed point problem is solved. This internal loop is slightly different from the classical one. At each internal iteration (from time t to time $t + \Delta t$), after generating the new mesh, instead of keeping the last solution (at time $t + \Delta t$) and continuing the computation, the computation is restarted with the initial solution (at time t) interpolated on the new adapted mesh. The inner process is repeated until the convergence is achieved for the transient fixed point solution (at time $t + \Delta t$). When the convergence is reached, we advance to the next period and the final

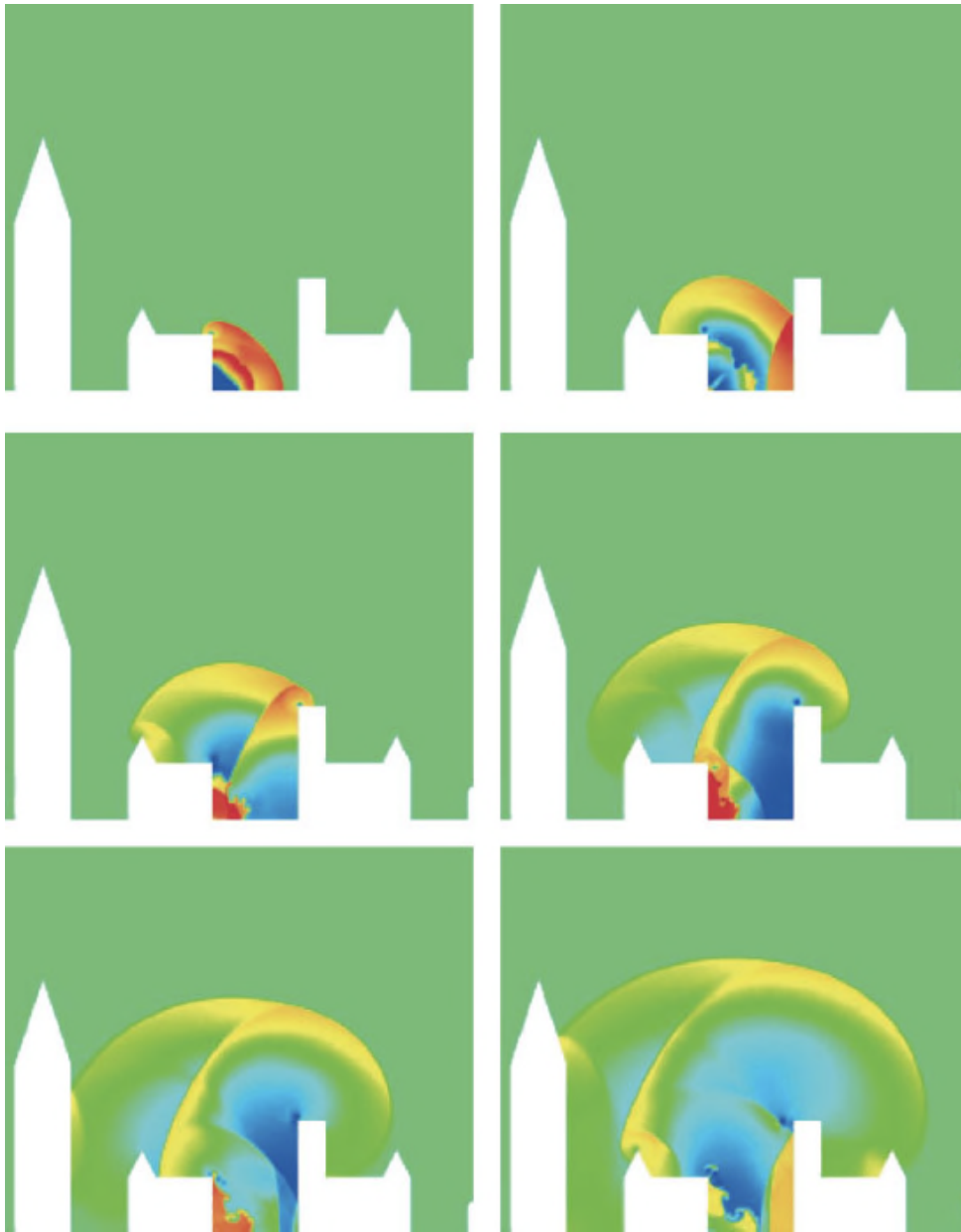


Plate 1. Evolution of the explosion in the 2D geometry. Snapshots of the density value at time 0.02, 0.04, 0.06, 0.08, 0.1 and 0.12.

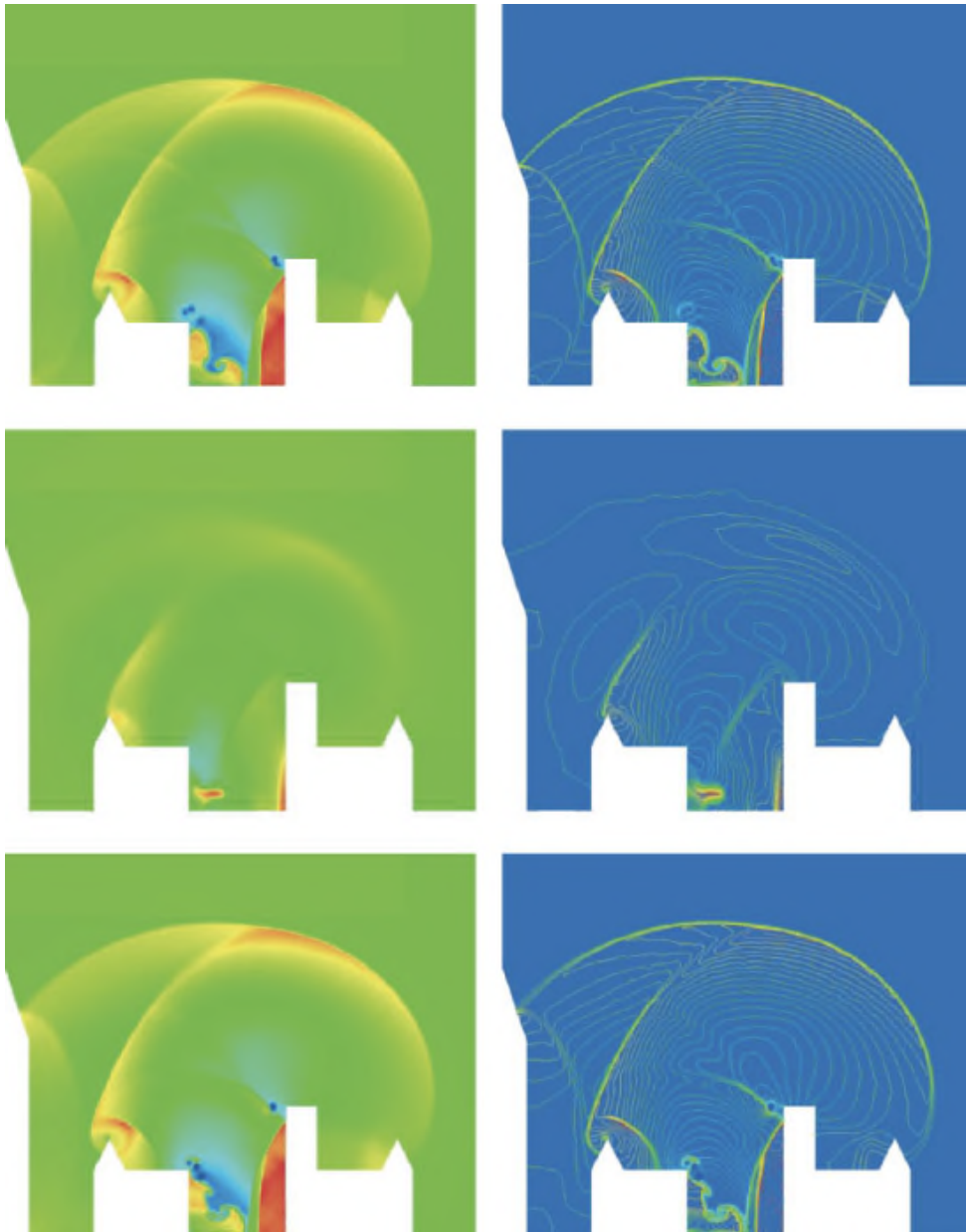


Plate 2. A snapshot of the density value at time 0.12. Up, the reference solution with a uniform mesh including 1 206 946 vertices. Middle, the solution computed with the classical adaptation (4376 vertices). Down, the solution computed with the transient fixed point mesh adaptation (53 487 vertices).

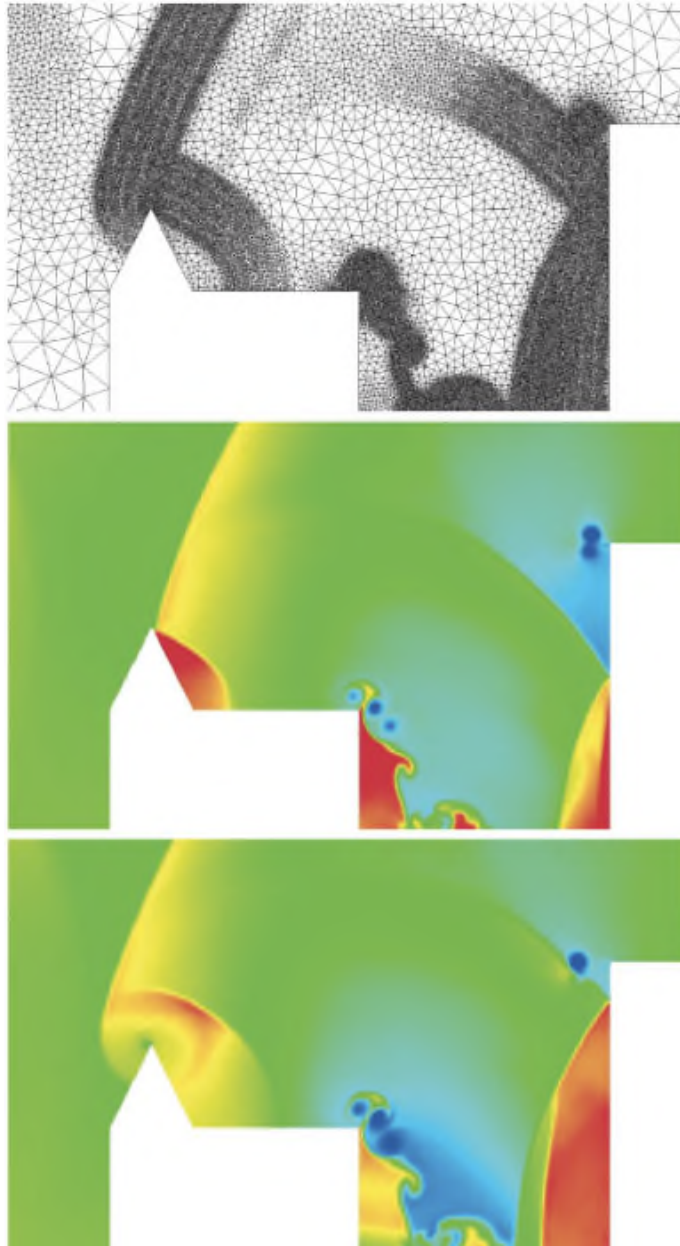


Plate 3. The evolution at the last period of the transient fixed point based mesh adaptation. Up, the last mesh obtained with the transient fixed point based mesh adaptation. Middle, density distribution at time 0.11, this is the initial solution of the period. Down, density distribution at time 0.12, this is the final solution of the period computed on the mesh. We note that all the areas where shock waves evolve are refined thanks to the metric intersection in time (five metrics are intersected).

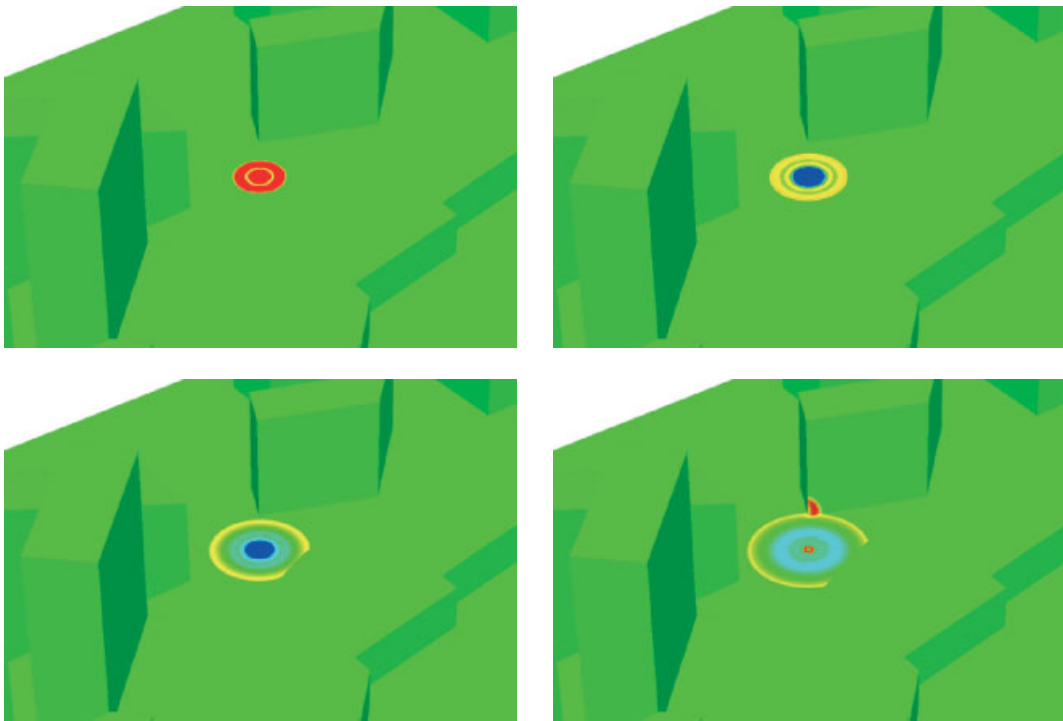


Plate 4. Evolution of the explosion in the 3D geometry. Snapshots of the density value at time 0.004, 0.008, 0.012 and 0.016.

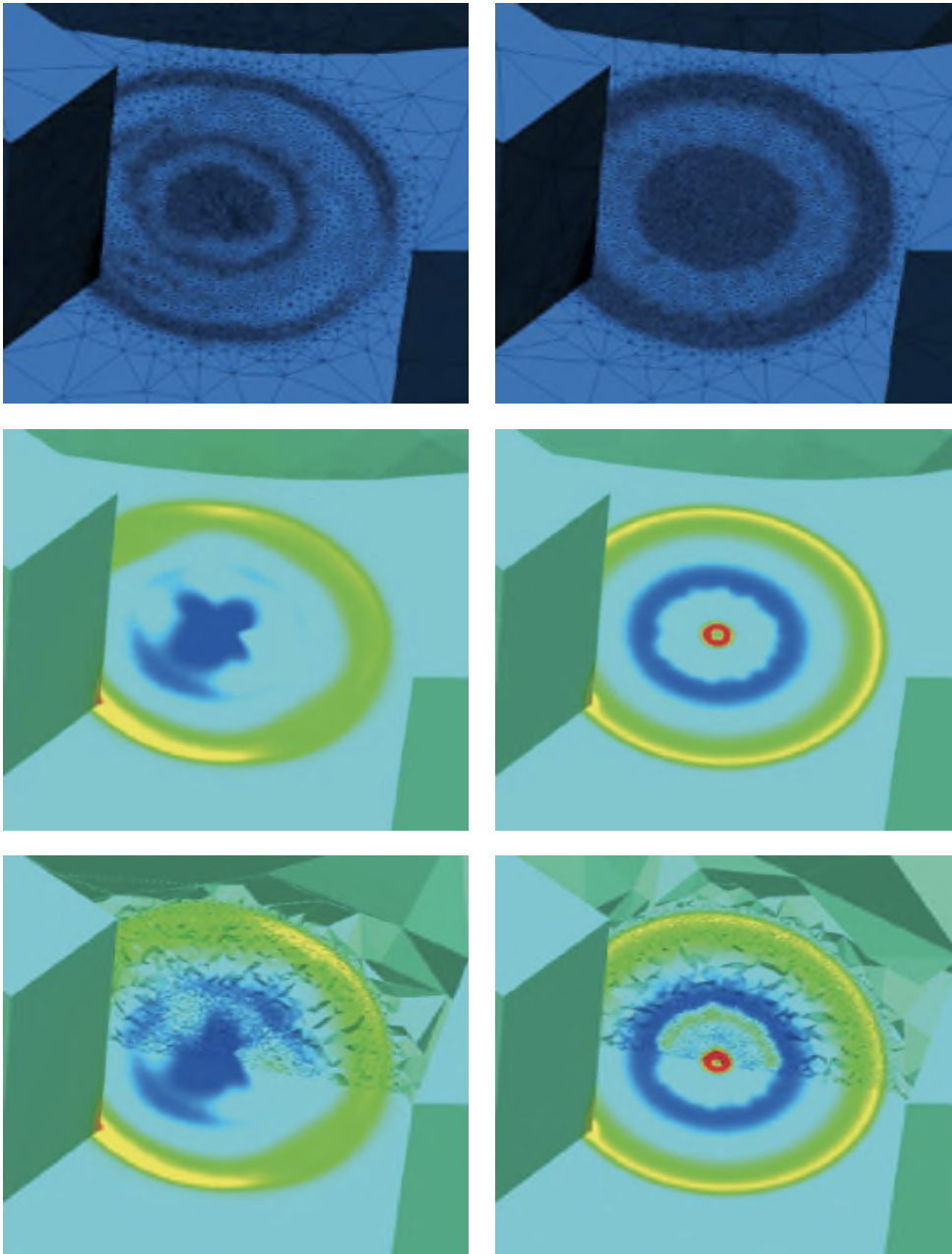


Plate 5. Comparison at time 0.016 of the classical mesh adaptation on the left and the transient fixed point based mesh adaptation on the right. Up, surface adapted mesh for the two methods (for the classical method the complete mesh has 309 972 vertices and for the transient method 583 106 vertices). Middle, density distribution on the surface. Down, density distribution inside the volume and on the surface.

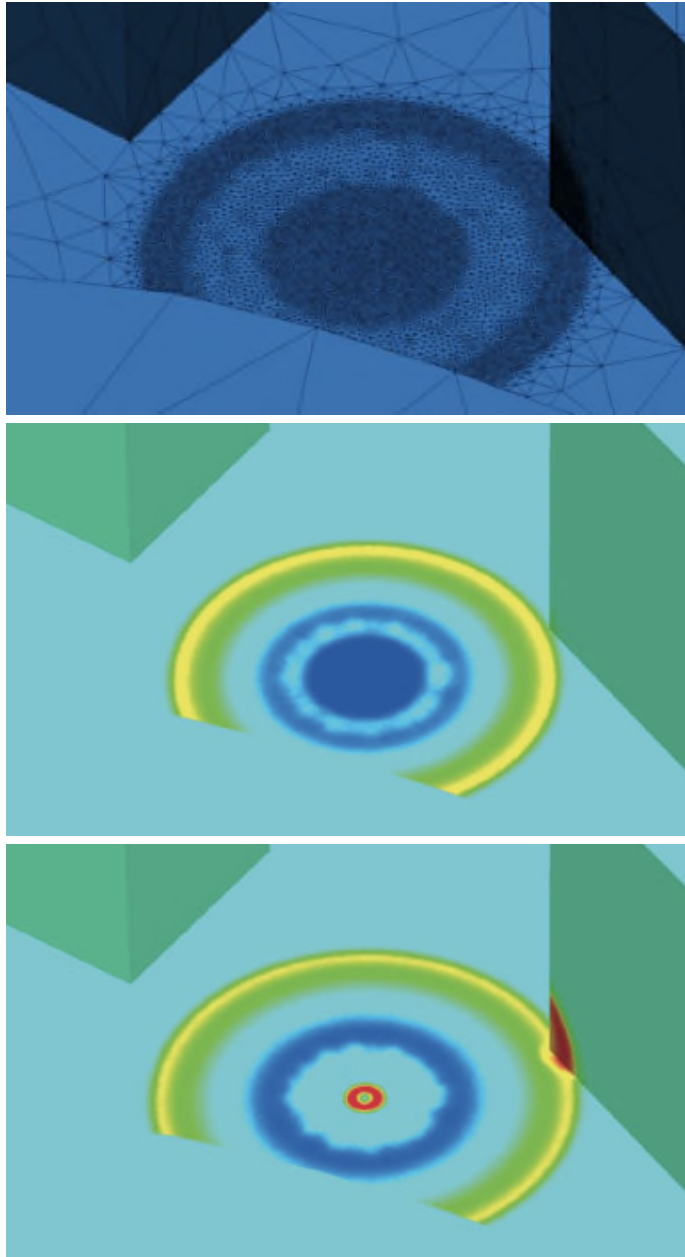


Plate 6. The evolution at the last period of the transient fixed point based mesh adaptation. Up, the last mesh obtained with the transient fixed point based mesh adaptation. Middle, density distribution at time 0.014, this is the initial solution of the period. Down, density distribution at time 0.016, this is the final solution of the period computed on the mesh. We note that all the areas where shock waves evolve are refined thanks to the metric intersection in time (two metrics are intersected).

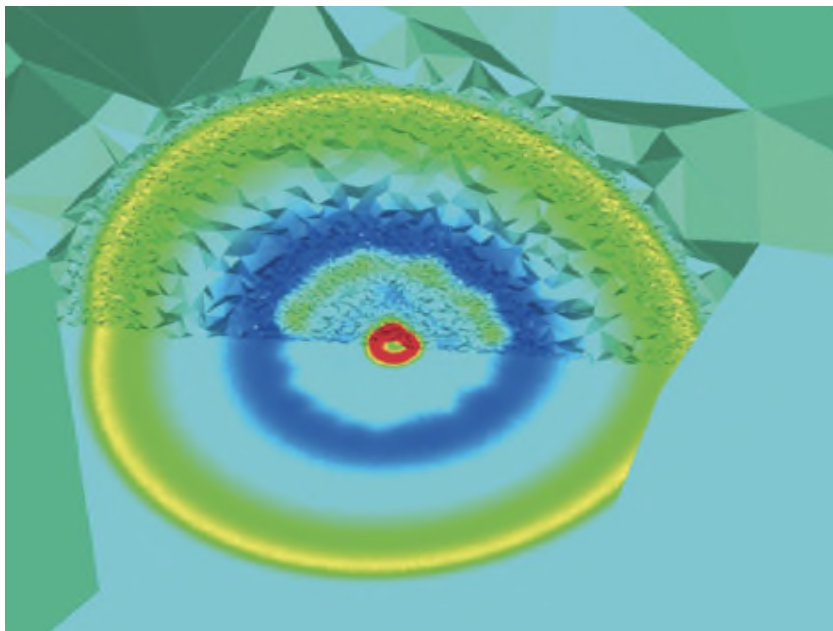
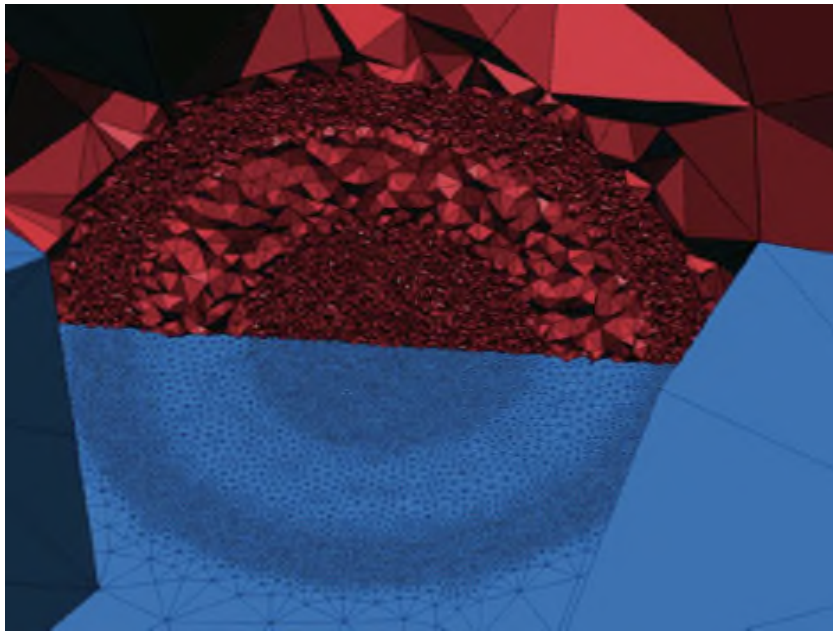


Plate 7. A cut plane in the volume to show mesh and density distribution inside the volume. Up, surface and volume adapted mesh at time 0.016 (583 106 vertices). Down, density distribution on the surface and inside the volume at time 0.016. Note that the volume mesh is also refined in all the area where the shock wave evolves.

solution becomes the new initial solution (at time $t + \Delta t$) and we restart the internal loop iterations.

To summarize, in the internal loop after each iteration all the information obtained is kept for the mesh and the computation is restarted.

Finally, a tolerance in the internal loop is introduced: the process stops when the desired accuracy for the solution is reached.

More precisely, when: $\|(\partial u_h / \partial t)_i^{j+1} - (\partial u_h / \partial t)_i^j\|_{L^2} \leq \text{TOL}$, where i (resp. j) denotes i th (resp. j th) iteration of the principal (resp. internal) loop, the internal loop is over.

For this algorithm we denote by N_{period} the number of computed periods, i.e. the number of iterations of the principal loop. Hence, at each period when the solution is computed we advance in time by $\Delta t = T_{\text{max}} / N_{\text{period}}$. We also denote by N_{ptfx} the number of the internal fixed point iterations and by \mathcal{E}_{tol} the wanted tolerance, before leaving the internal loop. The modified mesh adaptation algorithm reads:

- Initially at $t = 0$, we have: $\mathcal{H}_0, \mathcal{S}_0, \mathcal{M}_0, N_{\text{period}}, N_{\text{ptfx}}, \Delta t, \mathcal{E}_{\text{tol}}, i = 0$,
- Loop on the number of periods:
 - While** ($i < N_{\text{period}} \Leftrightarrow (t < T_{\text{max}})$) **Do**
 1. $t_0 = i \Delta t, j = 0$,
 2. Internal loop:
 - While** ($j < N_{\text{ptfx}}$ or $\text{TOL} > \mathcal{E}_{\text{tol}}$) **Do**
 - (a) **If** $j = 0$ **Then**
 - i. compute the solution over this mesh, we advance in time by Δt :
 $(\mathcal{H}_i^0, \mathcal{S}_i^0) \rightarrow (\mathcal{H}_i^1, \mathcal{S}_i^1)$,
 - (b) **Elseif** $j > 0$ **Then**
 - i. compute the metric:
 $(\mathcal{H}_i^j, \mathcal{S}_i^j) \rightarrow \hat{\mathcal{M}}_i^j$,
 - ii. compute the intersected metric over time:
 $(\bar{\mathcal{M}}_i^0, \hat{\mathcal{M}}_i^j) \rightarrow \mathcal{M}_i^j$,
 - iii. generate the new mesh using the time intersected metric:
 $(\mathcal{H}_i^j, \mathcal{M}_i^j) \rightarrow \mathcal{H}_i^{j+1}$,
 - iv. interpolate the solution at t_0 over the new mesh:
 $(\mathcal{H}_i^0, \mathcal{S}_i^0, \mathcal{H}_i^{j+1}) \rightarrow \bar{\mathcal{S}}_i^{j+1}$,
 - v. interpolate the metric at t_0 over the new mesh for the next time intersection:
 $(\mathcal{H}_i^0, \mathcal{M}_i^0, \mathcal{H}_i^{j+1}) \rightarrow \bar{\mathcal{M}}_i^0$,
 - vi. compute the solution over this mesh, advance in time by Δt :
 $(\mathcal{H}_i^{j+1}, \bar{\mathcal{S}}_i^{j+1}) \rightarrow \mathcal{S}_i^{j+1}$,
 - Endif**
 - $j = j + 1$
 - Done**
 3. compute the metric associated to the final solution:
 $(\mathcal{H}_i^{N_{\text{ptfx}}}, \mathcal{S}_i^{N_{\text{ptfx}}}) \rightarrow \mathcal{M}_i^{N_{\text{ptfx}}}$,
 4. redefine the initial variable for the next internal loop:
 $(\mathcal{H}_i^{N_{\text{ptfx}}}, \mathcal{S}_i^{N_{\text{ptfx}}}, \mathcal{M}_i^{N_{\text{ptfx}}}) \rightarrow (\mathcal{H}_{i+1}^0, \mathcal{S}_{i+1}^0, \mathcal{M}_{i+1}^0)$
 $i = i + 1$
- Done.**

5. NUMERICAL RESULTS

In this section, we present 2D and 3D results for a generalized Riemann problem. We follow the evolution of a non-linear wave propagation on a complex geometry. An isotropic metric is used for both surface and volume meshing. Several software codes, developed at INRIA-Rocquencourt, have been involved in these computations:

- *Flow solvers*: The flow solvers NSC2KE and NSC3KE [15, 16], based on finite volume—finite element, work on unstructured meshes for compressible Euler and Navier–Stokes equations in conservation form.
- *Mesh generators*: The adaptive mesh generators YAMS (2D and surface) and GAMHIC3D (volume) have been used [8, 10]. The flowcharts of the mesh adaptation algorithms are illustrated in Figure 1.

5.1. Non-linear wave propagation on 2D complex geometry

We consider here a non-linear wave propagation on a complex geometry, that can be seen as a generalization of the 1D shock tube case to higher space dimension. An initial Dirac perturbation is introduced in an uniform field to simulate an explosion (see Plate 1). For this simulation, all lengths are given in metres and the time unit is in seconds. For all computations a Courant number of 0.4 is taken.

The aim is to show the behaviour of the transient adaptation on a complex geometry. We compare the adaptive transient solution to the reference solution computed on a uniformly fine mesh. This is achieved by comparing the two solutions obtained at time 0.12 s. We also see, that the stationary fixed point-based adaptation is not suitable for such simulations.

To this end, a unstructured uniform mesh with a mesh size close to $h = 10^{-1}$ (1 206 946 nodes) is considered, in order to compute the reference solution. For both mesh adaptation methods, the metric is scalar (isotropic) and evaluated with only one variable: the density. We start the computation with a coarse uniform unstructured mesh (3131 nodes). To compare the solution the minimal mesh size has to be identical for all methods, so the metric parameters for the computation are: $h_{\min} = 10^{-1}$, $h_{\max} = 20$ and $\mathcal{E} = 2 \times 10^{-3}$, and a mesh

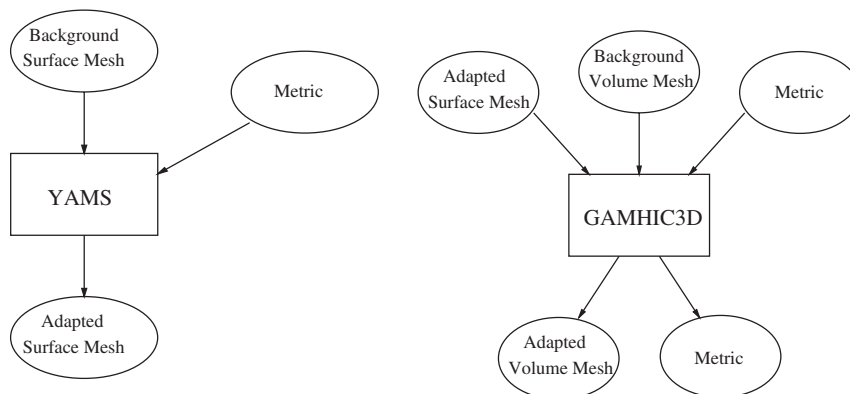


Figure 1. Flowcharts of the mesh adaptation algorithms for surface (left-hand side) and volume (right-hand side) meshing.

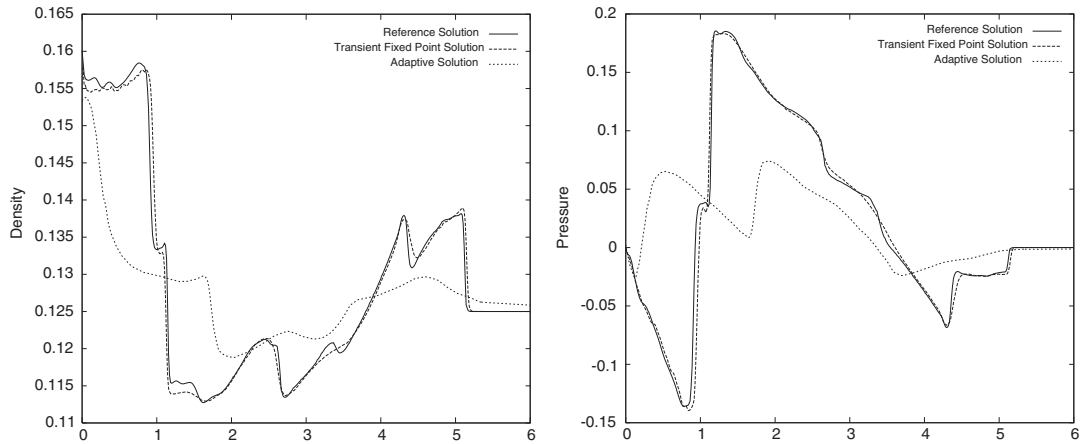


Figure 2. Cross-section of the density and pressure value at time 0.12 on the line $y = -2.5x + 15$.

gradation of 3 is used. For the transient adaptation the other parameters are: $N_{\text{period}} = 12$, $N_{\text{ptfx}} = 4$, $\mathcal{E}_{\text{tol}} = 10^{-4}$. In both cases, 48 adaptations have been requested, ($N_{\text{adap}} = 48$ in the classical method, it represents obviously an adaptation every 50 iterations). This is to have the same complexity for both approaches. Furthermore, as $\Delta t = 15 \times 10^{-3}$ was taken for the transient computation, five intermediate solutions have been used for the metric intersection in time.

Subsequently, the results of the two mesh adaptation methods with the reference solution are compared. As we based the metric on the density, we show in Plate 2 the density isovalues comparison at non-dimensional time 0.12. The stationary fixed point-based adaptation poorly captures the phenomenon whereas the transient fixed point-based adaptation method is close to the reference solution. In Figure 2, a cross-section of the density and the pressure, respectively, at time 0.12 confirms the previous conclusion. The classical method poorly captures the solution because shock waves always go out of refinement so the solution vanishes, and as the solution is not accurately computed, the resulting mesh refinement will be inappropriate in critical regions. This behaviour is obviously getting worse as the computation advances in time. This is also illustrated in Figure 3 where the final meshes obtained by both methods are shown. The mesh could be adapted more frequently to reduce the acuity of the problem, although the mesh will never be adapted to the final solution of each iteration.

Plate 3 shows the impact of the metric intersection in time. In the final mesh of the computation with the transient fixed point algorithm, all the areas where shock waves evolve have been finely refined by using five intermediate solutions for the intersection.

The adaptive transient solution is close to the reference solution, with only 53 487 nodes (almost 22 times fewer than the reference mesh with 1 206 946 nodes!). The global computation time is about 20 times less with the adaptation approach, see Table I.

Remark

For the stationary fixed point-based adaptation method the final mesh has 4376 nodes and the computing time is 0.2 percent of the reference time, but the phenomenon is not captured!

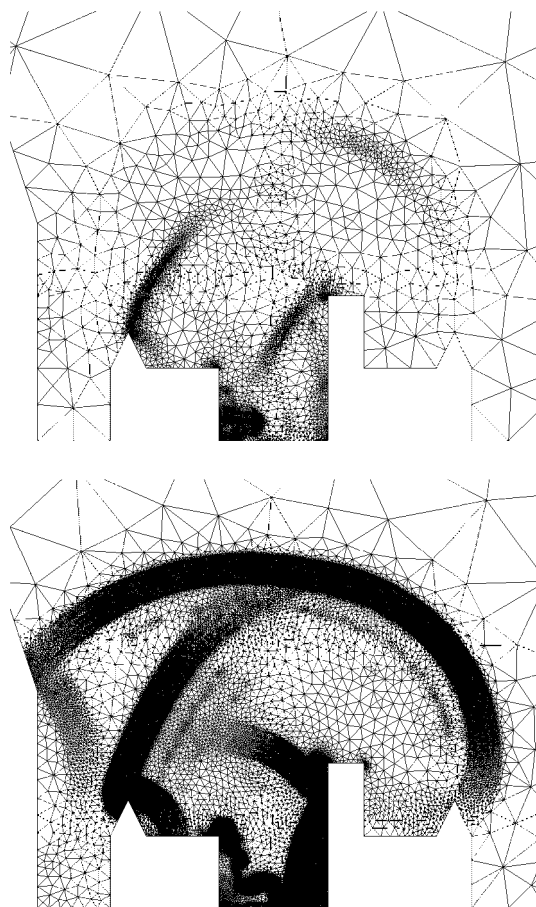


Figure 3. Final adapted mesh. Up, the final mesh with the classical mesh adaptation including 4376 vertices. Down, the final mesh with the transient fixed point mesh adaptation including 53 487 vertices.

Table I. CPU time and the number of nodes of the final mesh in 2D for each method computed on a hp 9000 work station PA 8600–550 MHz.

	CPU time	Number of nodes
Uniformly fine mesh	166 h	1 206 946
Classical mesh adaptation method	17 m	4376
Transient fixed-point mesh adaptation method	8 h 40 m	53 487

5.2. Non-linear wave propagation on 3D complex geometry

We consider here the extension of the previous simulation to 3D (Plate 4). As before, all lengths are given in metres and the time unit is in seconds. For all computations a Courant number of 0.6 is taken.

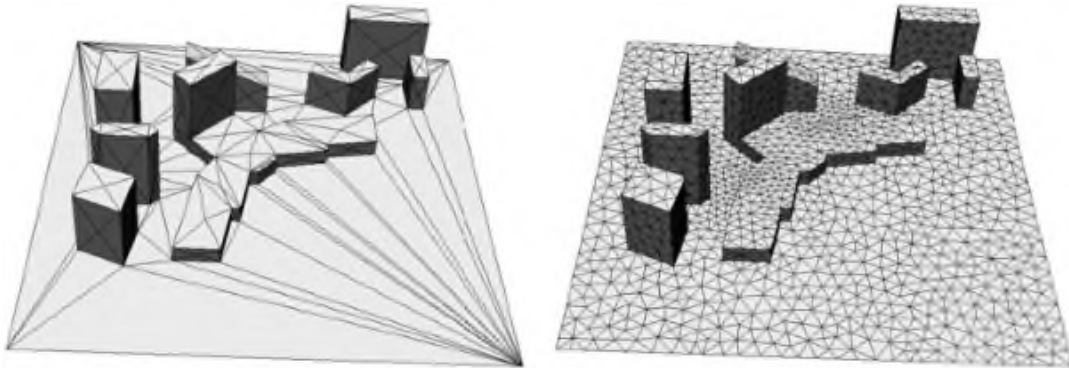


Figure 4. 3D geometry and initial surface mesh for the computation (4302 vertices for the surface mesh and 16467 vertices for the complete mesh).

Here the aim, is to compare the stationary fixed point-based adaptation with the new algorithm and show the adaptation of the volume. We compare the two solutions obtained at time 0.016s. In this simulation, the adaptive strategy has a greater impact than in 2D. Indeed, a reference mesh with a edge length of $h = h_{\min} = 10^{-1}$ everywhere in our geometry (see Figure 4) would require for the surface mesh slightly more than 6.5 millions vertices (and so the complete mesh will need more than 2.5 billions vertices).

For both mesh adaptation methods, the metric is isotropic and evaluated with the density variable. We start the computation with a coarse uniform unstructured mesh (16467 nodes, Figure 4). To compare the solution, each method have the same metric parameters for the computation: $h_{\min} = 10^{-1}$, $h_{\max} = 15$ and $\mathcal{E} = 10^{-2}$, and a mesh gradation of 3 is used. For the transient adaptation the other parameters are: $N_{\text{period}} = 8$, $N_{\text{pt,fx}} = 4$, $\mathcal{E}_{\text{tol}} = 10^{-4}$ and for the classical one $N_{\text{adap}} = 32$ (it represents obviously an adaptation every 15 iterations). Here, as $\Delta t = 2 \times 10^{-3}$ is small, for the metric intersection in time only two intermediate solutions have been used (i.e. the initial and the final solution of the time frame).

As in the 2D case, in Plate 5 we show that the shock wave was diffused with the classical mesh adaptation, whereas it was not with the transient fixed point-based mesh adaptation. Therefore, in the classical approach, the mesh is not well adapted, as emphasized by the shock wave regions in which unexpected coarsening takes place. We come to the same conclusion in the volume.

Plate 7 shows in a cut plane the adapted volume mesh and the density distribution in the volume. The volume mesh is clearly adapted to the solution and the surface and volume mesh element sizes are closely related. Finally, Plate 6 illustrate the metric intersection in time, all the area where the shock wave evolves is refined and it is the same in the volume, Plate 7.

Remark

The transient fixed point adapted final mesh at time 0.016 has 583 106 vertices and for the classical one 309 972 vertices (see Table II).

Table II. CPU time and the number of nodes of the final mesh in 3D for each method computed on a hp 9000 work station PA 8600–550 MHz.

	CPU time	Number of nodes
Uniformly fine mesh	?	Not accessible
Classical mesh adaptation method	17 h	309 972
Transient fixed-point mesh adaptation method	55 h	583 106

^{||} As, the complete mesh will need more than 2.5 billions vertices, this is exceeding our memory capacity.

6. CONCLUSIONS

A new adaptive algorithm for time-dependent simulations has been described and applied on a 2D and 3D complex geometries. According to the numerical results obtained on such cases, this approach appears to be more appropriate than the classical adaptive scheme to solve time-dependent problem.

Regarding the results obtained so far, the introduction of anisotropic unstructured mesh generation techniques in 3D seems very promising. The objective is potentially to reduce the number of nodes. This point would be made clear at the time an anisotropic mesh generation method (work currently in progress at INRIA) will be available. Moreover, a better definition of the metric for 3D cases is needed, notably to capture weak shock (in the 2D exemple, weak shocks were not precisely captured) as well as a better implementation of the Hessian estimation, in order to obtain a more accurate metric. Also, the theoretical background of the transient fixed point mesh adaptation needs further investigation.

REFERENCES

1. Borouchaki H, Castro-Diaz MJ, George PL, Hecht F, Mohammadi B. Anisotropic adaptive mesh generation in two dimensions for CFD. *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, 1996.
2. Castro-Diaz M, Hecht F, Mohammadi B. Anisotropic grid adaption for inviscid and viscous flows simulations. *International Journal for Numerical Methods in Fluids* 2000; **25**:475–491.
3. Hecht F, Mohammadi B. Mesh adaptation by metric control for multi-scale phenomena and turbulence. *AIAA*, paper 97-0859, 1997.
4. Löhner R, Baum JD. Adaptive h-refinement on 3D unstructured grids for transient problems. *International Journal for Numerical Methods in Fluids* 1992; **14**:1407–1419.
5. Pain CC, Umpelby AP, de Oliveira CRE, Goddard AJH. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**:3771–3796.
6. Rausch RD, Batina JT, Yang HTY. Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations. *AIAA Journal* 1992; **30**:1243–1251.
7. Frey PJ, George PL. *Mesh Generation, Application to Finite Elements*. Hermès Science: Oxford, Paris, 2000.
8. Frey PJ, Borouchaki H. Geometric surface mesh generation. *Computing and Visualization in Science* 1998; **1**:113–121.
9. George PL, Borouchaki H. *Delaunay Triangulation and Meshing, Application to Finite Element*. Hermès Science: Paris, 1998.
10. George PL, Borouchaki H, Laug P. An efficient algorithm for 3D adaptive meshing. *Advances in Engineering Software* 2002; **33**:377–387.
11. Borouchaki H, Hecht F, Frey PJ. Mesh gradation control. *International Journal for Numerical Methods in Engineering* 1997; **43**:1143–1165.
12. Watson DF. Computing the n -dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal* 1981; **24**:167–172.
13. Lo SH. Automatic mesh generation and adaptation by using contours. *International Journal for Numerical Methods in Engineering* 1991; **31**:689–707.

14. Ciarlet PG. Basic error estimates for elliptic problems. In Ciarlet PG, Lions JL (eds), *Handbook of Numerical Analysis*, vol. II. North Holland: Amsterdam, 1991.
15. Mohammadi B, George PL, Hecht F, Saltel E. 3D Mesh adaptation by metric control for CFD. *Revue Européenne des Éléments Finis* 2000; **9**(4):439–449.
16. Mohammadi B. *NSC2KE, a User-Guide*. RT INRIA 164, 1994.
17. Pichelin É, Fortin M, Boivin S. Étude numérique d'estimations d'erreur a posteriori. *Revue Européenne des Éléments Finis* 2000; **9**(4):467–486.