Multi-Scale Anisotropic Mesh Adaptation for Time-Dependent Problems

Frédéric Alauzet, Adrien Loseille, Géraldine Olivier

Multi-Scale Anisotropic Mesh Adaptation for Time-Dependent Problems

Frédéric Alauzet, Adrien Loseille and Géraldine Olivier

RESEARCH REPORT
N° 8929
June 2016
Project-Team Gamma3

ISSN 0249-6399 ISRN INRIA/RR--8929--FR+ENG

Multi-Scale Anisotropic Mesh Adaptation for Time-Dependent Problems

Frédéric Alauzet*, Adrien Loseille and Géraldine Olivier

Project-Team Gamma3

Research Report n° 8929 — June 2016 — 39 pages

Abstract: This paper deals with anisotropic mesh adaptation applied to unsteady inviscid CFD simulations. Anisotropic metric-based mesh adaptation is an efficient strategy to reduce the extensive CPU time currently required by time-dependent simulations from the perspective of performing this kind of computations on a daily basis in an industrial context. In this work, we detail the time-accurate extension of multi-scale anisotropic mesh adaptation for steady flows [26], *i.e.,* a control of the interpolation error in L^p norm, to unsteady flows based on a space-time error analysis and an enhanced version of the fixed-point algorithm [2]. We also show that each stage - remeshing, metric field computation, solution transfer, and flow solution - is important to design an efficient time-accurate anisotropic mesh adaptation process. The parallelization of the whole mesh adaptation platform is also discussed. The efficiency of the approach is emphasized on three-dimensional problems with convergence analysis and CPU data.

Key-words: Anisotropic Mesh Adaptation, Time-accurate, Multi-scale, Metric, Fixed-Point Algorithm, Unsteady Flows

* INRIA Saclay Ile-de-France, Projet Gamma3, 1, rue Honoré d'Estienne d'Orves, 91126 Palaiseau, France. email: frederic.alauzet@inria.fr

RESEARCH CENTRE SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Multi-Scale Anisotropic Mesh Adaptation for Time-Dependent Problems

Résumé : Ce document présente une méthode d'adaptation de maillage anisotrope pour les problèmes instationnaires.

Mots-clés : Adaptation de maillage anisotrope, précis en temps, multi-échelle, métrique, algorithme de point fixe, écoulements instationnaires

Contents

1 Introduction

The 3D simulation of unsteady flows for complex geometries, which is actually the typical situation for real-life problems, still remains a challenge. Indeed, these computations are very time consuming. The use of anisotropic metric-based mesh adaptation, which has already proven its efficiency for steady problems [4, 6, 21, 34, 35, 40, 42], seems to be appropriate in order to reduce the CPU time of such simulations while preserving their accuracy. However, extension to the unsteady case is far from straightforward as such simulations have many difficulties arising from unsteadiness. To achieve an efficient time-accurate mesh adaptation scheme for unsteady flows, we have to overcome the issues described in the following discussion.

First of all, we have to remedy the problem of the latency of the mesh with respect to the solution, *i.e.,* the mesh is lagging behind the solution in time. Indeed, if a mesh is adapted for a solution at time t, once the solution progress in time again, it is clear that this mesh is not adapted for the next time-steps. Consequently, no error control can be guaranteed.

Theoretically, error estimates used for steady flows [4, 26] need to be extended to the unsteady case. In other words, for time-dependent simulations, the temporal error should also be controlled. A space-time error analysis is then required.

At each remeshing, the solution has to be interpolated on the new adapted mesh. This stage becomes crucial in the context of unsteady problems and even more if a large number of transfers is performed, as the error due to this solution transfer accumulates throughout the simulation. Therefore, the error introduced by this stage can spoil the overall accuracy of the solution [5].

In regards to the meshing phase, a difficulty arises from the definition of the solver time-step dt (omitting CFL factor) which is homogeneous to the smallest element height of the mesh h_{min}. Consequently, a single small-height element in the whole mesh is sufficient to considerably reduce the time-step and thus increase the CPU time of the simulation. This is a serious problem, especially in the context of highly anisotropic mesh adaptation, which involves highly stretched elements. The only remedy is to reduce this constraint as much as possible by generating anisotropic meshes controlling the highest h_{min} value [30]. This implies a substantial effort on the anisotropic mesh generator as the quality of the mesh must be very good. For example, if the mesh generator fails to generate the minimal height element for which a size of h_{target} had been prescribed, and instead build an element of height $h_{min} = 0.01 \times h_{target}$, the number of solver iterations is multiplied by 100 for time-accurate simulations. Consequently, for a given mesh, *not a single meshing mistake is allowed despite millions of tetrahedra are generated* in order to obtain a coherent time-step and to not increase drastically the CPU time of the simulation.

State-of-the-art

Over the past few years, a rather large number of papers have been published dealing with mesh adaptation for *steady* numerical simulations, whereas only a small number have addressed *time-dependent* problems. For the unsteady case, three different approaches can be distinguished:

- the so called h-refinement method that consists of adapting the mesh frequently in order to maintain the solution within refined regions and to introduce a safety area around critical regions [24, 25, 36, 39]. This method is based on coarsening/refinement techniques without node displacement to reduce the solution transfer error. This technique have been used to produce isotropic meshes only

- using an unsteady mesh adaptation algorithm [10, 41]. This method is based on local or global remeshing techniques and the error is estimated every n_1 flow solver iterations. If the error is greater than a prescribed threshold, the mesh is readapted. Therefore, the mesh is adapted every $n_2 > n_1$ iterations with n_2 *a priori* unknown

- and more recently, perform local adaptive remeshing enabling the construction of anisotropic meshes. In this case, the mesh is frequently adapted in order to guarantee that the solution always evolves in refined regions [7, 18, 35, 37]. However, much care must still be paid to the interpolation error, notably if the projection step is performed on the fly after each mesh modification.

All these approaches involve a large number of mesh adaptations and tends to introduce unquantified errors due to the transfer of the solution from the old mesh to the new one. This is particularly a problem for hyperbolic problems [5]. Moreover, the first and the third approaches do not control explicitly the error made on the solution as they perform an arbitrary large and *a priori* number of adaptations. The second one authorizes the error to grow through the simulation. Note that the first approach cannot be extended to the case of anisotropic mesh adaptation. Finally, none of them considers the intrinsic non-linear nature of the mesh adaptation problem: the convergence of the mesh adaptation process is never addressed and therefore the obtention of the optimal mesh cannot be expected. And, none of them takes into account the temporal error in the error analysis.

Our approach

In order to overcome all the problems relative to mesh adaptation for time-dependent simulations stated in the introduction, an innovative strategy based on a transient fixed-point algorithm has been developed in [2]. The main goals were to guarantee a control of the spatial and temporal interpolation errors during the whole simulation and to control (reduce) the number of mesh adaptations in order to master (diminish) the error introduced by the transfers of solutions. This new strategy starts with the observation that direct extension of steady adaptation algorithms to unsteady problems is not appropriate: specific algorithms must be developed that truly take into account the transient nature of the solution. It relies on the assumption that the temporal error is always controlled by the spatial one, which is indeed the case when solving a linear advection problem under a CFL condition [2]. So far, the transient fixed-point mesh adaptation algorithm relies on:

- an isotropic mesh adaptation
- a control of the spatial and temporal interpolation error in L^∞ norm thanks to metric intersection in time
- a *local* transient fixed-point mesh adaptation algorithm to converge the non-linear problem of mesh adaptation.

But, this algorithm is valid only when the space-time interpolation error is controlled in L^∞ norm. Controlling the space-time error in L^p norm is interesting as additional guarantees hold: second of order of convergence of non smooth flows, the avoidance of prescribing a minimal size during the remeshing, and the ability to capture all the scales of the solution [29]. This requires the computation of a global normalization term. Unfortunately, this requirement is not compatible with the previous local approach in which each time sub-interval is adapted in a decoupled manner. Since multi-scale mesh adaptation has now proved its efficiency for steady CFD computations [4, 26], it seems quite relevant to extend the fixed-point algorithm proposed in [2] to this framework. In the following sections, this extension is done by proposing:

- a *global* fixed-point mesh adaptation algorithm to converge the mesh adaptation non-linear problem
- the extension of the multi-scale error estimate [26] to unsteady problems by proposing a L^p space-time error analysis
- a conservative solution transfer operator to considerably diminish the error introduced by this stage of the algorithm
- a high-quality anisotropic local remeshing controlling the heights of the elements to guarantee optimal solver time-steps and to ensure maximal robustness in the meshing process. This issue has been handled with care in [30].

The paper is outlined as follows. After recalling the steady case in Section 2, the space-time error analysis is given in Section 3. Section 4 describes the global fixed-point mesh adaptation algorithm and Section 5 explains how to use the mathematical analysis in practice. The parallelization of the mesh adaptation loop is discussed in Section 6 and numerical examples are provided in the last section.

2 The steady case

Let $\mathcal{H}$ be a mesh of a bounded domain Ω and let V_h be the usual linear finite element space:

$$V_h = \left\{ \phi \in \mathcal{C}^0(\Omega) \;\middle|\; \phi_{|K_i} \text{ is affine for all elements } K_i \in \mathcal{H} \right\}.$$

For a given continuous function u, we denote by $\Pi_h u$ the $\mathbb{P}^1$-interpolant of u on $\mathcal{H}$, which is the element of V_h such that $\Pi_h u(\mathbf{p}_i) = u(\mathbf{p}_i)$ for all vertices $\mathbf{p_i} \in \mathcal{H}$. In this study, we

focus on minimizing in space and in time the interpolation error on a sensor of interest. We first recall the generation of anisotropic adapted meshes based on the prescription of a metric field. Then, the optimality problem of finding the best mesh with respect to the control of the interpolation error in L^p norm is provided and solved analytically.

2.1 Metric-based generation of anisotropic adapted meshes

Metric-based generation of anisotropic adapted meshes uses the notion of Riemannian metric space [16, 27, 28]. For a computational domain $\Omega \subset \mathbb{R}^d$, a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is a spatial field that defines at any point of Ω a metric tensor $\mathcal{M}(\mathbf{x})$, e.g. a $d \times d$ symmetric positive definite matrix. It is then possible for a mesh generator to work, *i.e.,* to evaluate all geometric quantities, in this Riemannian metric space instead of working in the canonical Euclidean space. In a Riemannian metric space, the dot product is defined locally by $\mathcal{M}$: $\langle \mathbf{u}, \mathbf{v}\rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v}\rangle$ for $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}^3 \times \mathbb{R}^3$. Thus, the computation of geometric quantities requires integral formulae to take into account the variation of the metric field. In that case, the length of edge $\mathbf{e} = \mathbf{ab}$ is computed using the straight line parameterization $\gamma(t) = \mathbf{a} + t\,\mathbf{ab}$, where $t \in [0, 1]$:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \|\gamma'(t)\|_{\mathcal{M}}\,\mathrm{d}t = \int_0^1 \sqrt{\mathbf{ab}^T\,\mathcal{M}(\mathbf{a} + t\,\mathbf{ab})\,\mathbf{ab}}\,\mathrm{d}t\,,$$

and the volume of element K is:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,.$$

It is important to note that, in a Riemannian metric space, computing the length of a segment (*i.e.,* an edge) differs from evaluating the distance between the extremities of this segment. Indeed, the distance between two points requires to compute the geodesic between these two points. When the metric field is varying, this path is usually not straight anymore.

The main idea of metric-based mesh adaptation, initially introduced in [17], is to generate *a unit mesh* in the prescribed Riemannian metric space, e.g. a mesh of $\Omega \subset \mathbb{R}^3$ such that each edge has a unit length and each tetrahedron is regular (or equilateral) with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$:

$$\forall \mathbf{e},\quad \ell_{\mathcal{M}}(\mathbf{e}) = 1 \quad \text{and} \quad \forall K,\quad |K|_{\mathcal{M}} = \frac{\sqrt{2}}{12}\,.$$

The resulting mesh in the canonical Euclidean space will be anisotropic and adapted.

2.2 Summary of steady multi-scale anisotropic mesh adaptation

We seek for the best mesh approximating the smooth solution u of a steady scalar PDE over a domain Ω of $\mathbb{R}^d$. In our investigation, we focus on the specification of a mesh that is optimal for the interpolation error. Multi-scale mesh adaptation relies on the minimization of the spatial interpolation error in L^p norm [9, 20]. It is a new approach as compared to the now well-known L^∞ strategy which tends to equi-distribute the spatial error by controlling the spatial interpolation error in L^∞ norm [6, 7, 15, 18, 35, 37, 40]. As far as we are concerned, we choose to use the continuous mesh framework introduced in [27, 28] and used in [4, 26, 29].

The considered problem of mesh adaptation consists in finding the mesh $\mathcal{H}$ of Ω that minimizes the linear interpolation error $u - \Pi_h u$ in L^p norm, for a given sensor u and for a given number of mesh vertices N. The problem is thus stated in an *a priori* way:

$$\text{Find } \mathcal{H}_{L^p} \text{ having } N \text{ vertices such that } E_{L^p}(\mathcal{H}_{L^p}) = \min_{\mathcal{H}} \left(\int_{\Omega_h} |u(\mathbf{x}) - \Pi_h u(\mathbf{x})|^p\,\mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}\,. \tag{1}$$

As it, Problem (1) is a global combinatorial problem which turns out to be intractable in pratice. Indeed, both topology and vertices locations need to be optimized. This ill-posed problem can be reformulated in the continuous mesh framework [27, 28]. In this framework,

we propose the following continuous model of a mesh. A continuous mesh $\mathbf{M}$ of a domain Ω is identified to a Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ that, at each point $\mathbf{x}$ of Ω, prescribes a density, a set of anisotropy directions and the stretching along these directions, this information being locally contained in metric tensor $\mathcal{M}(\mathbf{x})$. The spatial size of the continuous mesh is given by its spatial complexity: $\mathcal{C}(\mathbf{M}) = \int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x})}\mathrm{d}\mathbf{x}$ which is the continuous counterpart of the number of vertices. Continuous mesh $\mathbf{M}$ defines a class of equivalence of discrete meshes, which are all unit meshes with respect to $\mathbf{M}$. We also define a continuous model of the linear interpolation operator Π_h denoted $\pi_\mathcal{M}$. It is then possible to recast (1) into a well-posed continuous global optimization problem of finding the optimal continuous mesh minimizing the continuous interpolation error in L^p norm, for a given sensor u and for a given spatial complexity $\mathcal{N}$:

$$\text{Find } \mathbf{M}_{L^p} \text{ such that } \mathcal{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \left(\int_\Omega |u(\mathbf{x}) - \pi_\mathcal{M} u(\mathbf{x})|^p \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}, \tag{2}$$

under constraint:

$$\mathcal{C}(\mathbf{M}) = \int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x})}\mathrm{d}\mathbf{x} = \mathcal{N}\,. \tag{3}$$

The continuous mesh spatial complexity $\mathcal{N}$ enables the user to control the level of accuracy of the mesh, and thus, to implicitly control the number of vertices of the resulting discrete mesh. According to [27], if $\mathcal{H}$ is a unit mesh with respect to $\mathbf{M}$ and u is a smooth function, then the following bound holds:

$$\|u - \Pi_h u\|_{L^p(\Omega_h)} \leq \|u - \pi_\mathcal{M} u\|_{L^p(\Omega)} = \left(\int_\Omega \left(\mathrm{trace}\big(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x})|H_u(\mathbf{x})|\mathcal{M}^{-\frac{1}{2}}(\mathbf{x})\big) \right)^p \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}, \tag{4}$$

where H_u is the Hessian of u and $|H_u|$ the matrix deduced from H_u by taking the absolute value of its eigenvalues. Writing the optimality conditions provides the unique (by convexity) optimal continuous mesh $\mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ solution of Problem (2) under Constraint (3):

$$\mathcal{M}_{L^p}(\mathbf{x}) = D_{L^p} \left(\det |H_u(\mathbf{x})|\right)^{-\frac{1}{2p+d}} |H_u(\mathbf{x})| \quad \text{with} \quad D_{L^p} = \mathcal{N}^{\frac{2}{d}} \left(\int_\Omega (\det |H_u(\mathbf{x})|)^{\frac{p}{2p+d}} \mathrm{d}\mathbf{x} \right)^{-\frac{2}{d}} \tag{5}$$

where d is the domain dimension. In Relation (5), we distinguish three terms. First, constant D_{L^p} is a global normalization constant resulting from Constraint (3). Second, matrix term $|H_u|$ specifies the local mesh orientation and anisotropy (or stretching). Third, scalar term $(\det |H_u|)^{\frac{-1}{2p+d}}$ modifies the local mesh density to take into account the sensitivity of the L^p norm used in the error estimate.

Now, to obtain an optimal discrete mesh $\mathcal{H}_{L^p}$, it is sufficient to generate a unit mesh with respect to $\mathbf{M}_{\mathbf{L}^p} = (\mathcal{M}_{L^p}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$ thanks to Relation (4). Finally, Bound (4) can be rewritten for $\mathbf{M}_{L^p}$, and the following bound follows up for a unit mesh $\mathcal{H}_{L^p}$ with respect to $\mathbf{M}_{L^p}$:

$$E(\mathcal{H}_{L^p}) = \|u - \Pi_h u\|_{L^p(\Omega_h)} \leq \mathcal{E}(\mathbf{M}_{L^p}) = d\mathcal{N}^{-\frac{2}{d}} \left(\int_\Omega (\det |H_u(\mathbf{x})|)^{\frac{p}{2p+d}} \mathrm{d}\mathbf{x} \right)^{\frac{2p+d}{dp}} \leq \frac{Cst}{\mathcal{N}^{2/d}}\,. \tag{6}$$

A main result arises from the previous bound: a global second-order asymptotic mesh convergence is expected for the considered variable u. Indeed, a simple analogy with regular grids leads to $\mathcal{N} = O\left(h^{-d}\right)$ so that the previous estimate becomes:

$$\|u - \Pi_h u\|_{L^p(\Omega_h)} \leq Csth^2.$$

For all $p \in [1, \infty[$, it is possible to analyze in which conditions the last integral in (6) is bounded. Following this idea, it is observed in [4, 26, 29] that the second-order convergence property still holds even when singularities are present in the flow field.

3 Space-time L^p interpolation error analysis and space-time optimal continuous mesh

The multi-scale mesh adaptation presented in the previous section only controls spatial errors. But, in the context of time-dependent problems, temporal errors must be controlled as well. In this study, we do not account for time discretization errors, rather we focus on a space-time analysis of the spatial error in unsteady simulations. In other words, we seek for the optimal space-time mesh controlling the space-time spatial discretization error. Taking into account time discretization errors is not so important for the type of calculations that are shown here, but it can be of paramount impact in many other cases, in particular, when implicit time advancing is considered. The following assumption is then made (it has been demonstrated under specific conditions in [2]): *as an explicit time scheme is used for time advancing, then the error in time is controlled by the error in space under CFL condition.* As far as the above hypothesis is true, the spatial interpolation error is a good measure of the total space-time error of the discretized unsteady system.

3.1 Error model

Our goal is to solve an unsteady PDE defined on computational space-time domain $\mathcal{Q} = \Omega \times [0,T]$ where T is the (positive) maximal time and $\Omega \subset \mathbb{R}^3$ is the spatial domain. Its extension to time-dependent functions reads:

$$(\Pi_h \varphi)(t) = \Pi_h(\varphi(t)), \ \forall\, t \in [0,T].$$

The considered problem of mesh adaptation consists in finding the space-time mesh $\mathcal{H}$ of $\mathcal{Q}$ that minimizes the space-time linear interpolation error $u - \Pi_h u$ in L^p norm, for a given sensor u and for a given number of space-time mesh vertices N_{st}. The problem is thus stated in an *a priori* way:

Find $\mathcal{H}_{opt}$ having N_{st} space-time vertices such that $\mathbf{E}_{L^p}(\mathcal{H}_{opt}) = \min\limits_{\mathcal{H}} \|u - \Pi_h u\|_{L^p(\Omega_h \times [0,T])}$.

In the continuous mesh framework, we rewrite this problem under the continuous form:

$$\text{Find } \mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x},t))_{(\mathbf{x},t)\in\Omega\times[0,T]} \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega\times[0,T])}, \tag{7}$$

under the space-time complexity constraint:

$$\mathcal{C}_{st}(\mathbf{M}) = \int_0^T \tau(t)^{-1} \left(\int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x},t)}\, \mathrm{d}\mathbf{x} \right) \mathrm{d}t = \mathcal{N}_{st}. \tag{8}$$

where the continuous mesh space-time complexity $\mathcal{N}_{st}$ is given. In its definition, $\tau(t)$ is the time-step used at time t of interval $[0,T]$ and $\mathcal{N}_{st}$ is the space-time mesh complexity. Introducing the continuous interpolation error, we recall that we can write the continuous error model as follows:

$$\mathbf{E}_{L^p}(\mathbf{M}) = \left(\int_0^T \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) |H_u(\mathbf{x},t)| \mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) \right)^p \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{\frac{1}{p}}.$$

where H_u is the Hessian of sensor u. To find the optimal space-time continuous mesh, Problem (7-8) is solved in two steps. First, a spatial minimization is done for a fixed t. Second, a temporal minimization is performed.

3.2 Spatial minimization for a fixed t

Let us assume that at time t, we seek for the optimal continuous mesh $\mathbf{M}_{L^p}(t)$ which minimizes the instantaneous error, *i.e.,* the spatial error for a fixed time t:

$$\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}(t)) = \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)\, |H_u(\mathbf{x},t)|\, \mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) \right)^p \mathrm{d}\mathbf{x}$$

under the constraint that the spatial complexity is constant equal to:

$$\mathcal{C}(\mathbf{M}(t)) = \int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x},t)}\, \mathrm{d}\mathbf{x} = \mathcal{N}(t). \tag{9}$$

Similarly to Section 2.2 with $d = 3$, solving the optimality conditions provides the *optimal instantaneous continuous mesh in L^p norm* $\mathbf{M}_{L^p}(t) = (\mathcal{M}_{L^p}(\mathbf{x},t))_{\mathbf{x}\in\Omega}$ at time t defined by:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}(t)^{\frac{2}{3}} \left(\int_\Omega (\det |H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}} \right)^{-\frac{2}{3}} (\det |H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} \, |H_u(\mathbf{x},t)| \,. \tag{10}$$

The corresponding optimal instantaneous error at time t is:

$$\widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t)) = 3^p\, \mathcal{N}(t)^{-\frac{2p}{3}} \left(\int_\Omega (\det |H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \right)^{\frac{2p+3}{3}} = 3^p\, \mathcal{N}(t)^{-\frac{2p}{3}}\, \mathcal{K}(t)^{\frac{2p+3}{3}} \,. \tag{11}$$

For the sequel of this paper, we denote: $\mathcal{K}(t) = \displaystyle\int_\Omega (\det |H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}$.

3.3 Temporal minimization

To complete the resolution of optimization Problem (7-8), we perform a temporal minimization in order to get the optimal space-time continuous mesh. In other words, we need to find the optimal time law $t \to \mathcal{N}(t)$ for the instantaneous mesh size. First, we consider the case where the time-step τ is specified by the user as a function of time $t \to \tau(t)$. Second, we deal with the case of an explicit time advancing solver subject to Courant time-step condition.

Temporal minimization for specified τ We consider the case where the time-step τ is specified by a function of time $t \to \tau(t)$. After the spatial optimization, the space-time error is:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \left(\int_0^T \widetilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t))\, \mathrm{d}t \right)^{\frac{1}{p}} = 3 \left(\int_0^T \mathcal{N}(t)^{-\frac{2p}{3}}\, \mathcal{K}(t)^{\frac{2p+3}{3}}\, \mathrm{d}t \right)^{\frac{1}{p}} \tag{12}$$

and we aim at minimizing it under the following space-time complexity constraint:

$$\int_0^T \tau(t)^{-1} \mathcal{N}(t)\, \mathrm{d}t = \mathcal{N}_{st}. \tag{13}$$

In other words, we concentrate on seeking *the optimal distribution of $\mathcal{N}(t)$ when the space-time complexity $\mathcal{N}_{st}$ is prescribed.* Let us apply the one-to-one change of variables:

$$\tilde{\mathcal{N}}(t) = \mathcal{N}(t)\, \tau(t)^{-1} \quad \text{and} \quad \tilde{\mathcal{K}}(t) = \tau(t)^{-\frac{2p}{3}}\, \mathcal{K}(t)^{\frac{2p+3}{3}} \,.$$

Then, our temporal optimization problem becomes, find space-time continuous mesh $\mathbf{M}_{L^p}$ such that:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = \min_{\mathbf{M}}\ 3^p \int_0^T \tilde{\mathcal{N}}(t)^{-\frac{2p}{3}}\, \tilde{\mathcal{K}}(t)\, \mathrm{d}t \quad \text{under constraint} \quad \int_0^T \tilde{\mathcal{N}}(t)\, \mathrm{d}t = \mathcal{N}_{st} \,.$$

The solution of this problem is given by:

$$\tilde{\mathcal{N}}_{opt}(t)^{-\frac{2p+3}{3}}\, \tilde{\mathcal{K}}(t) = Cst \quad \Rightarrow \quad \mathcal{N}_{opt}(t) = C(\mathcal{N}_{st})\ \tau(t)^{\frac{3}{2p+3}}\, \mathcal{K}(t) \,.$$

Here, constant $C(\mathcal{N}_{st})$ can be obtained by introducing the above expression in space-time complexity Constraint (13), leading to:

$$C(\mathcal{N}_{st}) = \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\, \mathcal{K}(t) \mathrm{d}t \right)^{-1} \mathcal{N}_{st} \,,$$

which completes the description of the optimal space-time metric for a prescribed time-step. Using Relation (10), the analytic expression of the optimal space-time metric in L^p norm $\mathbf{M}_{L^p}$ is:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}} \mathcal{K}(t)\mathrm{d}t \right)^{-\frac{2}{3}} \tau(t)^{\frac{2}{2p+3}} (\det |H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} |H_u(\mathbf{x},t)| \,. \quad (14)$$

The following optimal error is finally obtained:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\mathcal{N}_{st}^{-\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}} \mathcal{K}(t)\mathrm{d}t \right)^{\frac{2p+3}{3p}} . \quad (15)$$

This analysis provides the optimal continuous meshes for each time level.

Temporal minimization for explicit time advancing In the case of an explicit time advancing scheme subject to a Courant condition, the situation is trickier, since the time-step strongly depends on the smallest mesh size. We restrict ourselves to the case of smooth data and solution.

We still seek for the optimal continuous mesh that minimizes space-time Error (12) under complexity Constraint (13). Let $\Delta x_{min,1}(t) = \min_{\mathbf{x}} \min_i h_i(\mathbf{x})$ be the smallest mesh size of the optimal instantaneous continuous mesh in L^p norm at time t of unit complexity $\mathbf{M}_{L^p,1}(t)$, *i.e,* Relation (10) with $\mathcal{C}(\mathbf{M}_{L^p}(t)) = 1$:

$$\mathcal{M}_{\mathbf{L}^p,1}(\mathbf{x},t) = \mathcal{K}(t)^{-\frac{2}{3}} (\det |H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} \; |H_u(\mathbf{x},t)|. \quad (16)$$

Since the coefficients of a metric tensor have the same dimension as $1/h^2$, where h is the typical mesh size, we deduce the smallest mesh size of $\mathbf{M}_{L^p}(t)$ given by Relation (10):

$$\Delta x_{min}(t) = \mathcal{N}(t)^{-\frac{1}{3}} \, \Delta x_{min,1}(t),$$

where $\Delta x_{min,1}(t)$ is independent of the mesh complexity. A way to write the Courant condition for time-advancing is to define the time-step $\tau(t)$ by:

$$\tau(t) = c(t)^{-1} \, \Delta x_{min}(t) = \mathcal{N}(t)^{-\frac{1}{3}} \, c(t)^{-1} \, \Delta x_{min,1}(t) \,, \quad (17)$$

where $c(t)$ is the maximal wave speed over the domain at time t. Again, we search for the optimal distribution of $\mathcal{N}(t)$ when the space-time complexity $\mathcal{N}_{st}$ is prescribed by Relation (13), with

$$\mathcal{N}_{st} = \int_0^T \mathcal{N}(t)^{\frac{4}{3}} \, c(t) \, (\Delta x_{min,1}(t))^{-1} \, \mathrm{d}t \,.$$

We choose to apply the one-to-one change of variables:

$$\hat{\mathcal{N}}(t) = \mathcal{N}(t)^{\frac{4}{3}} \, c(t) \, (\Delta x_{min,1}(t))^{-1} \quad \text{and} \quad \hat{\mathcal{K}}(t) = \mathcal{K}(t)^{\frac{2p+3}{3}} \, c(t)^{\frac{p}{2}} \, (\Delta x_{min,1}(t))^{-\frac{p}{2}} \,.$$

Therefore, the corresponding space-time approximation error over the simulation time interval and space-time complexity reduces to:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = 3^p \int_0^T \mathcal{N}(t)^{-\frac{2p}{3}} \, \mathcal{K}(t)^{\frac{2p+3}{3}} \, \mathrm{d}t = 3^p \int_0^T \hat{\mathcal{N}}(t)^{-\frac{p}{2}} \hat{\mathcal{K}}(t) \, \mathrm{d}t \quad \text{and} \quad \int_0^T \hat{N}(t)\mathrm{d}t = N_{st} \,.$$

This optimization problem has for solution:

$$\hat{\mathcal{N}}_{opt}(t)^{-\frac{p+2}{2}} \hat{\mathcal{K}}(t) = Cst \quad \Rightarrow \quad \hat{\mathcal{N}}_{opt}(t) = C(N_{st}) \; \hat{\mathcal{K}}(t)^{\frac{2}{p+2}} \,,$$

and by considering the space-time complexity constraint relation we deduce:

$$C(\mathcal{N}_{st}) = \mathcal{N}_{st} \left(\int_0^T \hat{\mathcal{K}}(t)^{\frac{2}{p+2}} \, \mathrm{d}t \right)^{-1} .$$

For the sake of clarity, we set $\theta(t) = c(t)\,(\Delta x_{min,1}(t))^{-1}$. Using the definitions of $\hat{\mathcal{N}}$ and $\hat{\mathcal{K}}$ in the above relations, we get:

$$\mathcal{N}(t)^{\frac{4}{3}}\,\theta(t) = \mathcal{N}_{st}\,\left(\int_0^T \left(\mathcal{K}(t)^{\frac{2p+3}{3}}\,\theta(t)^{\frac{p}{2}}\right)^{\frac{2}{p+2}}\,\mathrm{d}t\right)^{-1}\left(\mathcal{K}(t)^{\frac{2p+3}{3}}\,\theta(t)^{\frac{p}{2}}\right)^{\frac{2}{p+2}}$$

$$\Longleftrightarrow\quad \mathcal{N}(t) = \mathcal{N}_{st}^{\frac{3}{4}}\,\left(\int_0^T \left(\mathcal{K}(t)^{\frac{2p+3}{3}}\,\theta(t)^{\frac{p}{2}}\right)^{\frac{2}{p+2}}\,\mathrm{d}t\right)^{-\frac{3}{4}}\left(\mathcal{K}(t)^{\frac{2p+3}{3}}\,\theta(t)^{-1}\right)^{\frac{3}{2(p+2)}}.$$

Consequently, after some simplifications, we obtain the following expression of the optimal space-time continuous mesh $\mathbf{M}_{L^p}$ and error:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}_{st}^{\frac{1}{2}}\,\left(\int_0^T \left(\theta(t)^{\frac{p}{2}}\,\mathcal{K}(t)^{\frac{2p+3}{3}}\right)^{\frac{2}{p+2}}\,\mathrm{d}t\right)^{-\frac{1}{2}}\theta(t)^{\frac{-1}{p+2}}\,\mathcal{K}(t)^{\frac{-1}{3(p+2)}}\,(\det|H_u(\mathbf{x},t)|)^{\frac{-1}{2p+3}}\,|H_u(\mathbf{x},t)| \tag{18}$$

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\,\mathcal{N}_{st}^{-\frac{1}{2}}\,\left(\int_0^T \left(\theta(t)^{\frac{p}{2}}\,\mathcal{K}(t)^{\frac{2p+3}{3}}\right)^{\frac{2}{p+2}}\,\mathrm{d}t\right)^{\frac{p+2}{2p}}, \tag{19}$$

where $\theta(t) = c(t)\,(\Delta x_{min,1}(t))^{-1}$ and $\mathcal{K}(t) = \int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}}\mathrm{d}\mathbf{x}$. This analysis provides the optimal continuous meshes for each time level.

4 Error analysis for the global fixed-point mesh adaptation algorithm

The computation of the optimal continuous mesh given by Relation (14) or (18) involves a global normalization term which requires the knowledge of quantities over the whole simulation time frame. For instance, Relation (14) has for global normalization term:

$$\mathcal{N}_{st}^{\frac{2}{3}}\,\left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\left(\int_\Omega (\det|H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}}\mathrm{d}\bar{\mathbf{x}}\right)\mathrm{d}t\right)^{-\frac{2}{3}},$$

which requires knowledge of all the time-steps $\tau(t)$ and Hessians $H_u(\mathbf{x},t)$ over time frame $[0,T]$. Thus, the complete simulation must be performed before evaluating any space-time continuous mesh. We consider a *global* fixed-point mesh adaptation algorithm covering the whole time frame $[0,T]$. This iterative algorithm is used to converge the non-linear mesh adaptation problem, *i.e.,* converging the mesh-solution couple. This is also a way to predict the solution evolution and to adapt the mesh accordingly.

Moreover, the previous analysis provides the optimal size of the adapted meshes for each time level. Hence, this analysis requires the mesh to be adapted at each flow solver time-step which is inconceivable in practical applications. We propose to use a coarse adapted discretization of the time axis. The basic idea consists in splitting the simulation time frame $[0,\,T]$ into n_{adap} adaptation sub-intervals:

$$[0,\,T] \;=\; [0=t^1,\,t^2]\cup\ldots\cup[t^i,\,t^{i+1}]\cup\ldots\cup[t^{n_{adap}},\,t^{n_{adap}+1}=T]\,,$$

and to keep the same adapted spatial mesh for each time sub-interval. On each sub-interval, the mesh is adapted to control the solution accuracy from t^i to t^{i+1}. Consequently, the time-dependent simulation is performed with n_{adap} different adapted meshes. This drastically reduces the number of generated meshes during the simulation, hence the number of solution transfers. Moreover, the flow solver performs many iterations (tens to hundreds) on the same fixed spatial mesh. This provides a first answer to the adaptation of the whole space-time mesh, the spatial mesh being kept constant for each sub-interval when the global space-time mesh is visualized.

Now, we want to extend the previous analysis (Section 3) to the fixed-point mesh adaptation algorithm context where the simulation time interval [0,T] is split into n_{adap} *sub-intervals* $[t^i, t^{i+1}]$ for $i = 1, .., n_{adap}$. Each spatial mesh $\mathbf{M}^i$ is then kept constant during each sub-interval $[t^i, t^{i+1}]$. We could consider this partition as a *time discretization of the mesh adaptation problem.* Here, the proposed approach leads to an optimal discrete answer.

4.1 Spatial minimization on a sub-interval

Given the continuous mesh spatial complexity $\mathcal{N}^i$ for the single adapted mesh used during time sub-interval $[t^i, t^{i+1}]$, we seek for the optimal continuous mesh $\mathbf{M}^i_{L^p}$ solution of the following problem:

$$\widetilde{\mathbf{E}}^i_{L^p}(\mathbf{M}^i_{L^p}) = \min_{\mathbf{M}^i} \int_\Omega \text{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\,\mathbf{H}^i_u(\mathbf{x})\,(\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\right)^p \mathrm{d}\mathbf{x} \quad \text{such that} \quad \mathcal{C}(\mathbf{M}^i) = \mathcal{N}^i\,,$$

where matrix $\mathbf{H}^i_u$ on the sub-interval can be defined by either using a L^1 or a L^∞ norm:

$$\mathbf{H}^i_{L^1}(\mathbf{x}) = \int_{t^i}^{t^{i+1}} |H_u(\mathbf{x},t)|\,\mathrm{d}t \quad \text{or} \quad \mathbf{H}^i_{L^\infty}(\mathbf{x}) = \Delta t^i \max_{t\in[t^i,t^{i+1}]} |H_u(\mathbf{x},t)|\,, \tag{20}$$

with $\Delta t^i = t^{i+1} - t^i$. As previously, we get the spatial optimality condition:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = (\mathcal{N}^i)^{\frac{2}{3}} \left(\int_\Omega (\det \mathbf{H}^i_u(\bar{\mathbf{x}}))^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}}\right)^{-\frac{2}{3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}}\ \mathbf{H}^i_u(\mathbf{x}).$$

The corresponding optimal error $\mathbf{E}^i(\mathbf{M}^i_{L^p})$ is:

$$\widetilde{\mathbf{E}}^i_{L^p}(\mathbf{M}^i_{L^p}) = 3^p\,(\mathcal{N}^i)^{-\frac{2p}{3}} \left(\int_\Omega (\det \mathbf{H}^i_u(\mathbf{x}))^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}} = 3^p\,(\mathcal{N}^i)^{-\frac{2p}{3}}\,(\mathcal{K}^i)^{\frac{2p+3}{3}}\,.$$

where $\mathcal{K}^i = \displaystyle\int_\Omega \left(\det \mathbf{H}^i_u(\mathbf{x})\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}$.

To complete our analysis, a temporal minimization must be carried out. Again, we first consider the case where the time-step τ is specified by a function of time. We then deal with the case of an explicit time advancing solver subject to Courant time-step condition.

4.2 Temporal minimization

Temporal minimization for specified τ After the spatial minimization, the temporal optimization problem becomes find the optimal space-time continuous mesh $\mathbf{M}_{L^p}$ such that:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = \min_{\mathbf{M}} \sum_{i=1}^{n_{adap}} \widetilde{\mathbf{E}}^i_{L^p}(\mathbf{M}^i_{L^p}) = \min_{\mathbf{M}} 3^p \sum_{i=1}^{n_{adap}} (\mathcal{N}^i)^{-\frac{2p}{3}}\,(\mathcal{K}^i)^{\frac{2p+3}{3}}\,,$$

under space-time complexity constraint:

$$\sum_{i=1}^{n_{adap}} \mathcal{N}^i \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right) = \mathcal{N}_{st}\,.$$

We set the one-to-one mapping:

$$\tilde{\mathcal{N}}^i = \mathcal{N}^i \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right) \quad \text{and} \quad \tilde{\mathcal{K}}^i = (\mathcal{K}^i)^{\frac{2p+3}{3}} \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{3}}\,,$$

then the optimization problem reduces to:

$$\min_{\mathbf{M}} \sum_{i=1}^{n_{adap}} (\tilde{\mathcal{N}}^i)^{-\frac{2p}{3}}\,\tilde{\mathcal{K}}^i \quad \text{such that} \quad \sum_{i=1}^{n_{adap}} \tilde{\mathcal{N}}^i = \mathcal{N}_{st}\,.$$

The solution is:

$$\tilde{\mathcal{N}}^i_{opt} = C(\mathcal{N}_{st})\,(\tilde{\mathcal{K}}^i)^{\frac{3}{2p+3}} \quad \text{with} \quad C(\mathcal{N}_{st}) = \mathcal{N}_{st}\left(\sum_{i=1}^{n_{adap}} (\tilde{\mathcal{K}}^i)^{\frac{3}{2p+3}}\right)^{-1}$$

$$\Rightarrow \quad \mathcal{N}^i = \mathcal{N}_{st}\left(\sum_{j=1}^{n_{adap}} \mathcal{K}^j \left(\int_{t^j}^{t^{j+1}} \tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{2p+3}}\right)^{-1} \mathcal{K}^i \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right)^{-\frac{3}{2p+3}} .$$

We deduce the following optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ and error:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}}\left(\sum_{j=1}^{n_{adap}} \mathcal{K}^j \left(\int_{t^j}^{t^{j+1}} \tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{2p+3}}\right)^{-\frac{2}{3}} \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right)^{-\frac{2}{2p+3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}}\, \mathbf{H}^i_u(\mathbf{x}) \tag{21}$$

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\,\mathcal{N}_{st}^{-\frac{2}{3}}\left(\sum_{i=1}^{n_{adap}} \mathcal{K}^i \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{2p+3}}\right)^{\frac{2p+3}{3p}} . \tag{22}$$

Temporal minimization for explicit time advancing Similarly to the previous section, the Courant-based time-step is:

$$\tau(t) = c(t)^{-1}\,\Delta x^i_{min} = (\mathcal{N}^i)^{-\frac{1}{3}}\, c(t)^{-1}\,\Delta x^i_{min,1} \quad \text{for } t \in [t^i, t^{i+1}]\,,$$

where $\Delta x^i_{min,1}$ is the smallest height of $\mathbf{M}^i_{L^p,1}$ and $c(t)$ is the maximal wave speed over the domain. The optimization problem is find the optimal space-time continuous mesh $\mathbf{M}_{L^p}$ such that:

$$(\mathbf{E}_{L^p}(\mathbf{M}_{L^p}))^p = \min_{\mathbf{M}} \sum_{i=1}^{n_{adap}} \widetilde{\mathbf{E}}^i_{L^p}(\mathbf{M}^i_{L^p}) = \min_{\mathbf{M}} 3^p \sum_{i=1}^{n_{adap}} (\mathcal{N}^i)^{-\frac{2p}{3}}\, (\mathcal{K}^i)^{\frac{2p+3}{3}}$$

under the constraint:

$$\sum_{i=1}^{n_{adap}} (\mathcal{N}^i)^{\frac{4}{3}} \left(\int_{t_{i-1}}^{t_i} c(t)\,(\Delta x^i_{min,1})^{-1}\mathrm{d}t\right) = \mathcal{N}_{st}\,.$$

For the sake of clarity, we set: $\theta^i = \displaystyle\int_{t_{i-1}}^{t_i} c(t)\,(\Delta x^i_{min,1})^{-1}\mathrm{d}t$. We specify again:

$$\hat{\mathcal{N}}^i = \left(\mathcal{N}^i\right)^{\frac{4}{3}}\theta^i \quad \text{and} \quad \hat{\mathcal{K}}^i = (\mathcal{K}^i)^{\frac{2p+3}{3}}\left(\theta^i\right)^{\frac{p}{2}} .$$

Then, the optimization problem becomes:

$$\min_{\mathbf{M}} 3^p \sum_{i=1}^{n_{adap}} (\hat{\mathcal{N}}^i)^{-\frac{p}{2}}\,\hat{\mathcal{K}}^i \quad \text{such that} \quad \sum_{i=1}^{n_{adap}} \hat{\mathcal{N}}^i = \mathcal{N}_{st}\,.$$

This optimization problem has for solution:

$$\hat{\mathcal{N}}^i_{opt} = C(\mathcal{N}_{st})\,(\hat{\mathcal{K}}^i)^{\frac{2}{p+2}} \quad \text{with} \quad C(\mathcal{N}_{st}) = \mathcal{N}_{st}\left(\sum_{i=1}^{n_{adap}} (\hat{\mathcal{K}}^i)^{\frac{2}{p+2}}\right)^{-1} ,$$

from which we deduce:

$$\mathcal{N}^i_{opt} = \mathcal{N}_{st}^{\frac{3}{4}}\left(\sum_{j=1}^{n_{adap}} \left((\mathcal{K}^j)^{\frac{2p+3}{3}}\,(\theta^j)^{\frac{p}{2}}\right)^{\frac{2}{p+2}}\right)^{-\frac{3}{4}} \left((\mathcal{K}^i)^{\frac{2p+3}{3}}\,(\theta^i)^{-1}\right)^{\frac{3}{2(p+2)}} .$$

The optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ and error read:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{1}{2}}\left(\sum_{j=1}^{n_{adap}} \left((\mathcal{K}^j)^{\frac{2p+3}{3}}\,(\theta^j)^{\frac{p}{2}}\right)^{\frac{2}{p+2}}\right)^{-\frac{1}{2}} (\theta^i)^{\frac{-1}{p+2}}\,(\mathcal{K}^i)^{\frac{-1}{3(p+2)}}\,(\det \mathbf{H}^i_u(\mathbf{x}))^{\frac{-1}{2p+3}}\,\mathbf{H}^i_u(\mathbf{x}) \tag{23}$$

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\,\mathcal{N}_{st}^{-\frac{1}{2}} \left(\sum_{i=1}^{n_{adap}} \left((\mathcal{K}^i)^{\frac{2p+3}{3}} (\theta^i)^{\frac{p}{2}} \right)^{\frac{2}{p+2}} \right)^{\frac{p+2}{2p}} . \tag{24}$$

5 From theory to practice

The global fixed-point mesh adaptation algorithm is schematized in Algorithm 1 where $\mathcal{H}$, $\mathcal{S}$ and $\mathcal{M}$ denote respectively meshes, solutions and metrics. And, $\mathbf{H}$ is the Hessian-metric given by Relation (20). In the following, we describe each step of this algorithm.

5.1 Computation of the optimal continuous mesh

Computation of the Hessian-metric Practically, it remains to know how to compute the Hessian-metric $\mathbf{H}_u^i$ on sub-interval i given by Relations (20), *i.e.*, how it is discretized. The strategy adopted in [2] is to sample the solution on the time sub-interval. More precisely, n_k solutions equally distributed on the sub-interval time frame are saved, including the initial solution at t^i and the final solution at t^{i+1}. Positive Hessian $|H_u(\mathbf{x}, t^k)|$, which is obtained by taking the absolute value of the eigenvalues of $H_u(\mathbf{x}, t^k)$, is evaluated for each sample. If the samples are balanced in time, the time elapsed between two samples is $\frac{\Delta t^i}{n_k - 1}$ where $\Delta t^i = t^{i+1} - t^i$ is the sub-interval time length. In practice, we need to choose enough sample to mesh properly the regions where physical phenomena evolve during the sub-interval. In this paper, twenty solution samples are sufficient for all the presented examples.

For Hessian-metric $\mathbf{H}_{L^1}^i$, the following discretization is done:

$$\mathbf{H}_{L^1}^i(\mathbf{x}) \approx \frac{1}{2}\frac{\Delta t^i}{n_k - 1}|H_u(\mathbf{x}, t^i)| + \frac{\Delta t^i}{n_k - 1}\sum_{k=2}^{n_k - 1}|H_u(\mathbf{x}, t^k)| + \frac{1}{2}\frac{\Delta t^i}{n_k - 1}|H_u(\mathbf{x}, t^{i+1})| = \Delta t^i\,|H_{\text{avg}}^i(\mathbf{x})|\,,$$

where $t^k = t^i + \frac{k-1}{n_k - 1}\Delta t^i$.

Algorithm 1 Mesh Adaptation Loop for Unsteady Flows

Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

 # Fixed-point loop to converge the global space-time mesh adaptation problem
For $j = 1, n_{ptfx}$

 # Adaptive loop to advance the solution in time on time frame $[0, T]$

1. For $i = 1, n_{adap}$
   (a) $\mathcal{S}_{0,i}^j$ = Interpolate conservatively next sub-interval initial solution from $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;
   (b) $\mathcal{S}_i^j$ = Compute solution on sub-interval from pair $(\mathcal{S}_{0,i}^j, \mathcal{H}_i^j)$;
   (c) $|\mathbf{H}|_i^j$ = Compute sub-interval Hessian-metric from solution sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k)\}_{k=1,nk})$;

   EndFor
2. $\mathcal{C}^j$ = Compute space-time complexity from all Hessian-metrics $(\{|\mathbf{H}|_i^j\}_{i=1,n_{adap}})$;
3. $\{\mathcal{M}_i^j\}_{i=1,n_{adap}}$ = Compute all sub-interval unsteady metrics $(\mathcal{C}^j, \{|\mathbf{H}|_i^j\}_{i=1,n_{adap}})$;
4. $\{\mathcal{M}_i^j\}_{i=1,n_{adap}}$ = Metric gradation on all sub-interval unsteady metrics $\{\mathcal{M}_i^j\}_{i=1,n_{adap}}$;
5. $\{\mathcal{H}_i^{j+1}\}_{i=1,n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}_i^j,\ \mathcal{M}_i^j\}_{i=1,n_{adap}})$;

EndFor

For Hessian-metric $\mathbf{H}^i_{L^\infty}$, the following discretization is used:

$$\mathbf{H}^i_{L^\infty}(\mathbf{x}) \approx \Delta t^i \bigcap_{k=1}^{n_k} |H_u(\mathbf{x}, t^k)| = \Delta t^i \, |H^i_{\max}(\mathbf{x})| \,,$$

where $\cap$ has to be understood as the metric intersection in time of all samples [2].

Choice of the optimal continuous mesh The optimal adapted mesh for each sub-interval is generated according to the analysis performed in Section 4. For the numerical results presented below, we select the optimal mesh given by Relation (21) and the following particular choice has been made:

- the Hessian-metric for sub-interval i is discretized in time in L^1 norm: $\mathbf{H}^i_{L^1}$
- all sub-intervals have the same time length $\Delta t = T/n_{adap}$.

Moreover, integral $\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t$ corresponds to the number of time-steps (iterations) performed by the flow solver during the ith sub-interval. In practice using the flow solver number of iterations of each sub-interval to define the continuous mesh may cause trouble because it highly depends on the discrete representation of the continuous mesh, *i.e.,* the generated discrete mesh. Thus, the number of iterations of a sub-interval may substantially vary between two fixed-point iterations. To avoid this issue, one may prefer considering the flow solver time-step constant on each sub-interval. In that case, we can just consider the time-step $\tau(t)$ constant and equal to Δt (thanks to the global normalization term) so the integral $\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t$ reduces to 1. Another approach is to compute the continuous time-step by means of the CFL condition associated with the continuous mesh. This will cancel any issue due to the spatial discretization. Here, we chose the first approach by considering the time-step constant.

With these choices, the optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ simplifies to:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} \left(\int_\Omega (\det |\mathbf{H}^j_{L^1}(\mathbf{x})|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \right) \right)^{-\frac{2}{3}} \left(\det |\mathbf{H}^i_{L^1}(\mathbf{x})|\right)^{-\frac{1}{2p+3}} |\mathbf{H}^i_{L^1}(\mathbf{x})| \,. \quad (25)$$

In that case, as we assume that theoretically one time-step is done by sub-interval, $\mathcal{N}_{avg} = \mathcal{N}_{st}/n_{adap}$ represents the average spatial complexity by sub-interval. We do not prescribe the temporal complexity, *i.e.,* we do not control the number of time-steps done at each sub-interval.

In practice, the user prescribes the number of sub-intervals n_{adap} and the sub-interval average spatial complexity $\mathcal{N}_{avg}$ leading to a total space-time complexity of

$$\mathcal{N}_{st} = n_{adap} \times \mathcal{N}_{avg} \,. \quad (26)$$

This prescription is the **theoretical** complexity. In fact, the total number of space-time vertices N_{st} of the simulation **discrete meshes** is directly proportional to the prescribed total space-time complexity:

$$N_{st} = c \, \mathcal{N}_{st} \,,$$

where coefficient c depends on the physics of the problem, the geometry of the problem, the mesh gradation (see Section 5.4), and the local remesher. Nevertheless, for the same simulation, if a different total space-time complexity is prescribed, e.g. $k\,\mathcal{N}_{st}$, then we observe that the resulting total number of space-time vertices of the discrete meshes is proportional with the same coefficient c, e.g. almost equal to $c\,k\,\mathcal{N}_{st}$.

Remark 5.1. The temporal minimization distributes optimally the number of spatial vertices for each sub-interval in order to minimize the error. This number is linked to the integral:

$$\int_\Omega (\det |\mathbf{H}^j_{L^1}(\mathbf{x})|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \,.$$

Consequently, the resulting discrete space-time mesh has a different number of spatial vertices for each sub-interval and the total number of spatial vertices of the space-time mesh is:

$$N_{spatial} = \sum_{i=1}^{n_{adap}} N^i \,. \tag{27}$$

5.2 Matrix-free $\mathbb{P}_1$-conservative solution interpolation

At each remeshing, the solution needs to be transferred from the previous mesh to the next one to pursue the computation. This stage becomes crucial in the context of unsteady problems and even more if a large number of transfers is performed, as the error introduced by this stage can spoil the overall accuracy of the solution. In the context of the resolution by a second order numerical scheme of a PDE system of conservation laws, as the compressible Euler system, it seems mandatory for the interpolation method to satisfy the following properties in order to obtain a consistent mesh adaptation scheme: mass conservation, $\mathbb{P}_1$ exactness preserving the second order of the adaptive strategy, and verify the maximum principle.

The mass conservation property of the interpolation operator is achieved by local mesh intersections, *i.e.,* intersections are performed at the element level. The use of mesh intersection to build a conservative interpolation process seems natural for unrelated - *a fortiori* non-embedded - meshes. The locality is primordial for efficiency and robustness. The idea is to find, for each element of the new mesh, its geometric intersection with all the elements of the background mesh it overlaps and to mesh this geometric intersection with simplices. We are then able to use a Gauss quadrature formula to exactly compute the mass which has been locally transferred.

High-order accuracy is obtained through the reconstruction of the gradient of the solution from the discrete data and the use of some Taylor formulae. Unfortunately, this high-order interpolation can lead to a loss of monotonicity. The maximum principle is recovered by correcting the interpolated solution in a conservative manner, using a limiter strategy very similar to the one used for Finite-Volume solvers. Finally, the solution values at vertices are reconstructed from this piecewise linear by element discontinuous representation of the solution. More details are given in [5]. The algorithm is summarized in Algorithm 2 where m_K stands for the integral of any conservative quantities (density, momentum and energy) on the considered element.

Algorithm 2 Conservative Interpolation Process

Piecewise linear (continuous or discontinuous) representation of the solution on $\mathcal{H}_{back}$

1. For all elements $K_{back} \in \mathcal{H}_{back}$, compute solution mass $m_{K_{back}}$ and gradient $\nabla_{K_{back}}$
2. For all elements $K_{new} \in \mathcal{H}_{new}$, recover solution mass $m_{K_{new}}$ and gradient $\nabla_{K_{new}}$:
   (a) compute the intersection of K_{new} with all $K^i_{back} \in \mathcal{H}_{back}$ it overlaps
   (b) mesh the intersection polygon/polyhedra of each couple of elements (K_{new}, K^i_{back})
   (c) compute $m_{K_{new}}$ and $\nabla_{K_{new}}$ using Gauss quadrature formulae

   $\Longrightarrow$ a piecewise linear discontinuous representation of the mass on $\mathcal{H}_{new}$ is obtained
3. Correct the gradient to enforce the maximum principle
4. Set the solution values to vertices by an averaging procedure.

Figure 1 points out the superiority of the $\mathbb{P}_1$-conservative solution transfer (right) with respect to the classic $\mathbb{P}_1$ interpolation (left) on an adaptive blast simulation in three dimensions, see Section 7.3. For both simulations, all parameters are the same except for the solution transfer stage. This figure shows the final solution obtained with 128 mesh adaptations, *i.e.,* a total of 128 solution interpolations. The error introduced during the classic $\mathbb{P}_1$ solution

interpolation accumulates throughout the simulation and clearly spoils the solution accuracy while the solution remains highly resolved with the $\mathbb{P}_1$-conservative operator.

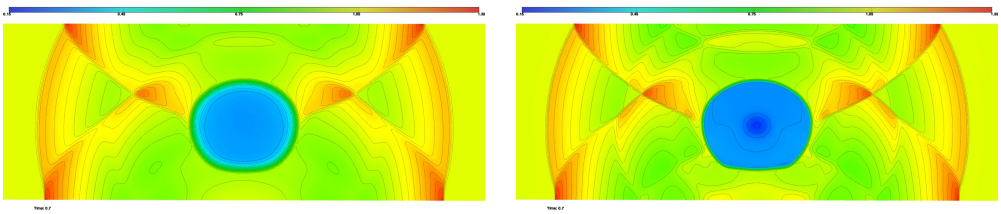

Figure 1: Final solution of a blast adaptive simulation (see Section 7.3) with 128 mesh adaptations. Left, using a classic $\mathbb{P}_1$ solution interpolation. Right, using the $\mathbb{P}_1$-conservative solution interpolation.

5.3 The flow solver: Wolf

In all the examples, the flow is modeled by the conservative Euler equations. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the Euler equations for mass, momentum and energy conservation read:

$$\frac{\partial W}{\partial t} + \nabla \cdot F(W) = 0\,, \tag{28}$$

where $W = {}^t(\rho, \rho\mathbf{u}, \rho E)$ is the conservative variables vector and vector F represents the convective operator:

$$F(W) = {}^t\left(\rho\mathbf{u}, \rho u\mathbf{u} + pe_x, \rho v\mathbf{u} + pe_y, \rho w\mathbf{u} + pe_z, \mathbf{u}(\rho E + p)\right)\,.$$

We have noted ρ the density, $\mathbf{u} = (u, v, w)$ the velocity vector, $E = T + \frac{\|\mathbf{u}\|^2}{2}$ the total energy and $p = (\gamma - 1)\rho T$ the pressure with $\gamma = 1.4$ the ratio of specific heat capacities. and T the temperature.

The Euler system is solved by means of a Finite Volume technique on unstructured meshes composed of tetrahedra. The proposed scheme is vertex-centered, achieves a second order accuracy in space thanks to a MUSCL type reconstruction method with a numerical dissipation of sixth order and a second order accuracy in time with an explicit Runge-Kutta scheme. More details can be found in [4, 11].

5.4 Metric field gradation

Sizes and orientations prescriptions in the generation of anisotropic meshes are achieved using metric fields which may have huge variations or may be quite irregular when evaluated from numerical solutions exhibiting discontinuities or steep gradients. This makes the generation of a unit mesh difficult or impossible, thus leading to poor quality anisotropic meshes. Generating high-quality anisotropic meshes requires smoothing of the metric field by bounding its variations in all directions. To this end, a mesh gradation control procedure is used [1, 23]. It consists of reducing in all directions the size prescribed at any points if the variation of the metric field is larger than a fixed threshold. We adopt the following continuous vision of the anisotropic mesh gradation control. Each point of the domain defines a metric field in the whole domain by growing its metric at a rate given by the desired gradation coefficient. These fields define well-graded smooth continuous meshes over the domain and represent the size constraint imposed by each point in the entire domain. Then, each point of the domain has to take into account these constraints to guarantee a metric field with a smooth variation controlled by the size gradation.

Let $\mathbf{p}$ be a point of a domain Ω supplied with a metric $\mathcal{M}(\mathbf{p})$ and β the specified gradation. Point $\mathbf{p}$ imposes at each point $\mathbf{x}$ of the domain a growth metric constraint $\mathcal{M}_p(\mathbf{x})$ given by:

$$\mathcal{M}_p(\mathbf{x}) = {}^t\mathcal{R}\,\Xi(\mathbf{px})\,\Lambda\,\mathcal{R} \quad \text{where} \quad \Xi(\mathbf{px}) = \begin{pmatrix} \eta_1^2(\mathbf{px}) & 0 & 0 \\ 0 & \eta_2^2(\mathbf{px}) & 0 \\ 0 & 0 & \eta_3^2(\mathbf{px}) \end{pmatrix}\,. \tag{29}$$

where a growth factor is associated independently with each eigenvalue of $\mathcal{M}$:

$$\eta_i^2(\mathbf{px}) = \left((1+\sqrt{\lambda_i}\|\mathbf{px}\|_2 \ln(\beta))^t\,(1+\ell_{\mathcal{M}_p}(\mathbf{px})\ln(\beta))^{1-t}\right)^{-2} \quad \text{for } \; 0 \le t \le 1 \; \text{ and } \; i = 1,\dots,3\,.$$

In the numerical examples, we generally consider $t = \frac{1}{8}$. From point $\mathbf{p}$, a well-graded smooth continuous mesh is defined all over domain Ω by: $(\mathcal{M}_p(\mathbf{x}))_{\mathbf{x}\in\Omega}$ with $\mathcal{M}_p(\mathbf{x}) = {}^t\mathcal{R}\,\Xi(\mathbf{px})\,\Lambda\,\mathcal{R}$. The reduced metric at a given point $\mathbf{x}$ of domain Ω is given by the strongest size constraint imposed by the metric at $\mathbf{x}$ and by the spanned metrics (parametrized by the given size gradation) of all the other points of the domain at $\mathbf{x}$:

$$\widetilde{\mathcal{M}}(\mathbf{x}) = \left(\bigcap_{\mathbf{p}\in\Omega} \mathcal{M}_p(\mathbf{x})\right) \cap \mathcal{M}(\mathbf{x})\,.$$

In the case of a mesh $\mathcal{H}$ supplied with a discrete metric field given at its vertices, each vertex provides a metric to all the other vertices that imposes its size constraints in all directions. The metric reduction is thus performed with all mesh vertices. To avoid this quadratic complexity algorithm, the mesh gradation problem is approximated with a linear complexity algorithm based on mesh edges. More precisely, the size correction is performed edge by edge. Let $\mathbf{pq}$ be an edge of the mesh with endpoints metrics $\mathcal{M}(\mathbf{p})$ and $\mathcal{M}(\mathbf{q})$. We define the metrics $\mathcal{M}_q(\mathbf{p})$ and $\mathcal{M}_p(\mathbf{q})$ with growth at both extremities of the edge which are given by Relation (29). Then, the reduction is performed for each vertex by a metric intersection. The information is propagated in the whole domain from an iterative algorithm depicted in Algorithm 3. To design a pseudo-exponential size variation law, the gradation coefficient β is increased while getting farther from a vertex. The gradation coefficient is increased by a factor α at each step as stated in Algorithm 3. This algorithm is topology-dependent because the increased gradation coefficient is not a function of the distance but it is a function of a topological distance, *i.e.,* the ball order. Therefore, to reduce the dependency on the mesh topology, the edges are randomly treated.

The result of the metric field size gradation is illustrated in Figure 2 where the metric fields are shown before and after the gradation process. The mesh gradation clearly regularizes the initial non-smooth metric field.

Algorithm 3 Metric Field Size Gradation

While (Correction = 1)

1. Correction = 0
2. Copy current metric field $\mathcal{M}$ in $\mathcal{M}^{new}$
3. Loop over the edge of $\mathcal{H}$ in a random order

   Let $\mathbf{pq}$ be the current edge;

   (a) Grow vertices metric to both edge extremities:
   $\mathcal{M}(\mathbf{q}) \longrightarrow \mathcal{M}_q(\mathbf{p})$ and $\mathcal{M}(\mathbf{p}) \longrightarrow \mathcal{M}_p(\mathbf{q})$;

   (b) Apply the reduction process to each vertex:
   $\mathcal{M}^{new}(\mathbf{p}) = \mathcal{M}^{new}(\mathbf{p}) \cap \mathcal{M}_q(\mathbf{p})$ and $\mathcal{M}^{new}(\mathbf{q}) = \mathcal{M}^{new}(\mathbf{q}) \cap \mathcal{M}_p(\mathbf{q})$;

   (c) If one metric is modified then correction = 1;

   EndEdgeLoop
4. Increase the size gradation factor to $\beta \leftarrow \alpha\,\beta$ with $\alpha > 1$;

EndWhile

5.5 The local adaptive remesher: AMG

The adaptive remesher used in this paper is based a combination of generalized standard operators (insertion, collapse, swap of edges and faces). The generalized operators are based

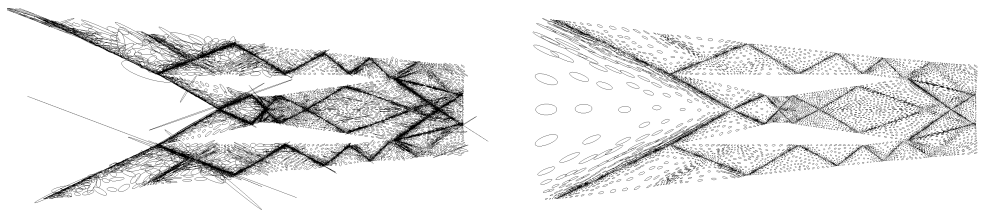

Figure 2: Global view of the metric field at each vertex, the unit ball scaled by one-fourth of each metric is plotted. Left, the initial metric field computed with the error estimate. Right, the new metric field after gradation. It clearly illustrates that mesh gradation regularizes the metric field.

on recasting the standard operators in a cavity framework [31]. Additional modifications on the cavity allow to either favor a modification, that would have been rejected with the standard operator, or to improve the final quality by combining automatically many standard operators at once. In addition, the CPU time is also improved and becomes independent of the current modification. The unit speed is around 20,000 points inserted or removed per second on Intel i7 architecture at 2,7Ghz. For robustness purpose, both the surface and the volume mesh are adapted simultaneously, and each local modification is checked to verify that a valid mesh is obtained. For the volume, the validity consists in checking that each newly created element has a strictly positive volume. For the surface, the validity is checked by ensuring that the deviation of the geometric approximation with respect to a reference surface mesh remains within a given tolerance [14].

For unsteady simulations, the mesh adaptation becomes critical as the CPU time of the simulation depends on the quality of the worse element. Indeed, when an explicit time-stepping is used, the minimal time-step governs the speed of the simulation. Consequently, the minimal size (or height) generated during the remeshing process may impact drastically the CPU time. If the generated size if 0.01 of the minimal target, then the while CPU time will be multiplied by 100. To overcome this issue, we add an additional control based on the height of the tetrahedra. We start from the definition of the minimal height of a tetrahedron:

$$h^2 = \frac{1}{3}\frac{V}{S_{max}}, \tag{30}$$

where h is the minimal height, V the volume and S_{max} the maximal area of the faces. For each provided metric, we consider then the regular tetrahedron of side h_1, h_2, h_3, where $(h_i)_i$ are the unit lengths along the eigenvectors of the metric. Then, assuming that the sizes may be in the range $[\frac{1}{\sqrt{2}}h_i, \sqrt{2}h_i]$, we can estimate the global minimal height h_{tar} using Relation (30). A mesh modification is then rejected if the minimal height of the new set of tetrahedra is lower than h_{tar} and the minimal height of the initial set of elements. Numerical experiments have proven that this additional constraint does not have a negative impact on the level of anisotropy while preserving an optimal CPU time-step for the flow solver.

6 Parallelization of the mesh adaptation loop

6.1 A shared memory multi-threaded parallelization

The main motivation is to take advantage of today's ubiquitous multi-core computers in mesh adaptive computations. However, the adaptive platform is highly heterogeneous as it contains several codes components that have different internal databases and that consider different numerical algorithms. Indeed, the time-accurate mesh adaptation loop involves four codes:

- `Wolf` the second order Finite-Volume flow solver
- `Metrix` to compute the continuous space-time mesh and perform the metric fields gradation

- `AMG` the local adaptive remesher based on the cavity operator
- `Interpol` for the $\mathbb{P}^1$-conservative solution transfer.

Two different strategies have been used to parallelize the different steps considering the **shared memory** paradigm. First, a semi-automatic parallelization based on the p-thread API and, second, a pipelining strategy.

P-threads parallelization The first strategy is an intrusive parallelization of the code using the p-threads paradigm for shared-memory cache-based parallel computers. One of the main assets of this strategy resides in a slight impact on the source code implementation and on the numerical algorithms. This strategy is applied to the flow solver and to the error estimate code. Parallelization is at the loop level and requires few modifications of the serial code [32, 33]. However, to be efficient, this approach requires a subtle management of cache misses and cache-line overwrites to enable correct scaling factor for loop with indirect addressing. The key point is to utilize a renumbering strategy based on Hilbert space-filling curves to minimize them [3]. Indeed, space-filling curves (SFCs) are mathematical objects that exhibit clustering properties very desirable in the context of renumbering algorithms [38]. Then, for loops with indirect addressing, the main block is split into many - several more than the number of threads - small blocks and dynamic scheduling is used to reduce concurrent memory access issues.

Pipelining parallelization After Step 3 of Algorithm 1, all continuous meshes, *i.e.,* unsteady metrics, for each sub-interval have been evaluated. Therefore, the metric gradation and the generation of the new adapted meshes for all sub-intervals can be done at the same time. The second strategy consists of using a pipelining parallelization: each available processor manages these two tasks in serial for a given sub-interval. When a processor is finished, it handles again these two tasks for another sub-interval as long as there are meshes to be generated. In this way, the local remesher and the metric gradation are run in parallel on the same computer or in a distributed manner on an heterogeneous architecture.

Data communication Our implementation of the mesh adaptation platform considers independent dedicated codes for each stage of the adaptation loop. The main drawback of this strategy is that between two stages, one code writes the data (e.g. the mesh and the fields) out-of-core and the next code reads them back and builds its internal database. This results in a larger part devoted to I/O as compared to the all-in-one approach. But, the CPU time for the I/O is generally negligible with respect to the global CPU time. The advantage of the proposed strategy is its flexibility. Each code can be developed independently with its own programming language and its own optimal internal database. Moreover, each code is interchangeable with another one, only the I/O between the different codes need to be compatible. To reduce communication, the amount of transfer ofout-of-core data is minimized and binary files are used.

6.2 Parallel performance

Parallel performance has been analyzed on two different multi-cores computers with different processors and memory access speeds:

- Computer 1:
  - 2 chips: Xeon E5-2670 10 cores 2.5 GHz
  - both chips are connected by 2 QPI links with a speed of 16 GB/s
- Computer 2:
  - 4 chips: Xeon E7-4850 10 cores 2 GHz
  - all chips are connected to all by 1 QPI link with a speed of 16 GB/s

The selected test case is the spherical blast of Section 7.3. First, the parallel efficiency of the flow solver and the solution transfer code is analyzed on one sub-interval. Then, timings for the full time-accurate mesh adaptation loop are given for two fixed-point loop iterations. Parallel timings are compared to the CPU time in serial on the same mesh(es), thus strong speed-ups are analyzed.

Currently, all distributed architectures have hyper-threading capabilities. The main function of hyper-threading is to increase the number of independent instructions in the pipeline. With hyper-threading, one physical core appears as two (or more) processors to the operating system, which can use each core to schedule two (or more) processes at once. If the memory resources for one process are not available, then another process can continue if its memory resources are available. It means that hyper-threading hides process memory latency and cache misses. As a result, super-scalar speed-ups can be obtained with respect to the serial code because of a large reduction of the memory latency and cache misses [33]. Here, for each parallel run, hyper-threading has been used by launching a number of threads equal to twice the number of cores. For instance, the 4 HT run means than 4 cores have been used but 8 threads have been launched. The benefits of hyper-threading are clearly visible in the parallel timings (see tables below) by comparing the serial run and the parallel run on one core with hyper-threading, *i.e.,* the 1HT columns. Hyper-threading is also mandatory to achieve good speed-up on multi-chips architectures by counterbalancing slower speed memory access between chips.

Flow solver For the flow solver profile, the solution is computed from a-dimensioned time 0.6945 to 0.7 which corresponds to the last sub-interval when $n_{adap} = 128$. The mesh size is $2,173,612$ vertices, $13,037,975$ tetrahedra and $75,090$ boundary triangles. To reach final time, the solver perform 36 Runge-Kutta iterations with a 5-steps scheme, leading to a total of 180 time-steps. I/Os and initialization are not taken into account in the timings. Timings and speed-ups for Computer 1 and 2 are summarized in Table 1 and 2, respectively. We observe an almost perfect strong speed-up up to 1 chip with 10 cores using hyper-threading. Speed-up decreases when 2 chips are used due to memory access speed between chips, but it still remains good and very good on Computer 1 and 2, respectively. However, speed-ups drop drastically on 4 chips on Computer 2 because there is not enough memory links between the chips.

Nbr. of cores	Serial	1 HT	2 HT	4 HT	8 HT	10 HT	20 HT
Timings (sec.)	1,054	878	423	236	133	112	71
Speed-up	1.0	1.2	2.5	4.5	7.9	9.4	14.9

Table 1: Flow solver timings and speed-up on computer 1 up to 20 cores with hyper-threading.

Nbr. of cores	Serial	1 HT	2 HT	4 HT	8 HT	10 HT	20 HT	40 HT
Timings (sec.)	2,072	1,506	759	393	228	193	121	117
Speed-up	1.0	1.4	2.8	5.2	9.1	10.7	17.1	17.7

Table 2: Flow solver timings and speed-up on computer 2 up to 40 cores with hyper-threading.

Solution transfer The five solution fields at a-dimensioned time 0.6945 of the spherical blast problem are interpolated. The background mesh size is $2,166,190$ vertices and $12,993,399$ tetrahedra and the new mesh size is $2,173,612$ vertices and $13,037,975$ tetrahedra. For that case, 226 millions tetrahedron-tetrahedron intersections have been computed and 1.7 billions tetrahedra have been generated to mesh the intersections. In the timings analysis, I/Os and initializations are not taken into account. Timings and speed-ups for Computer 1 and 2 are summarized in Table 3 and 4, respectively. We notice that the speed-ups are excellent on both computers and even super-linear for a low number of cores thanks

to the hyper-threading which reduces memory latency. However, for computer 2, we notice when more chips are used, the speed-up degrades. This is mainly due to slower memory access between the chips (only one link between each).

Nbr. of cores	Serial	1 HT	2 HT	4 HT	8 HT	10 HT	20 HT
Timings (sec.)	1,435	1,071	562	301	158	126	72
Speed-up	1.0	1.3	2.6	4.8	9.1	11.3	19.93

Table 3: Solution transfer timings and speed-up on computer 1 up to 20 cores with hyper-threading.

Nbr. of cores	Serial	1 HT	2 HT	4 HT	10 HT	20 HT	40 HT
Timings (sec.)	2,087	1,625	1,011	413	193	115	63
Speed-up	1.0	1.3	2.0	5.0	10.8	18.1	33.1

Table 4: Solution transfer timings and speed-up on computer 2 up to 40 cores with hyper-threading.

Full adaptation loop Let us now analyze the timings for the full adaptation loop meaning that I/Os, file manipulations and code initializations are included. Two global fixed-point iterations are performed and the time frame has been split into 16 sub-intervals, *i.e.*, $n_{adap} = 16$. The space-time complexity prescription leads to an average spatial mesh size for each sub-interval of $1,318,667$ vertices and $2,048,968$ tetrahedra. Timings and speed-ups for Computer 1 are given in Table 5. Note that the CPU time dedicated to the resolution of the flow equations represents about 90 % of the total CPU time, which demonstrates the limited CPU time overhead due to the adaptation loop. This very limited cost has to be compared with the enormous gains in terms of solution quality and mesh optimality, as long as with the ability to manage unsteady simulations complexity at will.

Nbr. of cores	Serial	2	4	8	20 HT
`Wolf` & `Interpol` total timings (min.)	2,014	1,060	586	307	140
`Wolf` & `Interpol` total speed-up	1.0	1.9	3.4	6.6	14.4
`AMG` & `Metrix` total timings (min.)	152	83	45	26	16
`AMG` & `Metrix` total speed-up	1.0	1.8	3.4	5.9	9.5
Adaptation loop total timings (min.)	2,166	1,143	631	333	156
Adaptation loop total speed-up	1.0	1.9	3.4	6.5	13.9

Table 5: Full adaptation loop timings and speed-up on computer 1 up to 20 cores with hyper-threading.

7 Space-time convergence analysis

The goal of this section is to analyse the space-time convergence of the adaptive process and to emphasize the adaptation of the time domain. The temporal adaptation is achieved by controlling the number of sub-intervals (defined by n_{adap}) while using the unsteady mesh adaptation loop presented in Algorithm 1.

We first demonstrate the loss of convergence order if the time domain is not adapted, then we describe how the time domain can be adapted using the unsteady mesh adaptation algorithm. Then, we present our choices for the computations of the space-time error and the simulation complexity. Finally, a space-time convergence analysis is carried out for two simulations. The first one is a 3D blast problem inside a simple geometry (a box) where a detailed analysis can be carried out. The second one is a realistic simulation of a blast with a geometry representing a city, demonstrating that such analysis can also be made on relatively complex real-life problems.

7.1 Adaptation of the time domain

When a convergence analysis is carried out, the rate of reduction of the error is analyzed with respect to the increase of the mesh size (e.g. the mesh size). A convergence rate α is obtained if we have:

$$\|u - u_h\| \leq Cst\ N^{-\frac{\alpha}{d}} \tag{31}$$

where u denotes the exact solution, u_h the approximate (*i.e.,* discretized) one, N stands for the mesh size (number of vertices) and d the dimension of the computational domain.

For the steady case, a loss of convergence order (it drops to order one or less) generally occurs in the presence of steep gradients (Navier-Stokes equations), or genuine discontinuities or singularities (Euler equations) in the flow, despite the use of a tried-and-tested spatially high-order method. Indeed, the numerical method is second-order far from singularity or discontinuity, and first-order at singularity or discontinuity vicinity. The computed mesh convergence order obtained in practice on uniformly refined meshes differs from the one expected in theory. Theoretically, Relation (6) demonstrates that an asymptotic second-order mesh convergence is obtained for smooth functions thanks to mesh adaptation in L^p norm. In several works [4, 26, 29], it has also been shown that the theoretical convergence order of numerical schemes - here a state-of-the-art second-order shock-capturing solver - is also recovered thanks to anisotropic mesh adaptation, even if the flow field exhibits singularities, genuine discontinuities, or steep gradients. It is because, in order to reduce the error by a factor four, mesh adaptation requires : a mesh size four times smaller in the singularity vicinity, a mesh size four times smaller in the direction normal to the discontinuity and two times smaller in the other two directions in the discontinuity vicinity, and a mesh size two times smaller in smooth regions. Thus, the overall mesh size is only increased by a factor 2^d.

When dealing with unsteady simulations, the space-time error convergence has to be analyzed with respect to the space-time mesh size (by adding the time discretization, *i.e.* $d = 4$). In our case, an explicit time integration scheme is considered, thus the time-step is governed by a CFL condition and is proportional to the mesh size, see Relation (17). Therefore, the time-step is divided by a coefficient a when the mesh size is divided by a coefficient a. We now detail the evolution of the space-time mesh size when the accuracy is increased. We still assume that the space-time approximation is second-order far from singularity or discontinuity, and is first-order at singularity or discontinuity vicinity.

Let us consider the simulation of a 1D constant planar shock wave (*i.e.,* a step function) moving at constant speed until final time $T = 1$. In that case, we have $d = 2$. The number of sub-intervals is set to n_{adap} and the sub-intervals have the same time length $\Delta t = 1/n_{adap}$. An average number of vertices equal to nx is prescribed for each sub-interval. As the shock moves at a constant speed and is constant in time, the following mesh characteristics can be deduced:

- each sub-interval is meshed with nx vertices
- the space interval $[0, \frac{1}{n_{adap}}]$ corresponds to the area swept by the discontinuity for the first sub-interval, thus the nx vertices are located in that region[1]. Then, the shock wave swept the interval $[\frac{1}{n_{adap}}, \frac{2}{n_{adap}}]$ which is refined with the nx vertices at second sub-interval and so on
- as the same mesh size is prescribed for each sub-interval, nt time-steps are performed for each sub-interval.

The resulting adapted space-time mesh is illustrated in Figures 3 and 4 (left). In that case, we end-up with a total space-time mesh size of

$$N_{st} = n_{adap} \times (nx \times nt)\,.$$

Now, we want to increase the mesh size to perform a convergence analysis. We assume the number of sub-intervals remains constant. If the spatial size is divided by two then the

[1] The number of vertices located outside that region can be neglected.

time-step is also divided by two, see right mesh in Figure 3, then the total space-time mesh size of the refined mesh is

$$n_{adap} \times (2\,nx \times 2\,nt) = 4\,N_{st}\,.$$

For this space-time mesh, the error is divided by 2 because the space-time approximation is first-order at the discontinuity vicinity. As N_{st} is multiplied by 4 (and we have $d = 2$), we obtain $\alpha = 1$ in Relation (31) which means a convergence at first order. This is due to the non-adaptation of the time axis leading to a non-optimal mesh adaptation process.

To adapt the time axis, we propose to multiply by two the number of sub-intervals and to keep the same average number of vertices nx on each sub-interval. As the size of the sub-interval is divided by two, the spatial size and the time-step are also divided by two, see right mesh in Figure 4. In that case, the total space-time mesh size of the refined mesh is:

$$2\,n_{adap} \times (\,nx \times\, nt) = 2\,N_{st}\,.$$

We observe that we achieved the same accuracy as previously but the space-time mesh size has been increased by a factor two instead of four. Now, the error is divided by 2 for N_{st} multiplied by 2, we thus obtain $\alpha = 2$ in Relation (31) meaning a convergence at second order. By doing so, we have performed an adaptation of the time axis and recover the second order convergence of the numerical scheme similarly to the steady case.

The same analysis can be done for the general case in nD where we assume that the regular part of the solution contributes to the space-time error at the same order as the singular (discontinuous) part of the solution. Following the above comments for the steady case and the time adaptation of the mesh, for a nD unsteady simulation (hence $d = n + 1$), the space-time error is divided by four if the adapted mesh has 2^n more vertices in smooth regions, is four times finer in singularity vicinity, and is four times finer orthogonally to discontinuities (and two times smaller in the other directions), so that the time-step is also four times smaller. Therefore, the space-time mesh size is increased by a factor 4×2^n. As convergence order α verifies $\varepsilon \sim N^{-\frac{\alpha}{d}}$, the reduction of the error and the increase in mesh size give:

$$\frac{1}{4} = (4 \times 2^n)^{-\frac{\alpha}{d}} = (\frac{1}{4})^{\frac{\alpha}{d}\left(\frac{n}{2}+1\right)} \quad \Rightarrow \quad \alpha = \frac{2\,d}{n+2}\,.$$

We obtain $\alpha = \frac{4}{3}$, $\frac{3}{2}$ and $\frac{8}{5}$ in 1D, 2D and 3D, respectively.

Remark 7.1. If the "regular" error is a lot smaller that the "singular" error, as all the mesh vertices will focus on singular regions, we can observe an arbitrary large order of convergence.

Remark 7.2. This simplified analysis highlights the usefulness of adapting the number of sub-intervals to carry out a convergence analysis. Note that if the mesh is adapted at *each* flow solver iteration (which is impracticable for real-life applications), the same time axis adaptation is automatically performed. Indeed, due to the CFL condition, dividing mesh size by two requires twice the solver iterations and thus the generation of twice the meshes. In consequence, the proposed adaptation of the time axis is logical.

Remark 7.3. Most unsteady adaptation algorithms in the literature consist of adapting the mesh at each n (n small) solver iterations - if not at *each* iteration - and one can legitimately question our choice of adapting on time sub-intervals. The above analysis and Remark 7.2 show that using time sub-intervals, limits considerably the number of generated meshes and thus the number of solution transfers. This is definitely a major problem with respect to the accumulation of transfer errors [5] and pleads in favor of using time sub-intervals for consistent space-time adaptation.

However, to perform this consistent time adaptation, the number of sub-intervals is divided by two when spatial size is divided by two. The direct consequence is that twice as many meshes are generated and therefore, twice as many solution transfers are performed. To avoid an accumulation of transfer errors, which can definitely hinder convergence, it is mandatory to use evolved transfer operators which are especially designed to limit these errors. The $\mathbb{P}_1$-conservative interpolation scheme recalled in Section 5.2 achieves this goal in a very efficient manner [5].

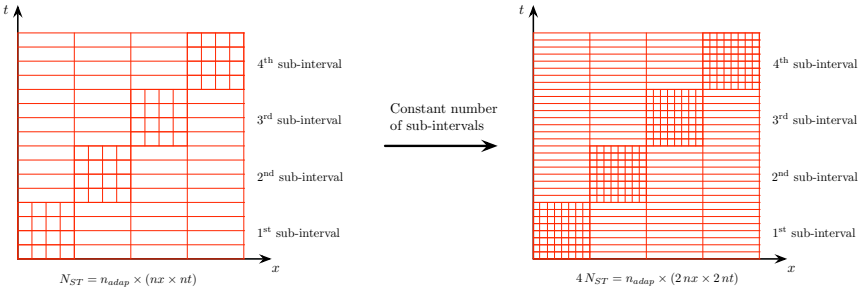

Figure 3: *Illustration in 1D of the refinement of the space-time mesh when the space-time mesh size is increased and the number of sub-intervals is kept constant.*

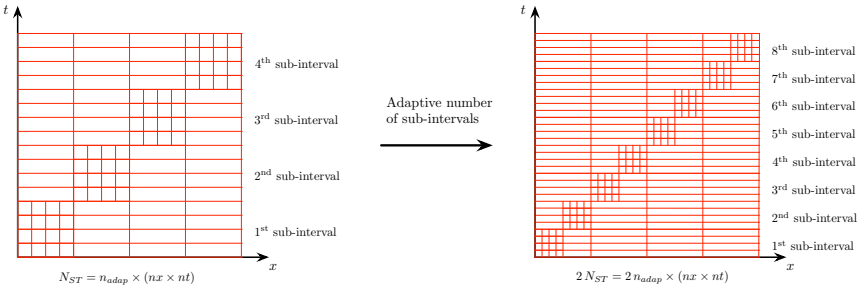

Figure 4: *Illustration in 1D of the refinement of the space-time mesh when the space-time mesh size is increased and the number of sub-intervals is increased too.*

7.2 Computation of the space-time error and associated complexity parameters

To analyze the accuracy of each simulation and to perform a convergence analysis, we compute the L^1-norm of the space-time error with respect to a reference solution:

$$err_{st} = \int_0^T \int_\Omega |u_{ref}(\mathbf{x},t) - u(\mathbf{x},t)| \mathrm{d}\mathbf{x}\mathrm{d}t \approx \sum_{i=1}^{n_{adap}^{ref}} \Delta t \left(\sum_{j=1}^{N_{tet}^{ref}(i)} |K_j| \, |u_{ref}(G_j, t^{i+1}) - u(G_j, t^{i+1})| \right) \tag{32}$$

with the notations:

- n_{adap}^{ref} is the number of reference adapted meshes (sub-intervals) used for the simulation
- $\Delta t = \frac{T}{n_{adap}^{ref}}$ is the sub-intervals time length
- $N_{tet}^{ref}(i)$ is the number of tetrahedra of the ith adapted mesh, denoted $\mathcal{H}_i^{ref}$, used to compute the reference solution for sub-interval $[t^i,\, t^{i+1}]$
- $|K_j|$ is the volume of the jth tetrahedron of $\mathcal{H}_i^{ref}$
- $u_{ref}(G_j, t^{i+1})$ and $u(G_j, t^{i+1})$ are the reference solution and the solution at j^{th} tetrahedron barycenter at time t^{i+1}, respectively.

To run the simulation, the theoretical total space-time complexity is prescribed as described in Section 5.1: $\mathcal{N}_{st} = n_{adap}\mathcal{N}_{avg}$. However, this value cannot be used for the convergence analysis. For this analysis, we propose to observe the convergence of the space-time error given by Relation (32) with respect to three parameters:

1. The total number of spatial vertices of the simulation run on n_{adap} adapted meshes which the sum of the number of vertices of each sub-interval mesh:

$$N_{spatial} = \sum_{i=1}^{n_{adap}} N^i \,.$$

In that case, the time discretization is not taken into account and we assume that only one time-step is performed on each sub-interval. This parameter is consistent with the theoretical prescription which does not take into account the time discretization.

2. The total number of space-time vertices of the simulation run on n_{adap} adapted meshes:

$$N_{st} = \sum_{i=1}^{n_{adap}} n_{iter}^{i} \times N^{i} \,,$$

where the time discretization corresponds to the number of time-steps of the flow solver at each sub-interval, e.g. n_{iter}^{i} is the number of time-steps for the ith sub-interval.

3. The effective CPU time of the simulation for the last two fixed-point iterations.

These parameters deserve some comments for a correct interpretation of the results in Sections 7.3 and 7.4.

Remark 7.4. The first parameter is consistent with the choices made in Section 5.1 to define the optimal continuous mesh given by Relation (25), and thus, the user prescription for the mesh complexity. However, as we assume that only one time-step is done for each sub-interval, the theoretical analysis made in Section 7.1 does not hold anymore. Indeed, this parameter is blind for any adaptations of the temporal domain, *i.e.,* the number of time-steps. Therefore, for this parameter, it is not advantageous to perform more adaptations (e.g. using more sub-intervals), and all the more so that the solution interpolation stage is source of errors.

Remark 7.5. The second parameter represents the real size of the discrete space-time mesh, so it is consistent with the theoretical analysis of Section 7.1, e.g. doing more mesh adaptation should improve the space-time convergence order of the mesh adaptation scheme. Notice that the time discretization highly depends on the generated discrete meshes[2] and thus may have some fluctuations due to the discrete representation.

Remark 7.6. The third parameter is a pragmatic parameter because it represents the real size of the domain space-time mesh used to run the simulation (like the first parameter) but also all the code's cost associated with the creation of this optimal space-time mesh in the adaptation process. It will show what is the most efficient strategy to apply in practice with the code considered in this work.

7.3 Spherical blast

The first example is a spherical Riemann problem between two parallel walls simulating a blast. The computational domain is a box of size $[-1.5, 1.5] \times [-1.5, 1.5] \times [0, 1]$. Initially, the gas is at rest with density $\rho_{out} = 1$ and pressure $p_{out} = 1$ everywhere except in a sphere centered at $(0, 0, 0.4)$ with radius 0.2. Inside the sphere the parameters are $\rho_{in} = 1$ and $p_{in} = 5$. For both regions, we have $\gamma = 1.4$. The initial pressure jump results in a strong outward moving shock wave, an outward contact discontinuity and an inward moving rarefaction wave, see Figure 5 (left). The main feature of the solution is the interactions between these waves. Another significant feature is the development of a low density region in the center of the domain. The solution remains cylindrically symmetric throughout the simulation and is computed until a-dimensioned time $T = 0.7$.

For all the adaptive simulations, the density of the flow is chosen as the sensor variable and the space-time interpolation error on the sensor is controlled in L^2 norm. Twenty solution samples are used to compute the hessian-metric $\mathbf{H}_{L^1}$ given in Section 5.1.

The adaptive *reference solution* has been computed using $n_{adap}^{ref} = 128$ sub-intervals, *i.e.,* the number of adapted meshes used to run the simulation, and a theoretical average complexity of $\mathcal{N}_{avg} = 640,000$ per sub-interval. This represents a total space-time complexity $\mathcal{N}_{st}$ equal to 82 million. Practically, the discrete meshes - used to compute the reference

[2] An explicit time integration is considered and the time-step is linked to the smallest element height of the mesh.

solution - have an average number of spatial vertices of $N_{avg}^{ref} = 1,712,282$ vertices, detailed statistics about some of the adapted meshes are given in Table 6. The total number of space-time vertices used to compute the reference solution is $N_{st}^{ref} = 11$ billion.

In order to perform the convergence study, 30 adaptive simulations have been run where the two varying parameters are:

- The number of adapted meshes (sub-intervals): $n_{adap} = \{4\ ;8\ ;16\ ;32\ ;64\ ;128\}$
- The average complexity: $\mathcal{N}_{avg} = \{40,000\ ;80,000\ ;160,000\ ;240,000\ ;320,000\}$.

Thus, the total space-time complexity $\mathcal{N}_{st}$ varies between $160,000$ and 41 million. Running all these simulations makes it possible to analyze the impact of increasing the theoretical complexity of the mesh on the solution accuracy either at fixed number of adaptations while increasing the average complexity or at fixed average complexity while increasing the number of adaptations.

Solution - density field - evolution at a-dimensioned time 0.175, 0.35, 0.525, 0.7, and adapted meshes evolution for sub-intervals $32, 64, 96, 128$ for the adaptive simulation with parameters $n_{adap} = 128$ and $\mathcal{N}_{avg} = 320,000$ are presented in Figure 5. These pictures illustrate the complex behavior of the physical phenomena (shock waves, contact discontinuities, rarefaction waves, low density region) and the many interactions between them.

Figure 6 (left) shows the final density field at a-dimensioned 0.7 for different prescribed space-time complexities where $n_{adap} = 16$ and $\mathcal{N}_{avg} = \{40,000\ ;80,000\ ;160,000\ ;320,000\}$. The improvement in solution accuracy is clearly visible as solutions have increasingly details, and the waves become increasingly fine. Figure 6 (right) shows the last sub-interval adapted

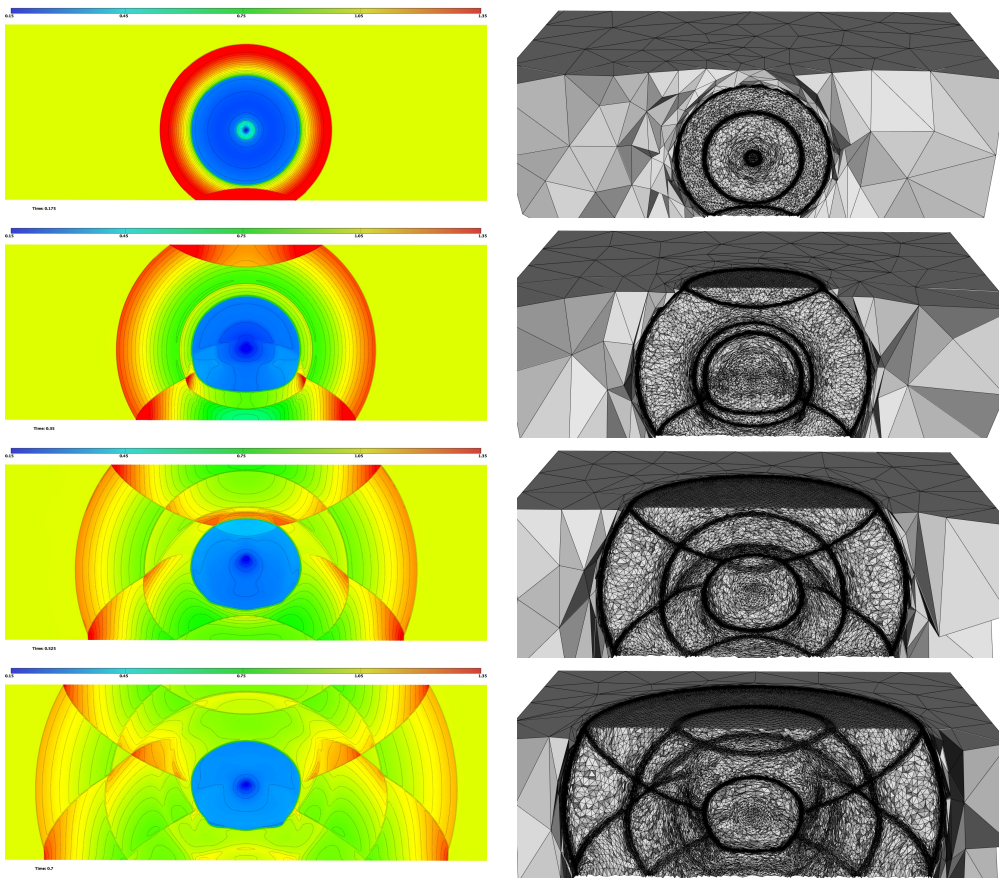

Figure 5: *Spherical blast. Left, solution - density field - evolution at a-dimensioned time* 0.175 *,* 0.35 *,* 0.525*, and* 0.7 *(from top to bottom), and right, adapted meshes evolution for sub-intervals* $32, 64, 96$ *and* 128 *(from top to bottom) for the adaptive simulation with parameters* $n_{adap} = 128$ *and* $\mathcal{N}_{avg} = 320,000$*.*

meshes for the adaptive simulations with $\mathcal{N}_{avg} = 320,000$ and $n_{adap} = \{4\ ;8\ ;16\ ;32\}$. The adaptation of the mesh for the whole time sub-interval is clearly visible. Indeed, the mesh refinement along band-shaped regions, which corresponds to the zone in which physical phenomena evolves during a time sub-interval, are visible. We also observe that the thickness of the refined band-shaped regions reduces when the number of adapted meshes increases, this points out the adaptation of the time axis as explained in Section 7.1.

The convergence curves of the space-time error err_{st} (given by Relation (32)) with respect to the three parameters $N_{spatial}, N_{st}$ and CPU time are given in Figure 7. Two plots also show the convergence for uniform meshes with a uniform size h ranging from 0.04 to 0.00625, and a number of vertices between $249,514$ and $61,230,237$. The number of space-time vertices varies between $12,475,700$ and $48,616,808,178$. Uniform meshes statistics are given in Table 7 (left).

The top plots show the convergence of err_{st} with respect to the number of spatial vertices $N_{spatial}$. Left, convergence curves for five prescribed $\mathcal{N}_{avg}$ with varying n_{adap} (from 4 to 128) are presented. We observe a convergence at order 2 (the black dashed curve is $c\,N^{-\frac{2}{4}}$) for the five curves at fixed $\mathcal{N}_{avg}$ while increasing the number of adaptations. Right, we compare the convergence for prescribed total complexities ranging from $640,000$ to $5,120,000$ when $\mathcal{N}_{avg}$ is fixed to $40,000$ and n_{adap} is increased from 4 to 128, and when n_{adap} is fixed to 16 and $\mathcal{N}_{avg}$ is increased from $40,000$ to $320,000$. We observe a faster convergence when $\mathcal{N}_{avg}$ is increased and n_{adap} is fixed. This is conforming to Remark 7.4 because the number of time-steps is not taken into account in the mesh size.

The middle plots show the convergence of err_{st} with respect to the number of space-time

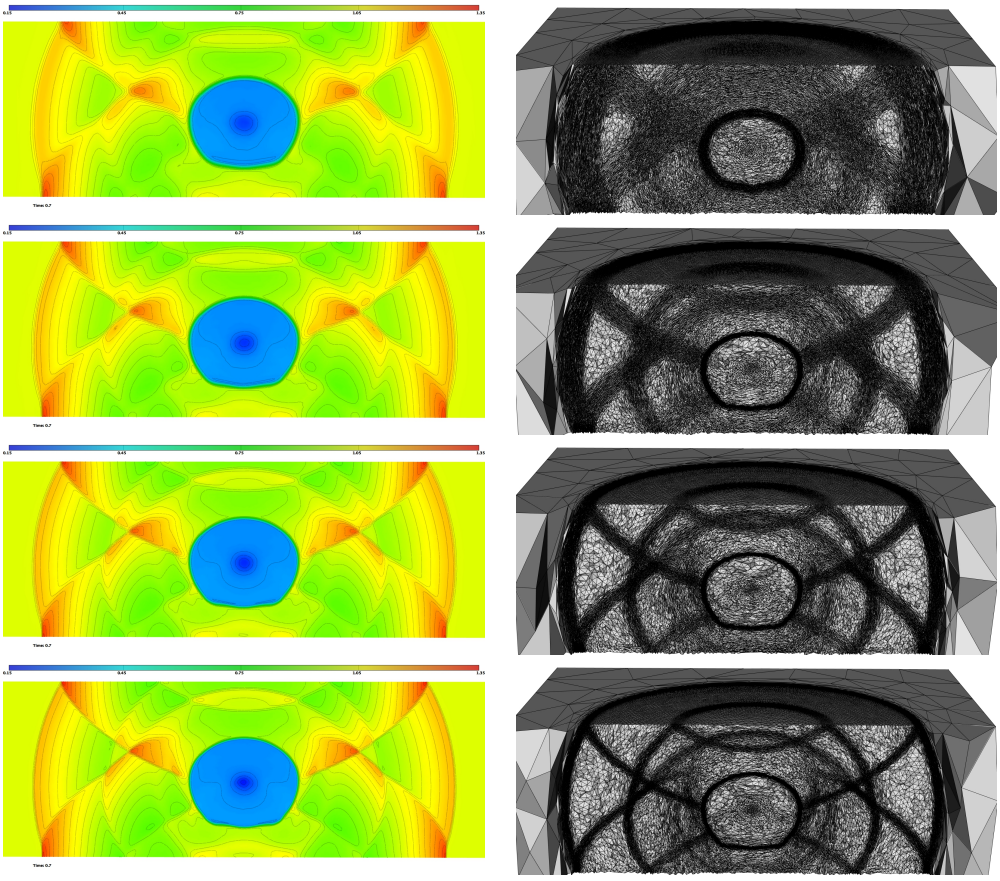

Figure 6: *Spherical blast. Left, final solution at a-dimensioned* 0.7 *for adaptive simulations with* $n_{adap} = 16$ *and* $\mathcal{N}_{avg} = \{40,000\ ;80,000\ ;160,000\ ;320,000\}$ *(from top to bottom). It points out the solution - density field - accuracy improvement while increasing the average theoretical complexity of the adaptive simulation. Right, last sub-interval adapted meshes for adaptive simulations with* $\mathcal{N}_{avg} = 320,000$ *and* $n_{adap} = \{4\ ;8\ ;16\ ;32\}$ *(from top to bottom).*

vertices N_{st}. Left, convergence curves for five prescribed $\mathcal{N}_{avg}$ with varying n_{adap} (from 4 to 128) are presented. All of them point out a convergence at order 3 (the black dashed curve is $c\,N^{-\frac{3}{4}}$). This order of convergence is larger than the one expected by the theory in Section 7.1. It should be of the order of 8/5. But, in this simulation, singular phenomena are preponderant with respect to regular phenomena. As explained in Remark 7.1, an arbitrary large order can be obtained in such case. On the contrary, we notice that the series of uniform meshes converge only at order one which is in accordance with the theory. The lack of accuracy of

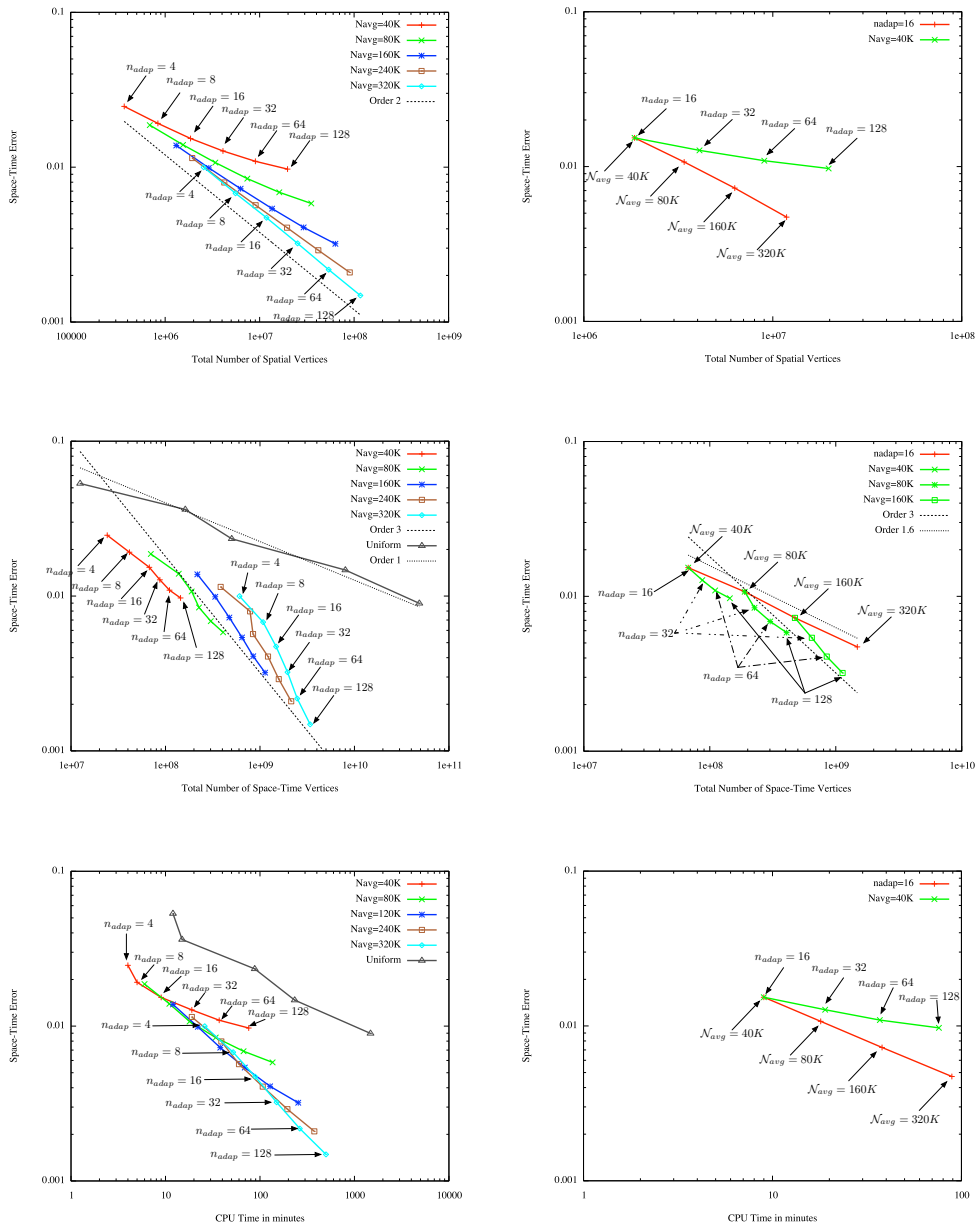

Figure 7: *Spherical blast. Convergence curves of the space-time error with respect to the number of spatial vertices (top), the number of space-time vertices (middle), and the CPU time (bottom). Left, convergence curves for five prescribed average complexity with varying number of adapted meshes (sub-intervals). Right, convergence comparison between fixed average complexity with varying number of adapted meshes and fixed number of adapted meshes with varying average complexity.*

the uniform meshes solutions with respect to the adaptive reference solution is emphasized in Table 7 (right) where final density solution extractions along a line are shown. For each adaptive curve, we observe that the curve directly to its left is always below, and the curve directly to its right is always above. This means that a better convergence rate is obtained by increasing n_{adap} than by increasing $\mathcal{N}_{avg}$. This is emphasized in the right plot where we compare the convergence for different prescribed total complexities when $\mathcal{N}_{avg}$ is fixed and n_{adap} is increased, and when n_{adap} is fixed and $\mathcal{N}_{avg}$ is increased. The convergence order is 3 when n_{adap} is increased while it is 1.6 when n_{adap} is fixed. This result is in accordance with Remark 7.5. Moreover, Section 7.1 theoretical analysis points out that, for the same prescribed total space-time complexity, the space-time discrete mesh size increases faster when n_{adap} is fixed and $\mathcal{N}_{avg}$ is increased than when $\mathcal{N}_{avg}$ is fixed and n_{adap} is increased. This is clearly illustrated in the right plot.

The bottom plots show the convergence of err_{st} with respect to the CPU time. These plots are more pragmatic because if the theory suggests to increase the number of adapted meshes to improve the convergence order, this has a cost in practice because:

- It increases the number of IOs to communicate between code
- The flow solver has to build its data base more often
- The local remesher has to generate more meshes
- The solution needs to be interpolated more times, and thus more error due to this transfer is accumulated.

All curves are nearly aligned (except the red one with the lowest prescribed average complexity) which means that whatever the set of parameters $\{n_{adap}, \mathcal{N}_{avg}\}$, they will achieve the same space-time error for the same CPU time. However, these plots suggest having a high $\mathcal{N}_{avg}$ value before increasing n_{adap}. We clearly see that the solution accuracy degrades significantly for lower $\mathcal{N}_{avg}$ values ($40,000$ and $80,000$) when a large number of adapted meshes is considered, and also slightly degrades for middle $\mathcal{N}_{avg}$ values ($120,000$ and $240,000$) when n_{adap} equals 128. This is due to the interpolation stage that spoils the solution accuracy even if a conservative interpolation is considered. This accuracy degradation is not visible when the error is plotted with respect to the mesh size (top and middle plots) but it appears when efficiency is concerned. The convergence for the series of uniform meshes is also plotted. It undeniably points out the superiority of the adaptive process in terms of accuracy and CPU time.

Sub-interval	# vertices	# tetrahedra	# triangles	min h	avg h	ratio	quotient
1	$2,870,443$	$17,424,113$	$9,626$	0.0002	0.0065	15	250
32	$1,311,468$	$7,883,090$	$11,582$	0.0003	0.0108	23	463
64	$1,715,800$	$10,285,064$	$34,860$	0.0004	0.0135	21	370
96	$1,908,009$	$11,414,259$	$55,484$	0.0003	0.0163	20	316
128	$2,173,612$	$13,037,975$	$75,090$	0.0003	0.0187	18	268

Table 6: Spherical blast. Adapted meshes statistics used to compute the reference solution: number of vertices, number of tetrahedra, number of triangles, minimal edge size, average edge size, average anisotropic ratio, and average anisotropic quotient.

7.4 A blast in a city

This second example is a purely three-dimensional blast problem in a relatively complex geometry representing a city. In this simulation, shock waves interact with each other and are reflected by the buildings. The city geometry is the same as in [2] with a domain size of $85\,m \times 70\,m \times 70\,m$. Initially, the ambient air is at rest $\rho_{out} = 1$ and $p_{out} = 1$. To simulate the blast, a high pressure and density region is introduced in a half-sphere of radius 2.5 m. In this region, the relevant parameters are $\rho_{in} = 10$, $p_{in} = 25$ and $\mathbf{u}_{in} = \mathbf{0}$. For both regions, we have $\gamma = 1.4$. The solution is computed until a-dimensioned time $T = 15$.

# vertices	# tetrahedra	# triangles	h
$249,514$	$1,482,072$	$41,620$	0.04000
$981,414$	$5,920,992$	$107,206$	0.02500
$3,865,472$	$23,625,435$	$270,146$	0.01575
$15,410,860$	$94,933,056$	$676,270$	0.00100
$61,230,237$	$381,366,990$	$1,725,984$	0.00625

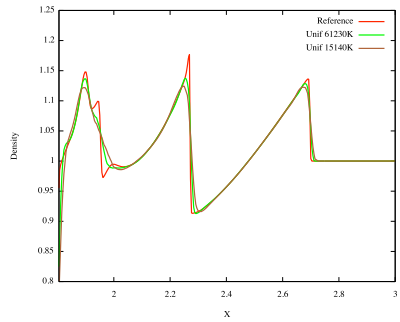

Table 7: Spherical blast. Left, uniform meshes statistics used to compute the solution: number of vertices, number of tetrahedra, number of triangles, and edge size. Right, final density solution extraction along a line for the reference solution (red) and the two solutions computed on the two finest uniform meshes (green and brown).

The density of the flow is chosen as the sensor variable for our mesh adaptation process. The space-time interpolation error on the sensor is controlled in L^2 norm. Twenty solution samples are used to compute the hessian-metric $\mathbf{H}_{L^1}$.

The adaptive *reference solution* has been computed using $n_{adap}^{ref} = 128$ adapted meshes, and a theoretical average complexity of $\mathcal{N}_{avg} = 800,000$ per sub-interval. This represents a total space-time complexity $\mathcal{N}_{st}$ equal to 102 million. Practically, the discrete meshes - used to compute the reference solution - have an average number of spatial vertices of $N_{avg}^{ref} = 3,327,382$ vertices, detailed statistics about some of the adapted meshes are given in Table 8. The total number of space-time vertices used to compute the reference solution is $N_{st}^{ref} = 11$ billion. Figures 8 and 9 show the evolution of the reference solution - density field - at a-dimensioned time 5, 10, 15, and the associated adapted meshes evolution for sub-intervals $43, 86, 128$ on the surface and in a cut plane inside the volume, respectively. It points out the complexity and the unpredictable behavior of the physical phenomena with a large number of shock waves interacting with the geometry, and instabilities developing in the region where the blast was initiated. Thanks to the feature-based mesh adaptation, all shock waves are automatically captured by the adaptation process and properly refined.

In order to perform the convergence study, 18 adaptive simulations have been run where the two varying parameters are:

- The number of adapted meshes (sub-intervals): $n_{adap} = \{4\,;8\,;16\,;32\,;64\,;128\}$
- The average complexity: $\mathcal{N}_{avg} = \{100,000\,;200,000\,;400,000\}$.

Thus, the total space-time complexity $\mathcal{N}_{st}$ varies between $400,000$ and 51 million. Again, running all these simulations makes it possible to analyze the impact of increasing the theoretical complexity of the mesh on the solution accuracy either at fixed number of adaptations while increasing the average complexity or at fixed average complexity while increasing the number of adaptations.

The convergence curves of the space-time error err_{st} with respect to the three parameters $N_{spatial}, N_{st}$ and CPU time are given in Figure 10. Comments on the results are identical to the one made for the previous example in Section 7.3.

For the top plots showing the convergence of err_{st} with respect to the number of spatial vertices $N_{spatial}$, we observe the same order of convergence and a faster convergence when $\mathcal{N}_{avg}$ is increased and n_{adap} is fixed which is conforming to Remark 7.4.

For the middle plots showing the convergence of err_{st} with respect to the number of space-time vertices N_{st}, a convergence at order 2 (the black dashed curve is $c\,N^{-\frac{2}{4}}$) instead of order 3 is observed. This is due to the instability regions that contain regular physical phenomena, and thus singular regions are less preponderant, as discussed in Remark 7.1. The plots again emphasize that a better convergence rate is obtained by increasing n_{adap} than by increasing $\mathcal{N}_{avg}$.

For the bottom plots showing the convergence of err_{st} with respect to the CPU time, we see that again the three curves are nearly aligned signifying that whatever the set of parameters $\{n_{adap}, \mathcal{N}_{avg}\}$, they will achieve the same space-time error for the same CPU time. Another time, we observe a slight accuracy degradation for the lowest average complexity $(100,000)$ when n_{adap} equals 128 suggesting to have a high $\mathcal{N}_{avg}$ value before increasing n_{adap}.

8 Other applications

The proposed adaptive method is generic and can be applied to a large variety of numerical problems. In this section, we show its application to aeronautic and multi-fluid flow simulations.

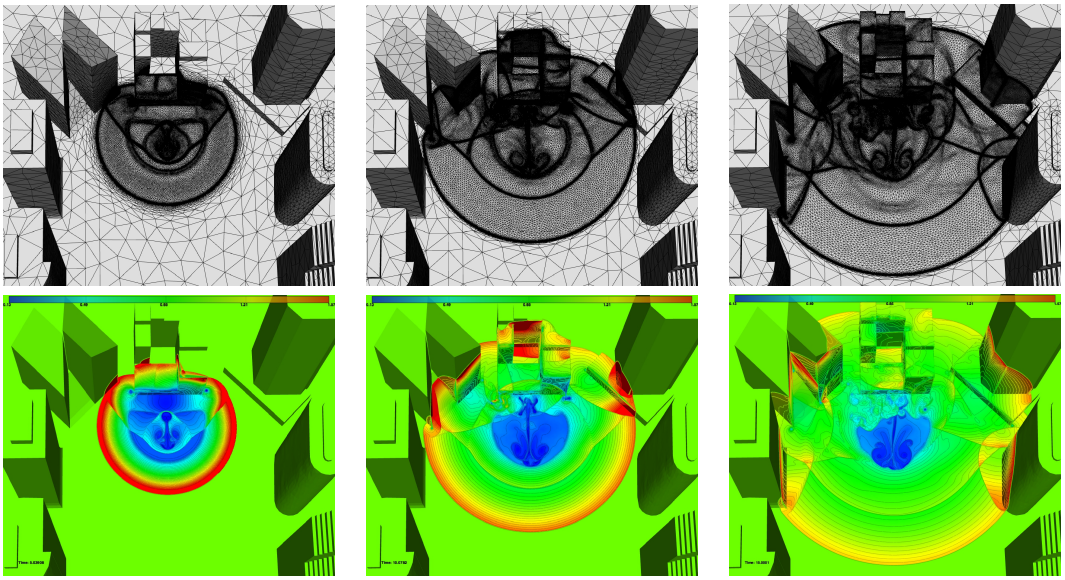

Figure 8: 3D blast in a city. Top, adapted surface meshes and, bottom, density iso-values on the surface. From left to right, meshes and solutions at sub-intervals 43, 86, and 128, respectively.

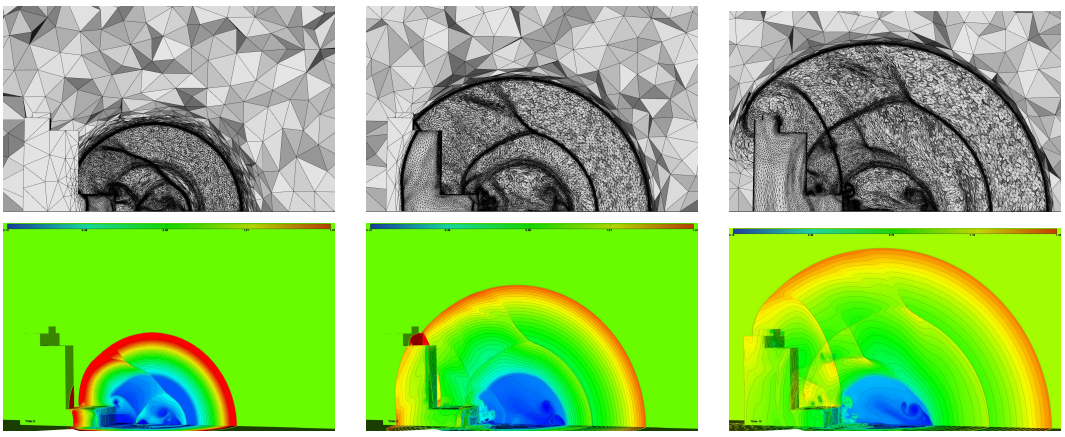

Figure 9: 3D blast in a city. Top, cut in the adapted volume meshes and, bottom, corresponding density iso-values. From left to right, meshes and solutions at sub-intervals 43, 86, and 128, respectively.

Sub-interval	# vertices	# tetrahedra	# triangles	min h	avg h	ratio	quotient
1	$829,275$	$4,927,323$	$44,902$	4 mm	123 mm	12	106
32	$3,218,351$	$19,451,629$	$114,206$	2 mm	149 mm	18	253
64	$3,459,242$	$20,822,072$	$194,166$	5 mm	208 mm	18	270
96	$3,772,838$	$22,675,024$	$266,450$	4 mm	250 mm	16	251
128	$4,187,548$	$25,249,618$	$329,610$	4 mm	288 mm	15	249

Table 8: 3D blast in a city. Adapted meshes statistics used to compute the reference solution: number of vertices, number of tetrahedra, number of triangles, minimal edge size, average edge size, average anisotropic ratio, and average anisotropic quotient.

8.1 Vortical flow behind a F117 fighter

We simulate a subsonic flow of a notional F117 fighter at high angle-of-attack. The aircraft length is 19 meters and it has a wing span of 16 meters. The fighter geometry is shown in Figure 11 (top left). The F117 fighter is flying at cruise with Mach number 0.4 with an angle-of-attack of 20°. For such conditions, a vortical flow develops in the wake of the F117 fighter, see Figure 12.

The local Mach variable is chosen as the sensor variable for the mesh adaptation process. The space-time interpolation error on the sensor is controlled in L^2 norm. The simulation is split into 128 sub-intervals, e.g. 128 adapted mesh to run the whole simulation, and an average complexity of $200,000$ has been set. This represents a total space-time complexity of 25.6 million. Twenty solution samples are used to compute the hessian-metric $\mathbf{H}_{L^1}$ on each sub-interval. And, three fixed-point iterations have been performed to converge the adaptive process.

Practically, the discrete meshes have an average number of spatial vertices of $N_{avg} = 741,885$ vertices and total number of space-time vertices used to run the simulation is $N_{st} =$ 17.7 billion. The last adapted mesh is shown in Figure 11 where different cut planes through the volume are presented. This mesh is composed of $1,037,548$ vertices and $4,421,690$ tetrahedra. The automatic adaptation of the wake region where the vortical flow develops is clearly visible. It results in the computation of a highly resolved solution where, vortices have been propagated far in the flow field.

8.2 Impact of a water column on an obstacle

The second example aims at demonstrating the proposed method on a long-time simulation involving a 3D complex liquid-air interface. The flow modeling relies on a Level Set formulation [12] of a two-fluid incompressible flow solved by a standard Projection method. This simulation consists of a water column falling in a parallelepiped-shaped domain containing a cubic obstacle leading to unpredictable development of complex liquid-air interface. This problem involves a violent transient flow, generating a very complex interface when the water impacts the obstacle and the opposite wall (for a physical time close to 2 seconds). Then, the flow returns to a smooth sloshing mode. Several calculations of this case have been presented in the literature, see for instance [13, 19, 22]. These works show that long-term accuracy is a difficult challenge and [19] shows that mesh adaptation is efficient to achieve that goal. The simulation has been run until the final time of 6 seconds which corresponds to a forward wave motion, a backward one, and then a second forward motion.

In regards to mesh adaptation, the sensor variable is the norm of the velocity and it is coupled with a specific interface metric described in [19]. The space-time interpolation error on the sensor is controlled in L^2 norm. The simulation is split into 120 sub-intervals of 0.05 seconds, e.g. 120 adapted mesh to run the whole simulation, and an average complexity of $50,000$ has been set. This represents a total space-time complexity of 6 million. Twenty solution samples are used to compute the hessian-metric $\mathbf{H}_{L^1}$ on each sub-interval. And, three fixed-point iterations have been performed to converge the adaptive process.

The evolution of the water-air interface and the associated adapted meshes is presented in Figure 13 at several physical times. The violence of the transient flow at the impact is illustrated at time 0.8 and 1.2 seconds, and the complexity of the simulation, notably by the presence of several tubes in the flow, is illustrated by the interface shape at time 1.6 and 2 seconds. In the background, on can see the associated adapted meshes used to compute these solutions. We clearly notice the mesh refinement in the neighboring region of the interface and related to the water dynamic. We observe that the mesh size is highly dependent of the flow behavior. Meshes with a smaller number of vertices than the average are generated at

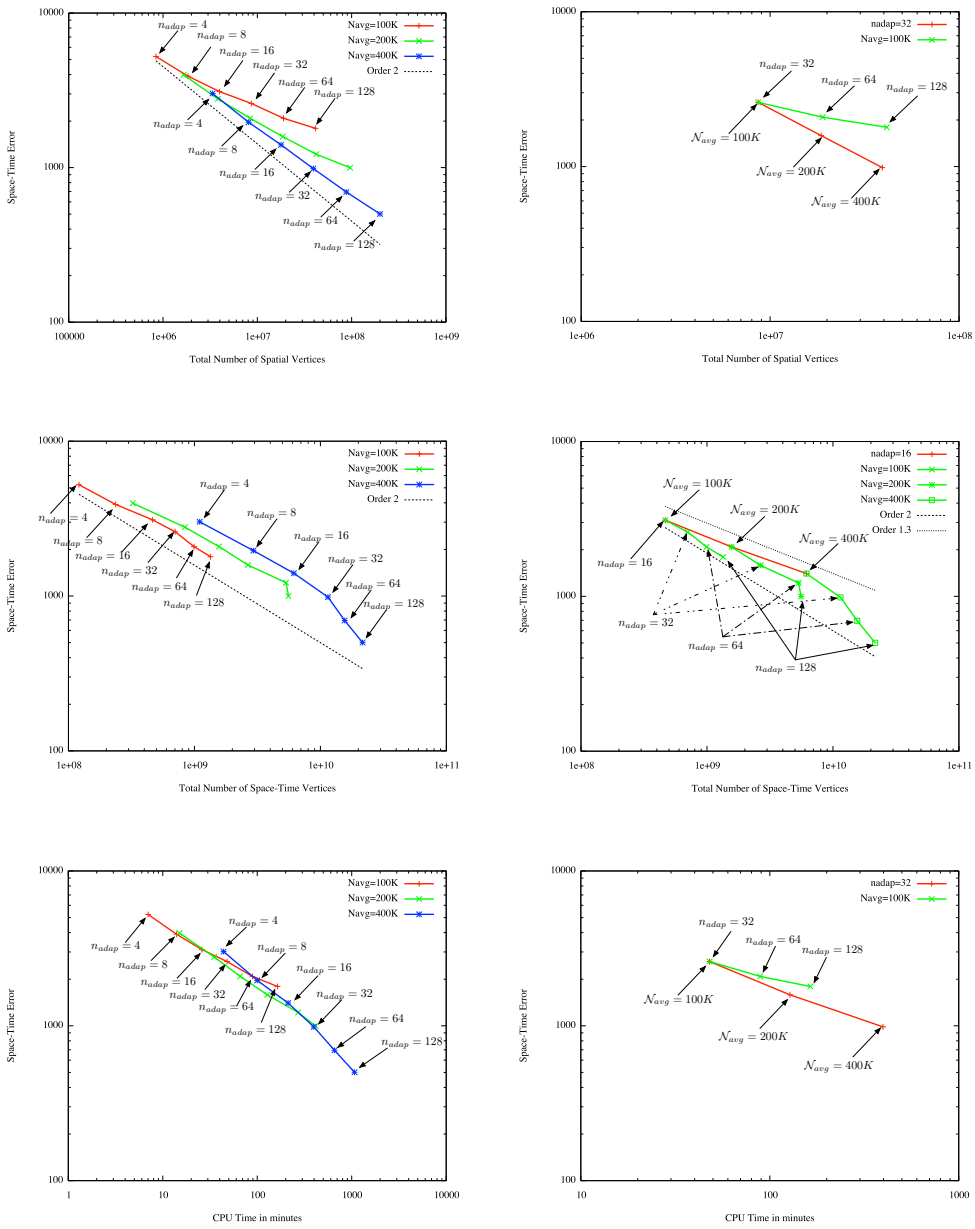

Figure 10: *3D blast in a city. Convergence curves of the space-time error with respect to the number of spatial vertices (top), the number of space-time vertices (middle), and the CPU time (bottom). Left, convergence curves for three prescribed average complexity with varying number of adapted meshes (sub-intervals). Right, convergence comparison between fixed average complexity with varying number of adapted meshes and fixed number of adapted meshes with varying average complexity.*

the beginning and at the end of the simulation when the flow is smooth, whereas larger size meshes are generated to simulate accurately the breaking wave after the impact on the wall.

9 Conclusions

In this paper, we have presented a new time-accurate multi-scale anisotropic mesh adaptation and its application to complex three-dimensional flows in computational fluid dynamics. This new approach relies on a new space-time L^p interpolation error analysis and a global fixed-point mesh adaptation algorithm. The space-time error analysis has been carried out for adaptations at each time-step or for sub-intervals, and for a fixed time-step or for explicit time-stepping. The use of the global fixed-point mesh adaptation algorithm is fundamental as it allows the adaptive process to predict the solution behavior, to compute the optimal space-time continuous mesh, to adapt the mesh in time, and to converge the non-linear mesh adaptation problem. The practical use of this theory has been thoroughly described, in particular with respect to the evaluation of the optimal space-time continuous mesh. The parallelization of the time-accurate multi-scale anisotropic mesh adaptation loop has been discussed and proven to be efficient on multi-core computers. And finally, a very detailed space-time convergence analysis has been given. This analysis emphasizes the benefits of using mesh adaptation for unsteady flows in terms of accuracy, convergence order and CPU

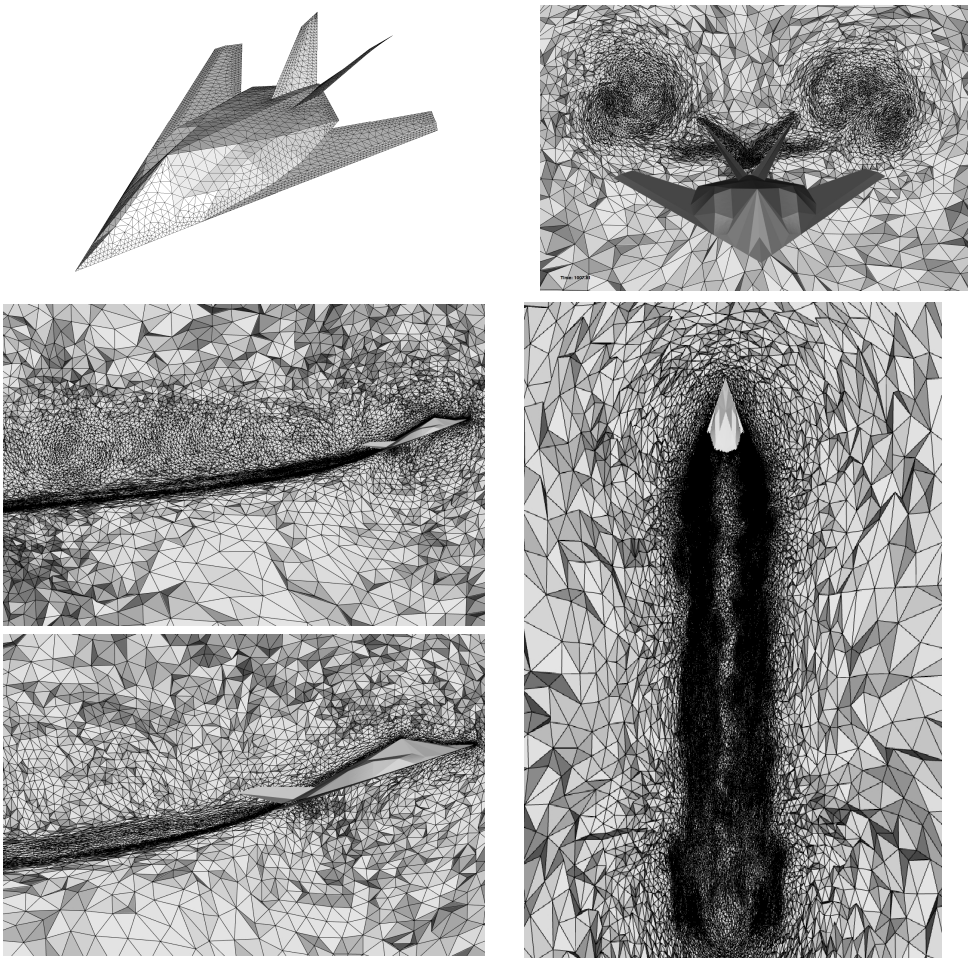

Figure 11: F117 fighter. Top left, F177 geometry and initial surface mesh. Other pictures show the last adapted meshes with cut planes through the volume: top right a back view, bottom left two side views, and bottom right a top view. Each picture illustrates the adaptation of the mesh in the wake region.

time. Many applications have been presented showing that the proposed method is generic, automatic, robust and efficient.

This strategy can be further improved. One point remaining is the adaptation of the sub-interval length that has not been addressed. It may lead to further improvement in the temporal adaptation of the space-time mesh. Moreover, in the presented approach, spatial and temporal domains are treated in a decoupled manner leading to a separate adaptation of the spatial and temporal meshes. Eventually, to envision a more optimal adaptation of the space-time mesh, without resorting to very frequent remeshing, the proposed idea is to consider dynamic meshes. An adequate prescribed motion of the mesh for each sub-interval may provide a more optimal answer.

This work has only discussed the case of an explicit temporal scheme. If an implicit scheme is considered with possibly arbitrary large time-steps, the size of the time-steps should be adapted in order to control the error introduced by the temporal scheme. A first work in this direction has been proposed in [8].

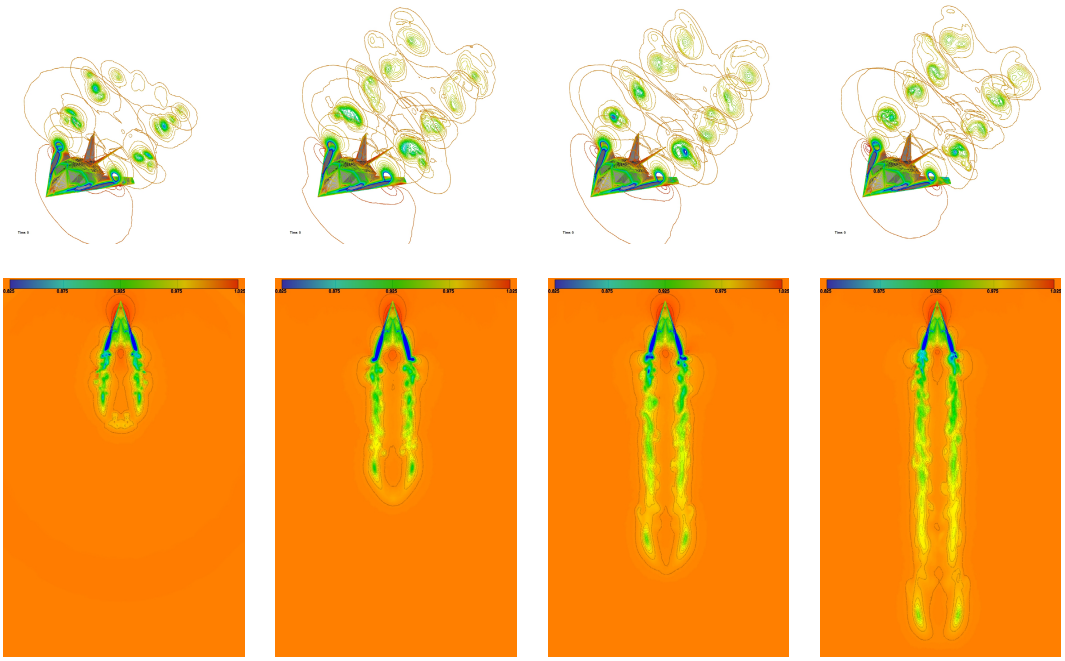

Figure 12: F117 fighter. Density solution field at a-dimensioned time $T/4$, $T/2$, $3T/4$ and T. Top, solution iso-lines for several cut planes behind the aircraft. Bottom, top view of the solution iso-values.

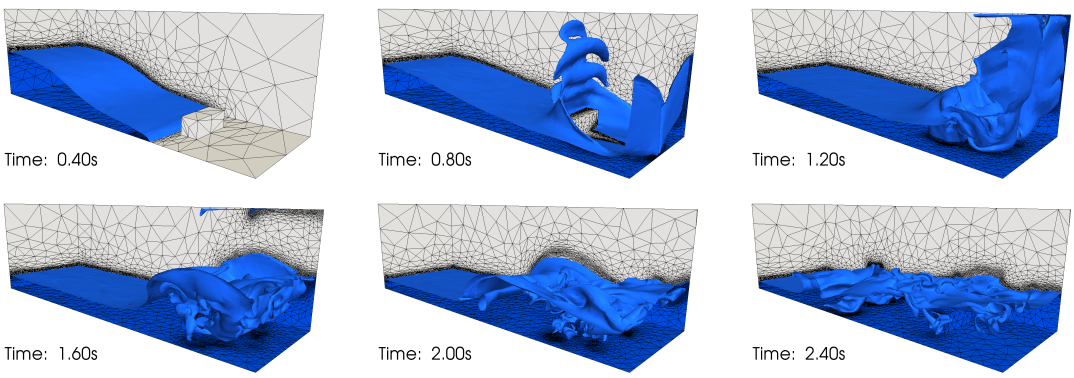

Figure 13: Water-air interface and associated adapted meshes evolution at physical time $0.4, 0.8, 1.2, 1.6, 2$ and 2.4 seconds.

Acknowledgments

This work was partly done in the MAIDESC ANR project which is supported by the french ministry of Research under contract ANR-13-MONU-0010.

We acknowledge D. Marcum and A. Dervieux for the fruitfull discussions and remarks on this work. We also acknowledge D. Guegan from Lemma Company for running the water column falling simulation.

References

[1] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elem. Anal. Des.*, 46:181–202, 2010.

[2] F. Alauzet, P.J. Frey, P.L. George, and B. Mohammadi. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *J. Comp. Phys.*, 222:592–623, 2007.

[3] F. Alauzet and A. Loseille. On the use of space filling curves for parallel anisotropic mesh adaptation. In *Proceedings of the 18th International Meshing Roundtable*, pages 337–357. Springer, 2009.

[4] F. Alauzet and A. Loseille. High order sonic boom modeling by adaptive methods. *J. Comp. Phys.*, 229:561–593, 2010.

[5] F. Alauzet and M. Mehrenberger. P1-conservative solution interpolation on unstructured triangular meshes. *Int. J. Numer. Meth. Engng*, 84(13):1552–1588, 2010.

[6] C.L. Bottasso. Anisotropic mesh adaption by metric-driven optimization. *Int. J. Numer. Meth. Engng*, 60:597–639, 2004.

[7] G. Compère, E. Marchandise, and J.-F. Remacle. Transient adaptivity applied to two-phase incompressible flows. *J. Comp. Phys.*, 227(3):1923–1942, 2008.

[8] T. Coupez, G. Jannoun, N. Nassif, H.C. Nguyen, H. Digonnet, and E. Hachem. Adaptive time-step with anisotropic meshing for incompressible flows. *J. Comp. Phys.*, 241:195–211, 2013.

[9] F. Courty, D. Leservoisier, P.L. George, and A. Dervieux. Continuous metrics and mesh adaptation. *Appl. Numer. Math.*, 56(2):117–145, 2006.

[10] P.A. de Sampaio, P.R. Lyra, K. Morgan, and N. Weatherill. Petrov-Galerkin solutions of the incompressible Navier-Stokes equations in primitive variables with adaptive remeshing. *Comput. Methods Appl. Mech. Engrg.*, 106:143–178, 1993.

[11] C. Debiez and A. Dervieux. Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations. *Comput. & Fluids*, 29:89–118, 2000.

[12] A. Dervieux and F. Thomasset. Multifluid incompressible flows by a finite element method. *Lecture Notes in Physics*, 11:158–163, 1981.

[13] R.N. Elias and A.L.G.A. Coutinho. Stabilized edge-based finite element simulation of free-surface flows. *Int. J. Numer. Meth. Fluids*, 54(6-8):965–993, 2007.

[14] P.J. Frey. About surface remeshing. In *Proceedings of the 9th International Meshing Roundtable*, pages 123–136, New Orleans, LO, USA, 2000.

[15] P.J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comput. Methods Appl. Mech. Engrg.*, 194(48-49):5068–5082, 2005.

[16] P.J. Frey and P.L. George. *Mesh generation. Application to finite elements.* ISTE Ltd and John Wiley & Sons, 2nd edition, 2008.

[17] P.L. George, F. Hecht, and M.G. Vallet. Creation of internal points in Voronoi's type method. Control and adaptation. *Adv. Eng. Software*, 13(5-6):303–312, 1991.

[18] C. Gruau and T. Coupez. 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. *Comput. Methods Appl. Mech. Engrg.*, 194(48-49):4951–4976, 2005.

[19] D. Guégan, O. Allain, A. Dervieux, and F. Alauzet. An L^∞-L^p mesh adaptive method for computing unsteady bi-fluid flows. *Int. J. Numer. Meth. Engng*, 84(11):1376–1406, 2010.

[20] W. Huang. Metric tensors for anisotropic mesh generation. *J. Comp. Phys.*, 204(2):633–665, 2005.

[21] W.T. Jones, E.J. Nielsen, and M.A. Park. Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom reduction. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2006-1150, Reno, NV, USA, Jan 2006.

[22] K.M.T. Kleefsman, G. Fekken, A.E.P. Veldman, B. Iwanowski, and B. Buchner. A Volume-of-Fluid based simulation method for wave impact problems. *J. Comput. Phys.*, 206(1):363–393, 2005.

[23] X. Li, J.-F. Remacle, N. Chevaugeon, and M.S. Shephard. Anisotropic mesh gradation control. In *Proceedings of the 13th International Meshing Roundtable*, pages 401–412. Springer, 2004.

[24] R. Löhner. Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing. *Computing Systems in Engineering*, 1(2-4):257–272, 1990.

[25] R. Löhner and J.D. Baum. Adaptive h-refinement on 3D unstructured grids for transient problems. *Int. J. Numer. Meth. Fluids*, 14(12):1407–1419, 1992.

[26] A. Loseille and F. Alauzet. Optimal 3D highly anisotropic mesh adaptation based on the continuous mesh framework. In *Proceedings of the 18th International Meshing Roundtable*, pages 575–594. Springer, 2009.

[27] A. Loseille and F. Alauzet. Continuous mesh framework. Part I: well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.

[28] A. Loseille and F. Alauzet. Continuous mesh framework. Part II: validations and applications. *SIAM J. Numer. Anal.*, 49(1):61–86, 2011.

[29] A. Loseille, A. Dervieux, P.J. Frey, and F. Alauzet. Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes. In *37th AIAA Fluid Dynamics Conference*, AIAA Paper 2007-4186, Miami, FL, USA, Jun 2007.

[30] A. Loseille and R. Löhner. Adaptive anisotropic simulations in aerodynamics. In *48th AIAA Aerospace Sciences Meeting*, AIAA Paper 2010-169, Orlando, FL, USA, Jan 2010.

[31] A. Loseille and R. Löhner. Cavity-based operators for mesh adaptation. *51th AIAA Aerospace Sciences Meeting*, Jan 2013.

[32] L. Maréchal. A parallelization framework for numerical simulation. The LP3 library. Documentation, INRIA, Jun 2010.

[33] L. Maréchal. Handling unstructured meshes in multithreaded environments with the help of Hilbert renumbering and dynamic scheduling. *Parallel Computing*, 2015. Submitted.

[34] T. Michal and J. Krakos. Anisotropic mesh adaptation through edge primitive operations. *50th AIAA Aerospace Sciences Meeting*, Jan 2012.

[35] C.C Pain, A.P. Humpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Comput. Methods Appl. Mech. Engrg.*, 190:3771–3796, 2001.

[36] R.D. Rausch, J.T. Batina, and H.T.Y. Yang. Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations. *AIAA Journal*, 30:1243–1251, 1992.

[37] J.-F. Remacle, X. Li, M.S. Shephard, and J.E. Flaherty. Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods. *Int. J. Numer. Meth. Engng*, 62:899–923, 2005.

[38] H. Sagan. *Space-Filling Curves.* Springer, New York, NY, 1994.

[39] W. Speares and M. Berzins. A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems. *Int. J. Numer. Meth. Fluids*, 25:81–104, 1997.

[40] A. Tam, D. Ait-Ali-Yahia, M.P. Robichaud, M. Moore, V. Kozel, and W.G. Habashi. Anisotropic mesh adaptation for 3D flows on structured and unstructured grids. *Comput. Methods Appl. Mech. Engrg.*, 189:1205–1230, 2000.

[41] J. Wu, J.Z. Zhu, J. Szmelter, and O.C. Zienkiewicz. Error estimation and adaptivity in Navier-Stokes incompressible flows. *Computational Mechanics*, 6:259–270, 1990.

[42] M. Yano, J.M. Modisette, and D.L. Darmofal. The importance of mesh adaptation for higher-order discretizations of aerodynamics flows. In *20th AIAA Computational Fluid Dynamics Conference*, AIAA-2011-3852, Honolulu, HI, USA, June 2011.

RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399