Metric-Based Anisotropic Mesh Adaptation for Three-Dimensional Time-Dependent Problems Involving Moving Geometries

Nicolas Barral*, Frédéric Alauzet† and Adrien Loseille ‡
Gamma3 Team, INRIA Paris-Rocquencourt, 78153, Le Chesnay, France

Anisotropic metric based mesh adaptation has proved its efficiency to reduce the CPU time of steady simulations while improving their accuracy. However its extension to time-dependent problems is far from straightforward, and the introduction of moving meshes adds new problems. This paper presents updates regarding mesh adaptation for unsteady problems with moving boundaries. A new space-time analysis of the interpolation error in the continuous mesh framework is proposed, which enables enhancements of the fixed-point unsteady mesh adaptation algorithm. The analysis is then extended to the case of moving geometries, within the range of body-fitted moving meshes and ALE simulations, and the appropriate modifications are made to the adaptation algorithm. Finally, three dimensional adaptative simulations with moving boundaries are exhibited to validate our approach.

I. Introduction

Simulating complex moving geometries evolving in unsteady flows in three dimensions, which is more and more required by industry, still remains a challenge, because it is very time consuming.

To reduce the CPU time of these simulations while preserving their accuracy, anisotropic metric-based mesh adaptation has already proved its efficiency for steady problems, and appears as a salutary perspective. However, its extension to the unsteady case is not straightforward. These simulations combine the difficulties arising from unsteadiness and geometrical complexity: global time step driven by the mesh smallest altitude, evolution of the phenomena in the whole domain, interpolation spoiling, but also three-dimensional meshing and remeshing issues with an imposed discretized surface. The introduction of moving geometries in this process even raises new difficulties, due to the handling of the mesh movement and the deterioration of its quality, the specific numerical schemes imposed by moving mesh schemes and their restrictions, as well as fluid/structure coupling and contact handling.

Three different approaches dealing with time-dependent mesh adaptation in literature can be distinguished. First,[18,20,29,32] an isotropic mesh is adapted frequently in order to maintain the solution within refined regions and introduce a safety area around critical regions. Another approach is to use an unsteady mesh adaptation algorithm[12,33] based on local or global remeshing techniques and the estimation of the error every n flow solver iterations. If the error is greater than a prescribed threshold, the mesh is re-adapted. More recently, local adaptive remeshing enabling the construction of anisotropic meshes has been considered. In this case,[27,30] the mesh is frequently adapted in order to guarantee that the solution always evolves in refined regions. All these approaches involve a large number of adaptations while introducing unquantified errors due to the transfer of the solution from the old mesh to the new one. Moreover, none of them considers the inherent non-linear nature of the mesh adaptation problem: the convergence of the mesh adaptation process is never addressed and therefore the obtaining of the optimal mesh cannot be expected.

A first answer to these issues has already been proposed.[2] It relies on the assumption that the temporal error is always controlled by the spatial error, which is indeed the case when solving a linear advection

*PhD Student, Nicolas.Barral@inria.fr, recipient of a PHD grant from the Airbus Group Foundation
†Senior Researcher, Frederic.Alauzet@inria.fr
‡Researcher, Adrien.Loseille@inria.fr

problem under a CFL condition. However, this algorithm is valid only when the space-time interpolation error is controlled in $\mathbf{L}^\infty$ norm. Since multiscale mesh adaptation has now proved its efficiency for steady CFD computations,[4,22–24] it seems relevant to extend the fixed-point algorithm proposed in Alauzet[2] to this framework.

A totally different approach has been developed, in which the mesh is moved at each time step into a mesh adapted to the solution with so called Moving-Mesh PDEs,[15] or more recently using the Monge-Ampère equation.[9,10] This approach is interesting, since it couples closely the mesh and the solution, but it seems very time-consuming for now, since the PDEs are solved at every solver time step. Moreover, the solution of the Monge-Ampère equation is known to be very difficult in three dimensions. Finally, it is not sure how these work could be extended to simulations with complex moving boundaries.

The handling of the moving geometries is crucial in that kind of simulations, both from the purely moving mesh point of view and from the solver point of view. How to keep a valid mesh all along the simulation at a lower cost ? What numerical schemes should be used to preserve the accuracy of the solution despite the specific treatments of the moving mesh ? These questions are out of the scope of this paper, but several papers [8,13,21,25,28] try to answer to them, and especially Barral[6] that details the approach followed in this paper.

Regarding adaptative strategies for moving mesh simulations, only a few attempts can be found in the literature among which the work of Löhner and Baum,[7,17,19] Saksono and al.,[31] Hassan and al.[14] and Compere and al.[11] As impressive as they can be, these results nevertheless still suffer from some of the weaknesses described above, *i.e.* mainly very frequent remeshing and spoiling interpolation stages.

A two dimension extension of the fixed point algorithm to moving-geometry problems was introduced by Olivier,[5,26] in which an effort was made to quantify the the impact of the mesh motion on the adaption process. The goal of the present paper is to update the error analysis leading to the unsteady adaptation algorithm, and to demonstrate numerically that three dimension moving mesh adaptation simulations can actually be run with this algorithm.

In this paper, we first focus on the unsteady fixed mesh case, the unsteady moving mesh case being derived from the fixed mesh case. We detail the proposed space-time multiscale error analysis. An enhanced global version of the fixed-point algorithm introduced in Alauzet[2] is presented, that is derived from this error analysis. The extension of the analysis and algorithm to moving mesh simulations is finally addressed. Numerical examples of three dimensional unsteady moving mesh simulations run with the adaptation algorithm are presented.

II. Multiscale anisotropic mesh adaptation for unsteady problems

Let us first describe the enhanced version of the unsteady fixed-point adaption algorithm for fixed meshes. This algorithm is based on a subdivision of the time interval into several sub-intervals, on which a unique mesh is computed, that is adapted to the whole sub-interval. Compared with the algorithm described in Alauzet,[2] the new algorithm is founded on a new space-time error analysis.

The new error analysis is based on the continuous mesh model[22,23] for the mesh adaptation framework. However, the multiscale mesh adaptation described in these papers controls only spatial errors. In the context of time-dependent problems, temporal error must be controlled as well. In what follows, we do not account for time discretization errors but 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.

The following assumption is made (it has been demonstrated under specific conditions[2]): *as an explicit time scheme is used for time advancing, 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.

A. The continuous mesh model

We consider the continuous mesh model.[22,23] This model draws a correspondence between the discrete computational domain and the continuous domain, allowing us to use powerful mathematical tools such as calculus of variations.

A continuous element is a $d \times d$ positive symmetric matrix $\mathcal{M}$ (*i*.e. a metric tensor). A continuous mesh

of a domain $\Omega \in \mathbb{R}^3$ is defined by a collection of continuous elements $\mathbf{M} = (\mathbb{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. The local density of the mesh is the saquare root of the determinant of the metric $d_{\mathcal{M}}(\mathbf{x}) = \sqrt{\det(\mathcal{M}(\mathbf{x}))}$. The complexity of a continuous mesh (which corresponds to the number of vertices of a discrete mesh) is the integral of the density $\mathcal{C}(\mathbf{M}) = \int_\Omega d_{\mathcal{M}}(\mathbf{x})\mathrm{d}\mathbf{x}$. Finally, a continuous linear interpolate $\pi_{\mathcal{M}}u$ can be defined as well, and the continuous interpolation error is given by $\|u - \pi_{\mathcal{M}}u\| = \mathrm{trace}(\mathcal{M}^{-\frac{1}{2}}|H_u|\mathcal{M}^{-\frac{1}{2}})$.

The model was first designed for cases where the mesh does not depend on time, but its extension to time-dependent problems is almost straightforward, as described below.

B. Space-time $\mathbf{L^p}$ error analysis

1. Error model

Our goal is to solve an unsteady PDE which is set in the 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. Let Π_h be the usual $\mathbb{P}^1$ projector. Its extension to time dependent functions reads:

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

The problem of mesh adaptation considered 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 $\mathbf{L^p}$ norm, for a given sensor u. The problem is thus stated in an *a priori* way:

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

where $\mathbf{E}$ is the interpolation error. In the continuous mesh framework (see founding papers[22,23]), this problem is rewritten in the continuous form:

$$\text{Find } \mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x},t))_{(\mathbf{x},t)\in\mathcal{Q}} \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}}u\|_{L^p(\Omega\times[0,T])}, \tag{1}$$

under the space-time constraint:

$$\mathcal{C}_{st}(\mathbf{M}) = \int_0^T \tau(t)^{-1}\left(\int_\Omega d_{\mathcal{M}}(\mathbf{x},t)\ \mathrm{d}\mathbf{x}\right)\ \mathrm{d}t = N_{st}\,. \tag{2}$$

where $\tau(t)$ is the time step used at time t of interval $[0,T]$. 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 (1-2) is solved in two steps. First, a spatial minimization is done for a fixed t. Second, a temporal minimization is performed.

2. Spatial minimization for a fixed t

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

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

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

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

Similarly to Loseille,[22] solving the optimality conditions provides the *optimal instantaneous continuous mesh in* $\mathbf{L^p}$ *norm* $\mathbf{M_{L^p}}(t) = (\mathcal{M}_{\mathbf{L^p}}(\mathbf{x},t))_{\mathbf{x}\in\Omega}$ at time t defined by:

$$\mathcal{M}_{\mathbf{L^p}}(\mathbf{x},t) = N(t)^{\frac{2}{3}} \left(\int_\Omega (\det|H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}}\right)^{-\frac{2}{3}} (\det|H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} |H_u(\mathbf{x},t)|\,. \tag{4}$$

The corresponding optimal instantaneous error at time t writes:

$$\tilde{\mathbf{E}}_{\mathbf{L^p}}(\mathbf{M_{L^p}}(t)) = 3^p\, N(t)^{-\frac{2p}{3}} \left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}} = 3^p\, N(t)^{-\frac{2p}{3}}\, \mathcal{K}(t)^{\frac{2p+3}{3}}\,. \tag{5}$$

Later in this paper, we denote: $\mathcal{K}(t) = \left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}\right)$.

3. Temporal minimization

To complete the resolution of optimization Problem (1-2), 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 $t \to N(t)$ for the instantaneous mesh size. Here, we consider the case where the time step τ is specified by the user as a function of time $t \to \tau(t)$. A similar analysis can be done to deal with the case of an explicit time advancing solver subject to Courant time step condition, but for conciseness we do not give the result here.

After the spatial optimization, the space-time error writes:

$$\mathbf{E}_{\mathbf{L^p}}(\mathbf{M_{L^p}}) = \left(\int_0^T \tilde{\mathbf{E}}_{\mathbf{L^p}}(\mathbf{M_{L^p}}(t))\,\mathrm{d}t\right)^{\frac{1}{p}} = 3\left(\int_0^T N(t)^{-\frac{2p}{3}}\,\mathcal{K}(t)^{\frac{2p+3}{3}}\,\mathrm{d}t\right)^{\frac{1}{p}} \tag{6}$$

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

$$\int_0^T \tau(t)^{-1} N(t)\,\mathrm{d}t = N_{st}. \tag{7}$$

In other words, we focus on seeking for *the optimal distribution of* $N(t)$ *when the space-time total number of nodes* N_{st} *is prescribed*. Solving this temporal optimization problem leads to the expression of the optimal space-time metric $\mathbf{M}_{L^p}$ for a prescribed time step:

$$\mathcal{M}_{\mathbf{L^p}}(\mathbf{x},t) = N_{st}^{\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\mathcal{K}(t)\mathrm{d}t\right)^{-\frac{2}{3}} \tau(t)^{\frac{2}{2p+3}} (\det|H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}} |H_u(\mathbf{x},t)|\,. \tag{8}$$

The following optimal error is finally obtained:

$$\mathbf{E}_{\mathbf{L^p}}(\mathbf{M_{L^p}}) = 3\, N_{st}^{-\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\mathcal{K}(t)\mathrm{d}t\right)^{\frac{2p+3}{3p}}. \tag{9}$$

C. Space-time $\mathbf{L^p}$ error analysis with time sub-intervals

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. Now, we want to extend the previous analysis to the fixed-point mesh adaptation algorithm context where the simulation time interval [0,T] is split into n_{adap} *sub-intervals* $[t_{i-1}, t_i]$ for $i = 1, .., n_{adap}$, see Section D. Each spatial mesh $\mathbf{M}^i$ is then kept constant during each sub-interval $[t_{i-1}, t_i]$. We could consider this partition as a *time discretization of the mesh adaptation problem.*

As previously, we get the spatial optimality condition:

$$\mathcal{M}^i_{\mathbf{L^p}}(\mathbf{x}) = N_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} \mathcal{K}^j \left(\int_{t_{j-1}}^{t_j} \tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{2p+3}}\right)^{-\frac{2}{3}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1}\mathrm{d}t\right)^{-\frac{2}{2p+3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}}\, \mathbf{H}^i_u(\mathbf{x}) \tag{10}$$

where $\mathbf{H}^i_u(\mathbf{x})$ is a mean Hessian matrix on each sub-interval, and is called Hessian-metric thereafter.

D. Global fixed-point mesh adaptation algorithm

Finally, the unsteady adaptation algorithm can be derived from this error analysis. The basic idea consists in splitting the simulation time frame $[0,\, T]$ into n_{adap} adaptation sub-intervals:

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

and to keep the same adapted 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 remeshing during the simulation, hence the number of solution transfers. This can been seen as a coarse adapted discretization of the time axis, the spatial mesh being kept constant for each sub-interval when the global space-time mesh is visualized, thus providing a first answer to the adaptation of the whole space-time mesh.

To converge the non-linear mesh adaptation problem, *i.e.,* converge the mesh-solution couple, we propose a fixed-point mesh adaptation algorithm. This is also a way to predict the solution evolution and to adapt the mesh accordingly.

Previously,[2] the optimal metric of a sub-interval could be computed directly once the simulation on the sub-interval had been run. However, the computation of the optimal continuous mesh for sub-intervals given by Relation (10) involves a global normalization term which requires the knowledge of quantities over the whole simulation time frame. This term writes:

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

which requires to know 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 continuous mesh.

To solve this issue, we suggest to consider a *global* fixed-point mesh adaptation algorithm covering the whole time frame $[0, T]$. All the solutions and Hessian-metrics are computed, and only then can the global normalization term and thus the metrics for each sub-interval be computed. This 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.

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 N_{st}

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

1. *// Adaptive loop to advance the solution in time on time frame* $[0, T]$
   For $i = 1, n_{adap}$
   (a) $\mathcal{S}_{0,i}^j$ = Interpolate conservatively next sub-interval initial sol. 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 sol. 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 $(\mathcal{C}^j, \{|H_{\max}|_i^j\}_{i=1,n_{adap}})$;
4. $\{\mathcal{H}_i^{j+1}\}_{i=1,n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}_i^j,\ \mathcal{M}_i^j\}_{i=1,n_{adap}})$;

EndFor

III. From theory to practice

Several steps of the previous algorithm are not straightforward, and need to be further detailed.

A. Computation of the Hessian-metric

The optimal $\mathbf{L^p}$ metric involves an averaged Hessian-metric $\mathbf{H}_u^i$ on sub-interval i, but it still remains to know how to compute it practically, *i.e.*, how it is discretized. The strategy adopted[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-1} and the final solution at t^i. Positive Hessian $|H_u(\mathbf{x}, t^k)|$ is evaluated for each sample.

The Hessian matrices are computed using a double least-squares procedure: first the gradients are computed using a linear least-squares reconstruction procedure, and a similar procedure is applied to recover the Hessians. Once these Hessians are computed, one need to average them. In the previous algorithm,[2] the following discretization was used:

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

where $\cap$ has to be understand as the metric intersection in time of all samples. This corresponds to an integration in time in $\mathbf{L}^\infty$ norm of the Hessians.

However, the new error analysis leads to write H^* as the integral over time of the Hessian matrices, and thus the following discretization is preferred:

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

where $\Delta t_i = t_i - t_{i-1}$ is the sub-interval time length and $t_k = t_{i-1} + \frac{k-1}{n_k - 1}\Delta t_i$. This corresponds to an integration in time in $\mathbf{L}^1$ norm of the Hessians.

Comparisons of both discretizations were performed on a large number of test cases, and it seems indeed that the second discretization (the "sum of metrics") captures more accurately the small physical phenomena, as shown in Figure 1.

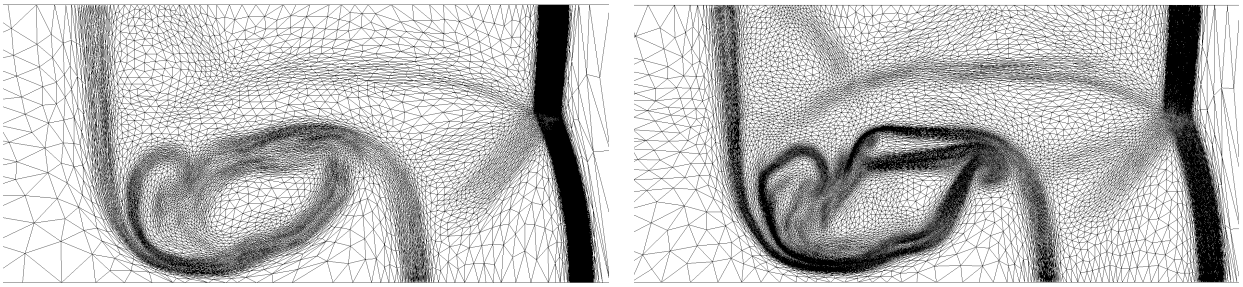

Figure 1. Zoom on the final mesh of a shock-bubble interaction problem, adapted using two time-discretizations for the Hessian-metric. Left, a $\mathbf{L}^\infty$ integration (intersection of metrics) is used, and right, a $\mathbf{L}^1$ integration (a sum of metrics).

B. Matrix-free $\mathbb{P}_1$-exact conservative solution transfer

At each remeshing step, 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, such 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: (i) mass conservation, (ii) $\mathbb{P}_1$ exactness preserving the second order of the adaptive strategy and (iii) 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 unconnected 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 as 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. Finally, the solution values at vertices are reconstructed from this piecewise linear by element discontinuous representation of the solution.

The algorithm is summarized in Algorithm 2:

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/polyhedron 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 2 points out the superiority of the $\mathbb{P}_1$-conservative solution transfer (right) compared with the classic $\mathbb{P}_1$ interpolation (left) on an adaptive blast simulation in two dimensions. For both simulations, all parameters are the same except for the solution transfer stage. This figure shows the final solution obtained with 70 mesh adaptations, *i.e.,* a total of 70 solution transfers. The diffusion and the error introduced by the classic $\mathbb{P}_1$ solution transfer clearly spoils the solution accuracy while the solution remains very accurate with the $\mathbb{P}_1$-conservative operator.

As regards CPU time overhead, it is minor in 2D but a major issue in 3D. Fortunately, the procedure is easily parallelized and scales very well, and the CPU overhead becomes admissible with the parallelization.

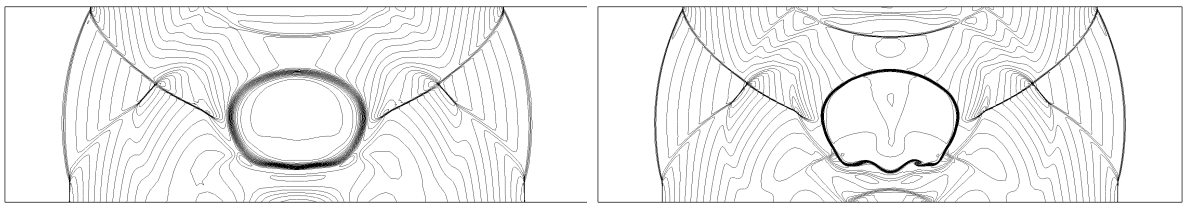

Figure 2. Final result of a blast adaptive simulation with 70 mesh adaptations. Left, using a classic $\mathbb{P}_1$ solution transfer. Right, using the $\mathbb{P}_1$-conservative solution transfer.

C. Choice of the optimal continuous mesh.

The optimal adapted mesh for each sub-interval is generated according to analysis of Section C. For the numerical results presented below, we select the optimal mesh given by Relation (10) and the following particular choice has been made:

- the Hessian-metric for sub-interval i is discretized in time in L^1 norm.
- function $\tau : t \to \tau(t)$ is constant and equal to 1
- all sub-intervals have the same time length Δt.

D. The remeshing step

[TODO AWAITING ADRIEN'S CONTRIBUTION]

E. Parallelization of the mesh adaptation loop

All the steps of the adaptation loop have been parallelized. Two different approaches are used for the two big parts of the loop. The solution computation and the solution transfer procedure are parallelized using a p-thread paradigm at the element loop level.[3] As regards the computation of the metrics, the metric gradation and the generation of the adapted meshes, a pipeline approach was used. These operations have to be done at the end of the loop, as many times as there are sub-intervals, but the operations for one sub-interval are totally independent from the operations for another sub-interval. Consequently, they can all be run in parallel: if N processors are available, N metrics and their associated meshes can be generated simultaneously in serial on one processor.

F. Example of a 3D blast test case

Let us illustrate the unsteady fixed-point algorithm. We consider a purely three-dimensional blast problem proposed by LeVeque.[16] In this simulation, shock waves are reflected on a box and interact with each other.

The gas is initially at rest in a box of dimension $[-1.5, 1.5] \times [-1.5, 1.5] \times [0, 1]$. The density and pressure equal one everywhere, except for a sphere of radius 0.2 centered in $(0, 0, 0.4)$, in which the pressure is 5. The gas is then left to evolve freely until time $t = 0.7$. Spherical shock waves emanate from the central region due to the overpressure. Wall conditions are imposed on the boundaris, so that the shock waves are relfected on them, and create complex pattern when they interact with each other.

The density of the flow is chosen as sensor variable for our mesh adaptation process. The space-time interpolation error on the solution is controlled in $\mathbf{L}^1$ norm. The time frame was split into 20 adaptation sub-intervals and 5 fixed-point iterations were used to converge the non-linear mesh adaptation problem. It is very difficult to predict the space-time complexity of a simulation, because it involves the number of vertices and the number of solver time steps. So we consider a simplified complexity, that is the sum of the numbers of vertices of the meshes of each sub-intervals, which does not depend on the time discretization of the sub-intervals. The desired accuracy was set to reach a simplified space-time complexity of $40,000,000$.

Figures 3 show the solutions and the adapted meshes at different dimensionless times. The mesh adaptation for the whole sub-interval is clearly illustrated. Indeed, the mesh refinement along band-shaped regions, which is typical of the fixed-point algorithm, is clearly visible. These band-shaped areas correspond to the evolution zone of the physical phenomena during an adaptation sub-interval. The size of the depicted meshes varies between $600,000$ and 1.3 millions of vertices.

Thanks to multiscale mesh adaptation, the complex patterns of reflected weaker shocks interacting with each other are well captured even if strong shocks are moving in the flow field. It results in an accurate solution with very few vertices compared with a uniform mesh with the smallest element size. In the moving shock region, the mesh accuracy is around $1.2\,e^{-4}$ at dimensionless time 0.07 and $9.8\,e^{-5}$ at dimensionless time 0.7 for a domain size of $[-1.5, 1.5] \times [-1.5, 1.5] \times [0, 1]$.

The CPU time to perform the whole adaptive computation (*i.e.,* the 5 loops with the 40 iterations of flow solver and solution transfer, the computation of the metric, the generation of the adapted mesh) is about 8 hours on a 20-core Intel Xeon processor at 2.5Ghz.

IV. Extension to moving mesh simulations

In this section, the above fixed-point algorithm is extended to unsteady simulations with moving geometries. We are more specifically interested in the case of geometries undergoing large displacement, for which an approach based on an elasticity-like PDE and generalized edge/face swapping is used to move the mesh keeping a good mesh quality without remeshing. Due to the boundaries displacement, the volume mesh is also (largely) distorted over time, and this distortion does not necessarily follow the physical phenomena, which would in particular break the adaptation of the mesh during the movement. Consequently, the error estimate (and thus the metric and adapted mesh) have to somehow take into account this mesh deformation. In this section, after recalling the mesh-connectivity-change moving mesh strategy that is used, the extension

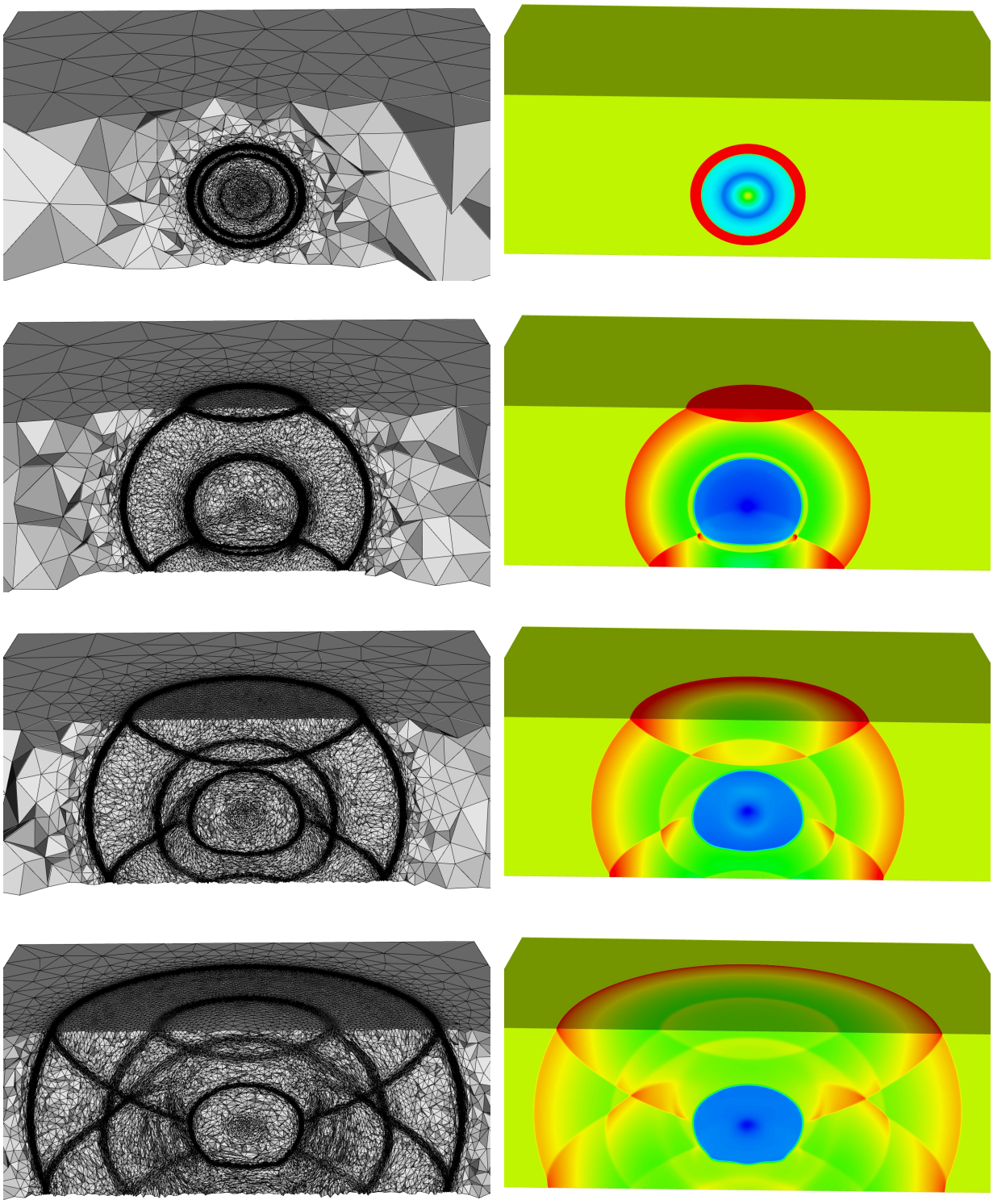

Figure 3. Adapted meshes and density solutions for the 3D blast test case, at dimensionless times 0.07, 0.315, 0.455 **and** 0.7**. Cuts into the volume mesh are made along the plane** $x = 0$**.**

of the unsteady error estimate to moving meshes will be presented, and we will explain how this new metric is plugged into the fixed-point algorithm.

A. Moving mesh strategy

The moving mesh strategy used in this paper is detailed in Alauzet.[1] The mesh adaptation makes us consider body-fitted simulations. Our strategy is designed to deal with large displacement of the boundaries, when the volume mesh quickly gets too distorted. The aim of this strategy is to preserve a good mesh quality all along the movement without costly remeshings, thanks to an improved mesh deformation step and an improved mesh optimization step.

The aim of the mesh deformation step is to compute a displacement for all inner vertices from the displacement of the boundaries, so that the volume mesh "follows" the moving geometries and thus remains valid. This is achieved solving a linear-elasticity-like PDE on the whole domain, the inside of the meshed domain being taken for a soft elastic material. To improve the efficiency of this step, the number of such solutions are significantly reduced: the mesh deformation problem is solved for large time steps, and and the trajectories are considered constant over these large time steps. To improve the precision of this step, higher order trajectories are computed: two elasticity problems are solved, thus providing a speed and an acceleration to the inner vertices.

To preserve a good mesh quality, local mesh optimizations are performed between two mesh deformation steps, using only vertex smoothing and generalized edge/face swapping. No vertices are added or removed. Vertex smoothing consists in moving vertices close to the center of gravity of their vertex ball, and helps recovering nicely shaped elements. The swap operator changes the connectivity of the mesh, and is especially powerful in handling shear and large deformation movement. This optimization step is performed not at every solver time step, but only when vertices have crossed a certain predetermined number of elements.

This algorithm was devised to fit in the Arbitrary-Lagrangian-Eulerian (ALE) framweork, for flow simulations where the mesh is moving independently from the physical phenomena. In this framework, the connectivity of the mesh is supposed to be constant, so a two dimensional ALE formulation of the swap operator was proposed.[26] This algorithm was successfully coupled to an ALE flow solver,[6] and several three dimensional imposed motion and FSI simulations were presented.

B. Optimal space-time ALE metric

We consider an time evolving sensor $u(t)$, and a mesh moved with the strategy we just described. The problem is now: how to make sure that the mesh adapted to the beginning of a sub-interval will still be adapted all along the sub-interval, as it is moved with the displacement prescribed by the elasticity solution?

Let us first consider a simplified problem. t^n and t^{n+1} are two times, Ω^n and Ω^{n+1} the spatial domain at t^n and t^{n+1} respectively, and Φ is a mapping between those two domains (we assume that this mapping exists and is a diffeomorphism):

$$\begin{array}{rccl} \boldsymbol{\phi}: & \Omega^n & \longrightarrow & \Omega^{n+1} \\ & \mathbf{x}^n & \longmapsto & \mathbf{x}^{n+1} = \boldsymbol{\phi}\left(\mathbf{x}^n\right) . \end{array} \tag{11}$$

and $\mathbf{d}$ is the corresponding mesh displacement field, such that:

$$\mathbf{x}^{n+1} = \boldsymbol{\phi}(\mathbf{x}^n) = \mathbf{x}^n + \mathbf{d}(\mathbf{x}^n) \tag{12}$$

We want to find the metric $\mathcal{M}^{n,\mathrm{ALE}}_{\mathbf{L^P}}$ from which we will generate a mesh at time t^n that, once moved with displacement $\mathbf{d}$, will be adapted to the sensor u at time t^n.

In Alauzet,[5] analyzing an edge between t^n and t^{n+1}, and considering that the optimal metric at time t^{n+1} is the multiscale $\mathbf{L^P}$ metric found previously in Section C, the following ALE metric was proposed [a] :

$$\mathcal{M}^{\mathrm{ALE}}_{\mathbf{L^P}}(\mathbf{x}^n) = D^{\mathrm{ALE}}_{\mathbf{L^P}} \left[\det\left(\widehat{H^{n+1}}(\mathbf{x}^n)\right)\right]^{-\frac{1}{2p+n}} \left(\nabla^k \boldsymbol{\phi}(\mathbf{x}^n) \cdot \widehat{H^{n+1}}(\mathbf{x}^n) \cdot \nabla^k \boldsymbol{\phi}^T(\mathbf{x}^n)\right) \tag{13}$$

with

$$D^{\mathrm{ALE}}_{\mathbf{L^P}} = \left(N^{n+1}\right)^{\frac{2}{n}} \left(\int_{\Omega^{n+1}} \left[\det\left(H^{k+1}(\mathbf{x}^{n+1})\right)\right]^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x}^{n+1} \right)^{-\frac{2}{n}} \tag{14}$$

[a] ∇^n denotes the gradient operator performed on domain Ω^n. Note that the gradient is not the Jacobian, *i.e.* for an arbitrary vector field $\mathbf{f} = (f_1, \ldots, f_n)$, its gradient matrix is $\nabla \mathbf{f} = \left(\frac{\partial f_j}{\partial x_i}\right)_{ij}$

where N^{n+1} stands for the complexity $\mathcal{C}\left(\mathcal{M}_{\mathbf{L^P}}^{k+1}\left[u^{n+1}\right]\right)$, and the $\widehat{.}$ operator transports a quantity from Ω^{n+1} to Ω^n: $\widehat{H^{n+1}}(\mathbf{x}^n) = H^{n+1}(\Phi(\mathbf{x}^n))$.

This can be rewritten in the more compact form:

$$\mathcal{M}_{\mathbf{L^P}}^{\mathrm{ALE}}(\mathbf{x}^n) = \left(\frac{N^{n+1}}{\int\limits_{\Omega^k}\left[\det H^*\right]^{\frac{p}{2p+n}}\mathrm{d}\mathbf{x}^n}\right)^{\frac{2}{n}} \left\{\det\left(H^*\right)\right\}^{-\frac{1}{2p+n}} H^* \tag{15}$$

with

$$H^* = \left[\det \nabla^k\boldsymbol{\phi}(\mathbf{x}^n)\right]^{\frac{1}{p}} \left(\nabla^k\boldsymbol{\phi}(\mathbf{x}^n)\cdot\widehat{H^{n+1}}(\mathbf{x}^n)\cdot\nabla^k\boldsymbol{\phi}^T(\mathbf{x}^n)\right) \tag{16}$$

The latter formulation is similar to the classical unsteady $\mathbf{L^P}$ metric formulation of Equation (10), and is thus the one used in practice.

C. Analytic example

Let us illustrate this result with an analytic example in two dimensions. A uniform mesh $\mathcal{H}_0^n$, a displacement field $\mathbf{d}$ between two times t^n and t^{n+1}, and a sensor u^{n+1} at the final time are given. The goal is to generate a mesh $\mathcal{H}^n$ at the initial time that, once moved into $\mathcal{H}^{n+1} = \mathbf{d}(\mathcal{H}^n)$ will be adapted to the sensor. The following functions are used:

$$u^{n+1}(x,y) = \begin{cases} 0.01\sin(50xy) & \text{if } |xy| \le \dfrac{\pi}{50} \\ \\ \sin(50xy) & \text{if } |xy| \le \dfrac{2\pi}{50} \end{cases}$$

and

$$\mathbf{d}(x,y) = \begin{bmatrix} \begin{cases} -0.3\,(x+1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x \ge 0 \\ 0.3\,(x-1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x < 0 \end{cases} \\ \\ \begin{cases} -0.3\left(x^2-1\right)(y+1)\exp\left(-5y^2\right), & \text{if } y \ge 0 \\ 0.3\left(x^2-1\right)(y-1)\exp\left(-5y^2\right), & \text{if } y < 0 \end{cases} \end{bmatrix}$$

We start from uniform mesh $\mathcal{H}_0^n$. The following procedure is applied, in L^1 norm, with a target complexity of $10,000$ vertices for the adapted mesh:

$$\begin{aligned} \nabla^n\boldsymbol{\phi} &= \texttt{ComputeTransformationGradient}(\mathcal{H}_0^n, \mathbf{d}) \\ \mathcal{H}_0^{n+1} &= \texttt{MoveMesh}(\mathcal{H}_0^n, \mathbf{d}) \\ u^{n+1} &= \texttt{ComputeTargetSensor}(\mathcal{H}_0^{n+1}) \\ \mathcal{M}_{\mathbf{L}^p}^{n,ALE} &= \texttt{ComputeALEMetric}(\mathcal{H}_0^{n+1}, u^{n+1}, \nabla^n\boldsymbol{\phi}) \\ \mathcal{H}^n &= \texttt{AdaptMesh}(\mathcal{H}^n, \mathcal{M}_{\mathbf{L}^p}^{n,ALE}) \\ \mathcal{H}^{n+1} &= \texttt{MoveMesh}(\mathcal{H}_0^n, \mathbf{d}) \end{aligned}$$

According to the above developments, we expect that the mesh $\mathcal{H}_{n+1}$, obtained by moving the vertices of $\mathcal{H}^n$ with $\mathbf{d}$, to be optimal for the control of the interpolation error of u^{n+1} in $\mathbf{L}^1$ norm.

Figure 4 presents the case: it shows the sensor function u^{n+1}, the initial uniform mesh $\mathcal{H}_0$, and $\mathcal{H}_0^{n+1}$ moved with $\mathbf{d}$. The sensor function exhibits features of different scales, small oscillations of amplitude 0.01 and big oscillations of amplitude 1, which makes it a suitable example for our multiscale mesh adaptation process.

Figure 5 shows $\mathcal{H}^n$ the mesh generated using the ALE metric at time t^n (center), $\mathcal{H}^{n+1}$ the mesh after displacement (right), and for comparison, $\mathcal{H}_{\text{target}}^{n+1}$ the mesh adapted to the sensor using a classical adaptation procedure on a fixed mesh (left), which is the target for $\mathcal{H}_{n+1}$. We can see that the moving mesh in final position is well adapted to the sensor, and is similar to the target.

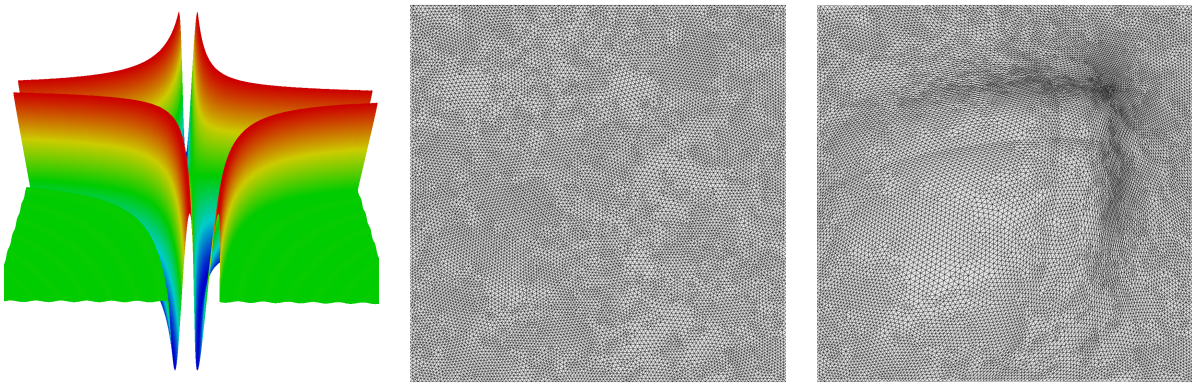

Figure 4. Sensor function used u^{n+1}, initial mesh $\mathcal{H}_0^n$, and initial mesh moved $\mathcal{H}_0^{n+1}$.

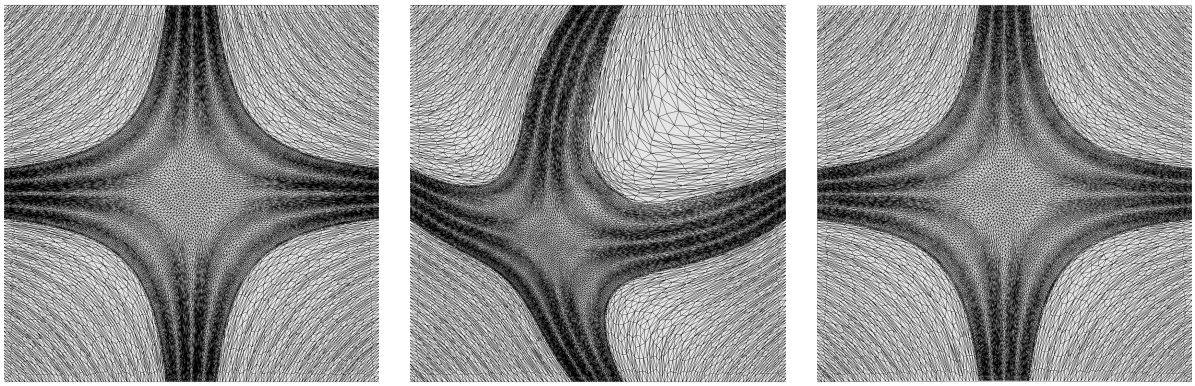

Figure 5. Mesh $\mathcal{H}_{\text{target}}^{n+1}$ adapted with a classical procedure, which is our target, mesh $\mathcal{H}^n$ adapted with the optimal ALE metric at time t^n, and final mesh $\mathcal{H}^{n+1}$ moved until time t^{n+1}.

D. The ALE fixed-point algorithm

Only few things need to be modified to extend the fixed-point algorithm to ALE simulations. The overall algorithm remains the same: the simulation time interval $[0, T]$ is still cut into n_{adap} identical sub-intervals of length δt and things work almost as described in Section D. The main difference lies in the computation of the metric in the solver. The metric is no more associated to a fixed location in space, but it is rather attached to the moving vertices. Moreover, the ALE metric must now be used: the mean Hessian-metric is computed averaging the ALE modified H^* from Equation (16). If the solution is sampled at $t^{i,k} \in [t^i, t^{i+1}]$, then the Hessian field associated with this sample is given by:

$$H^*_{i,k} = \left\{ \det\left(\nabla^i \boldsymbol{\phi}_{i,k}(\mathbf{x}^i)\right) \right\}^{\frac{1}{p}} \nabla^i \boldsymbol{\phi}_{i,k}(\mathbf{x}^i) \cdot \widehat{H_{i,k}}(\mathbf{x}^i) \cdot \nabla^i \boldsymbol{\phi}_{i,k}{}^T(\mathbf{x}^i)$$

with $\boldsymbol{\phi}_{i,k} : \Omega_h(t^i) \longrightarrow \Omega_h(t^{i,k})$ the mapping between the mesh at t^i and the one at $t^{i,k}$ and $\mathbf{x}^i = \mathbf{x}(t^i)$. The Hessian fields $\left(H^*_{i,k}\right)_{1 \leq k \leq n_k}$ are then averaged like in the fixed-mesh case. A mesh adapted for this sub-interval is then generated as described previously. This way, the mesh generated at t^i will be adapted at each $t^{i,k}$. The mesh is then moved using the algorithm described in section A according to the prescribed geometry movement. To perform mesh optimizations without losing the anisotropy of the adapted mesh, the metric of the previous fixed-point iteration is used to compute qualities. The other steps of the algorithm (the computation of the global normalization term, the computation of the metrics from the Hessian-metrics, the generation of the adapted meshes and the interpolation steps) remain unchanged.

V. Numerical examples

We now present several CFD simulations to illustrate the moving-mesh unsteady fixed-point adaptation algorithm, that demonstrate the efficiency of this algorithm. All the examples are in three dimensions.

A. The flow solver: Wolf

The following examples were run using our in-house flow solver, `Wolf`.[6] It solves the compressible Euler equations in the ALE framework, required by the movement of the mesh. A vertex-centered Finite Volume scheme is used, with an HLLC approximate Riemann solver to compute the numerical fluxes. A high-order scheme is derived according to a MUSCL type method combined with a generalization of the Superbee limiter with three entries. As regards the temporal accuracy, the considered SSPRK schemes are based on the strict application of the Discrete Geometrical Conservation Law (DGCL).

B. An shock tube in expansion

The first example is a variation of the classical Sod schock-tube problem, with a tube being homogeneously expanded in one direction. This *a priori* simple test case shows that the refined regions follow the physical phenomena of interest, and that the anisotropy is well preserved despite the mesh being moved.

The tube initial dimensions are $[0,1]\times[-0.15,015]\times[0,1]$, and the gas is split in two regions: for $x \leq 0.5$ the state is $(\rho_{left}, \mathbf{u}_{left}, p_{left}) = (1, \mathbf{0}, 1)$ whereas for $x > 0.5$ the state is $(\rho_{right}, \mathbf{u}_{right}, p_{right}) = (0.125, \mathbf{0}, 1)$. The gas is then left to evolve freely, and the classical rarefaction wave, contact discontinuity and shock discontinuity appear and evolve in the tube. The tube is expanded to the right with the following movement imposed to the mesh vertices:

$$\begin{cases} x(t) &= x_0 + 1.5x_0 \cdot t \\ y(t) &= y_0 \\ z(t) &= z_0 \end{cases}$$

The simulation is run until time $t = 0.66$ so that the size of the tube doubles. Special transmitting conditions are prescribed in $x = 0$, so that the rarefaction wave can go out of the tube. For that case, 20 sub-intervals were prescribed, and 4 fixed-point iterations were performed. The density field is used as sensor for the adaptation. A simplified space-time complexity of $400,000$ was prescribed, and the meshes have an average size of $21,000$ vertices. [TODO]

The meshes at different time steps and the corresponding solution are shown in Figure 6. Not only do the waves evolve in the refined bands, but those bands also move as the tube is expanded. The adapted regions are transported together with the mesh in accordance with our algorithm, that associates metrics to a moving vertex rather than to a fixed position in space. In Figure 8, we zoom on the solution at the beginning and the end of the eighth sub-interval to stress the evolution of the physical phenomena inside an adaptation band during a sub-interval. In Figure 7, we show the mesh at the beginning and at the end of the last sub-interval, to emphasize the movement of the adapted regions together with the movement of the tube boundaries, still preserving the anisotropy.

C. A moving ball in a shock tube 3D

In the second example, we take the same shock tube as previously, but we add a moving ball inside the tube, that interacts with the shock the contact discontinuity.

More precisely, the dimensions of the tube are: $[0,1] \times [-0.15, 0.15] \times [-0.1, 0.1]$. At initial time, the gas is split in two states: for $x \leq 0.5$ the state is $(\rho_{left}, u_{left}, v_{left}, w_{left}, p_{left}) = (1, 0, 0, 0, 1)$ whereas for $x > 0.5$ the state is $(\rho_{right}, u_{right}, v_{right}, w_{right}, p_{right}) = (0.125, 0, 0, 1)$. A ball of radius $r = 0.02$ is immersed in the gas, its center being in position $(0.75, 0, 0)$. The tube is fixed, while the ball has a constant speed $\mathbf{v}_{ball} = -0.3\ \mathbf{e_x}$. The simulation is run until final time $t_f = 0.25$. After a while, the ball goes through the shock and the contact discontinuity, creating complex patterns both in front of it and in its wake. We expect all these physical phenomena to be captured correctly by the multiscale adaptation algorithm. For that case, 40 sub-intervals were prescribed, and 5 fixed-point iterations were run, and the density field is used as sensor for the adaptation. A simplified space-time complexity of $3,000,000$ was prescribed, and the meshes have an average size of $70,000$ vertices with a size range going from $50,000$ to $130,000$ vertices.

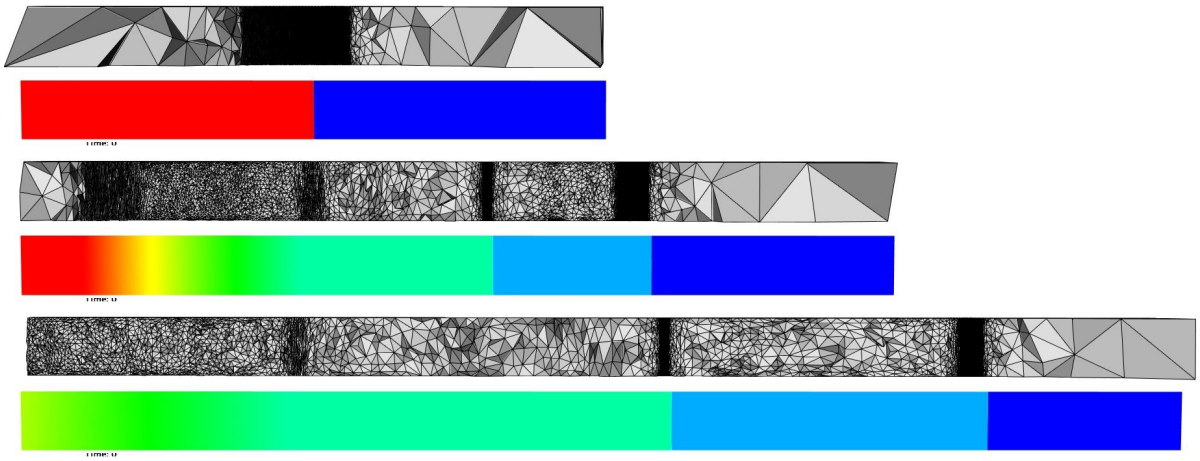

Figure 6. Adapted meshes and density solutions for the two expading shock tube case at time 0, 0.5 and 0.66. Cuts into the volume mesh are made along the plane $y = 0$.

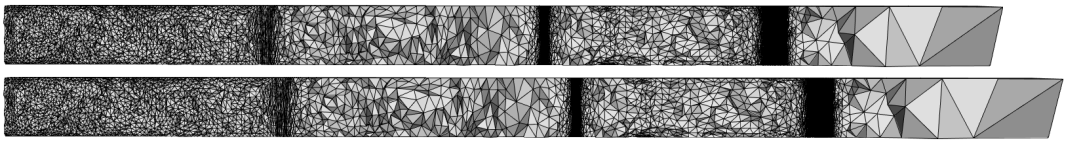

Figure 7. Mesh at the beginning (top) and end of the eighth sub-interval. One can see the bands are move forward as the tube is being expanded.

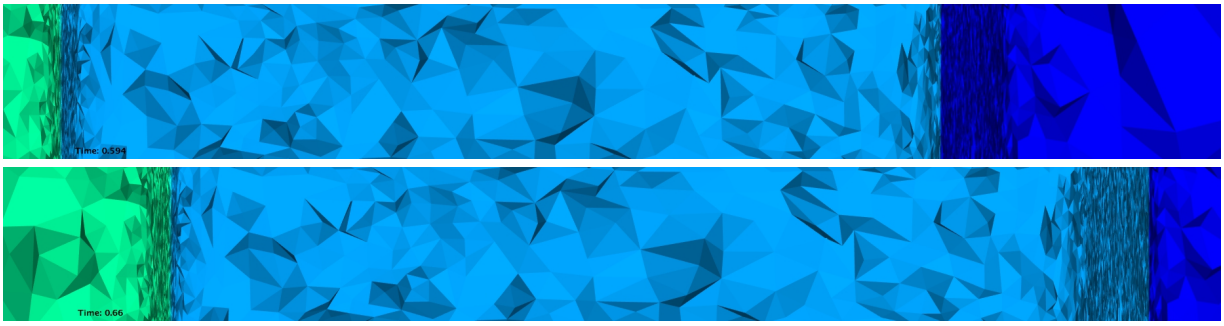

Figure 8. Zoom on the mesh and solution at the beginning (top) and at the end (bottom) of the last sub-interval. Once can see the two waves move forward in the adapted bands.

The meshes at different time steps and the corresponding solution are shown in Figure 9. Since we are interested in the interaction of the ball with the shock and contact discontinuity, we only show the part $x > 0.5$ of the tube. The mesh shown are the meshes at the end of the sub-intervals, *i.e.* the meshes that have been moved. The adaptation allows us to capture the two waves with a good accuracy all along the simulation, but also to recover some more complex features around the ball. It is interesting to note that the shock is perfectly meshed as soon as the ball has gone through it. A closer look at the surface of the ball also shows that it is adapted with highly anisotropic elements, and the anisotropy close to the surface is well preserved.

For this simulations, allowing mesh optimization (smoothing and edge/face swapping) is mandatory, otherwise, the shearing created by the advance of the ball in the volume mesh quickly creates invalid elements. Consequently, it is crucial to compute the quality criteria that trigger optimization with regards to the actual metric of the mesh. In this case, it is clear on the figures that the swaps didn't break the anisotropy in the

anisotropic adapted regions.

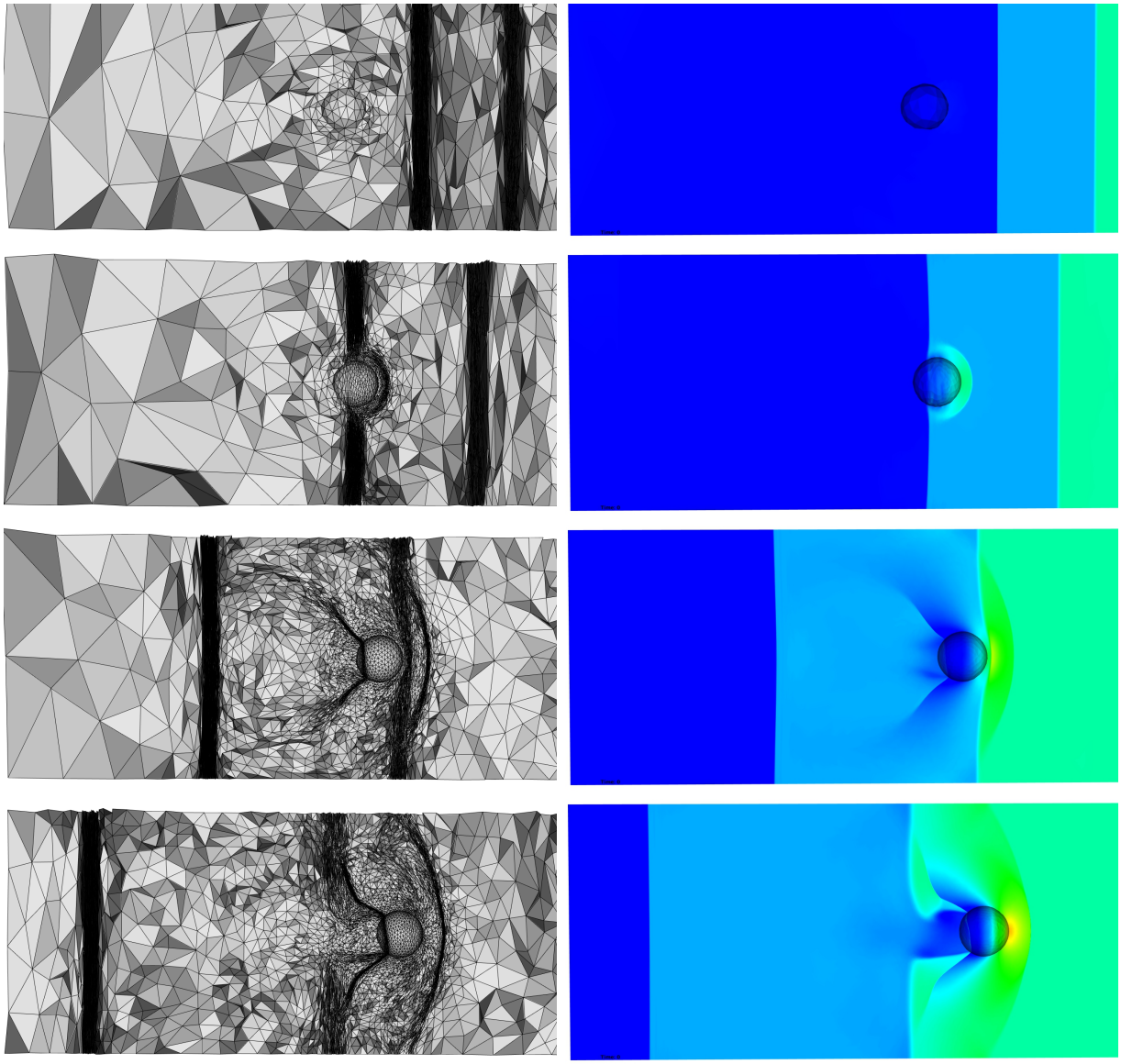

Figure 9. Adapted meshes and density solutions for the ball in a shock tube test case, at times 0.09375, 0.125, 0.19375 **and** 0.25**. Cuts into the volume mesh are made along the plane** $y = 0$**.**

D. Two F117 aircraft flight paths crossing

This case is an example of moving mesh simulation, described in Barral.[6] It models two F117 aircrafts having crossing flight paths, translating and rotating. This problem is difficult in terms of mesh movement and it illustrates the efficiency of the topology-change moving mesh algorithm in handling large displacements of complex geometries without any remeshing. When both aircrafts cross each other, the mesh deformation encounters a large shearing due to the opposite flight directions. The topology-change mesh deformation algorithm handles easily this complex displacement thanks to the mesh local reconnection.

As concerns the fluid simulation, the aircrafts are moved at a speed of Mach 0.4, in an initially inert uniform fluid: at $t = 0$ the speed of the air is null everywhere. Transmitting boundary conditions are used on the sides of the surrounding box, while slipping conditions are imposed on the two F117 bodies. After a short phase of initialization, the flow is established when the two F117s pass each other. Accoustic waves are created in front of the F117s, and the density fields around the aircrafts and in their wake interact.

The adaptation parameters are the following: the adaptation is performed on the density field, the time interval is divided into 50 sub-intervals, 5 fixed-point iterations are performed, and a simplified space-time complexity of $20,000,000$ is targeted. The meshes have between $170,000$ and $420,000$ vertices, for an average mesh size of $380,000$ vertices. The smallest altitude generated is $5.5\,e^{-4}$ at the beginning of the tenth sub-interval, for a domain of dimension $[-1800, 2400] \times [-800, 1000] \times [-1200, 1200]$

The meshes and density solutions at different time steps are shown in Figure 11. The meshes are well adapted to the solution. Thanks to the adaptation, the trails of the F117s are captured far from the aircrafts, and their interactions with the waves is clearly visible. The anistropy of the meshes is clearly visible, in front of the aircrafts (corresponding to the acoustic waves created by their setting in motion at $t = 0$), and in their wake, even if the anisotropic ratios are not so large, due to limitations in terms of computing power and time. On Figure 10, a close up on the two aircrafts is made, that shows the adapted surface mesh and solution. In this case again, a correct handling of the mesh optimization is necessary to preserve the anisotropy.

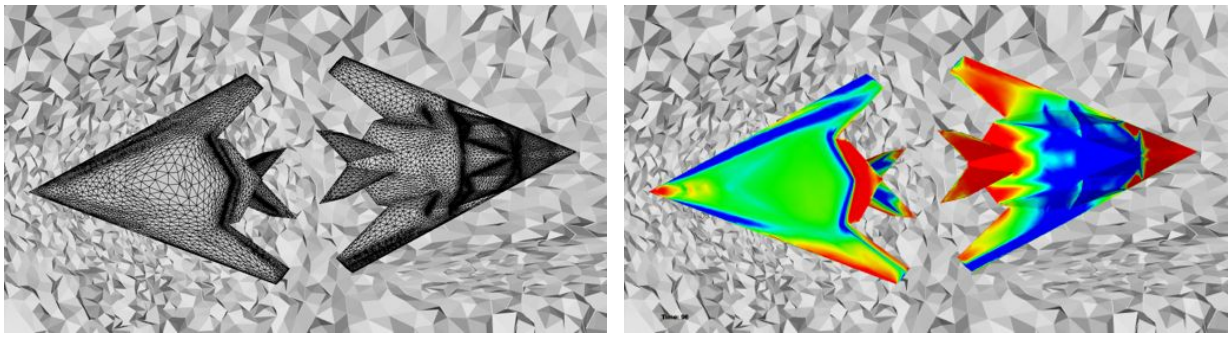

Figure 10. Close-ups on the surface mesh and solution of the two F117s.

VI. Conclusion

In this paper, a new space-time analysis of the error for time-dependent problems has been established within the continuous mesh framework. It completes previous analysis, and allows us to correct the fixed-point multiscale adaption algorithm for unsteady simulations. A new version of this algorithm has been presented, where the normalization of the metrics and the computation of a mean Hessian-metric has been updated, and details of its practical implementation have been given. The case of moving meshes, within the range of body-fitted ALE simulations has then been addressed. The analysis leading to the optimal ALE metric has been presented. This metric allows us to generate meshes that will remain adapted once moved as required by the geometry movements. The integration of this metric to the mesh adaptation is then direct if the appropriate form of the optimal ALE metric is used. Finally, several numerical examples were shown in three dimensions, that validate the proposed approach: the adapted bands, corresponding to the adaptation sub-intervals, are correctly moved in the same direction as the phenomena at stake, even if the moving mesh algorithm moves vertices differently.

There is still a lot of room for improvement. In particular a quantitative analysis of the results of the moving mesh adaptation algorithm has to be carried out. Some preliminary work studied the difference between the ALE metric and the classical one in terms of mesh quality.[5] However, the impact of the ALE correction of the metric has to be evaluated within the whole adaptation process, as well as the accuracy of the solution, and to a lesser extent the loss of accuracy due to the moving mesh. It is necessary to make sure that no weaker phenomenon is left aside. At a longer term, an effort on the time discretization of the adaptation algorithm also has to be made, to adapt the size of the sub-intervals, and control the sizes of the elements during the movement. Finally, work on the moving mesh strategy will also be necessary in order to run more complex simulations. Fluid-structure simulations have been run with this strategy,[6] and we aim at running adaptative versions of these simulations soon.

References

[1]F. Alauzet. A changing topology moving mesh technique for large displacements. *Engineering with Computers*, submitted.

[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 prediction by utilizing mesh adaptive methods. In *48th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2010-1390, Orlando, FL, USA, Jan 2010.

[5]F. Alauzet and G. Olivier. Extension of metric-based anisotropic mesh adaptation to time-dependent problems involving moving geometries. In *49th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2011-0896, Orlando, FL, USA*, page 143, 2011.

[6]N. Barral and F. Alauzet. Large displacement body-fitted FSI simulations using a mesh-connectivity-change moving mesh strategy. In *44th AIAA Fluid Dynamics Conference, AIAA-2014, Atlanta, GA, USA*, June 2014.

[7]J.D. Baum and R. Löhner. Numerical Simulation of a Shock Interaction with a Modern Battlefield Tank. In *AIAA-91-1666*, 1991.

[8]J.D. Baum, H. Luo, and R. Löhner. A new ALE adaptive unstructured methodology for the simulation of moving bodies. In *32th AIAA Aerospace Sciences Meeting*, AIAA Paper 1994-0414, Reno, NV, USA, Jan 1994.

[9]P.A. Browne, C.J. Budd, C. Piccolo, and M. Cullen. Fast three dimensional r-adaptive mesh redistribution. *Journal of Computational Physics*, 275(0):174 – 196, 2014.

[10]L. Chacón, G.L. Delzanno, and J.M. Finn. Robust, multidimensional mesh-motion based on monge–kantorovich equidistribution. *Journal of Computational Physics*, 230(1):87 – 103, 2011.

[11]G. Compere, J-F. Remacle, J. Jansson, and J. Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Engng*, 82(7):843–867, 2010.

[12]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.

[13]O. Hassan, K.A. Sørensen, K. Morgan, and N. P. Weatherill. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. *Int. J. Numer. Meth. Fluids*, 53(8):1243–1266, 2007.

[14]O. Hassan, K.A. Sorensen, K. Morgan, and N.P. Weatherill. A Method for Time Accurate Turbulent Compressible Fluid Flow Simulation with Moving Boundary Components Employing Local Remeshing. *Int. J. Numer. Meth. Fluids*, 53:1243–1266, 2007.

[15]Weizhang Huang and Robert D. Russell. Adaptive mesh movement — the {MMPDE} approach and its applications. *Journal of Computational and Applied Mathematics*, 128(1–2):383 – 398, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[16]J.O. Langseth and R. J. LeVeque. A wave propagation method for three-dimensional hyperbolic conservation laws. *J. Comp. Phys.*, 165:126–166, 2000.

[17]R. Löhner. Adaptive Remeshing for Transient Problems. *Int. J. Numer. Meth. Engng*, 75(1–3):195–214, 1989.

[18]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.

[19]R. Löhner and J.D. Baum. Numerical Simulation of a Shock Interaction with Complex Geometry Three-Dimensional Structures using a New Adaptive H-Refinement Scheme on Unstructured Grids. In *AIAA-90-0700*, 1990.

[20]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.

[21]R. Löhner, J.D. Baum, E. Mestreau, D. Sharov, C. Charman, and D. Pelessone. Adaptive embedded unstructured grid methods. *Int. J. Numer. Meth. Engng*, 60:641–660, 2004.

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

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

[24]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 and Exhibit*, AIAA-2007-4186, Miami, FL, USA, Jun 2007.

[25]S. Murman, M. Aftosmis, and M. Berger. Simulation of 6-DOF motion with cartesian method. In *41th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2003-1246, Reno, NV, USA, Jan 2003.

[26]G. Olivier. *Anisotropic metric-based mesh adaptation for unsteady CFD simulations involving moving geometries*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2011.

[27]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.

[28]C.S. Peskin. Flow patterns around heart valves: a numerical method. *J. Comp. Phys.*, 10:252–271, 1972.

[29]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.

[30]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.

[31]P.H. Saksono, W.G. Dettmer, and D. Perić. An Adaptive Remeshing Strategy for Flows with Moving Boundaries and Fluid-Structure Interaction. *Int. J. Numer. Meth. Engng*, 71(9):1009–1050, 2007.

[32]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.

[33]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.

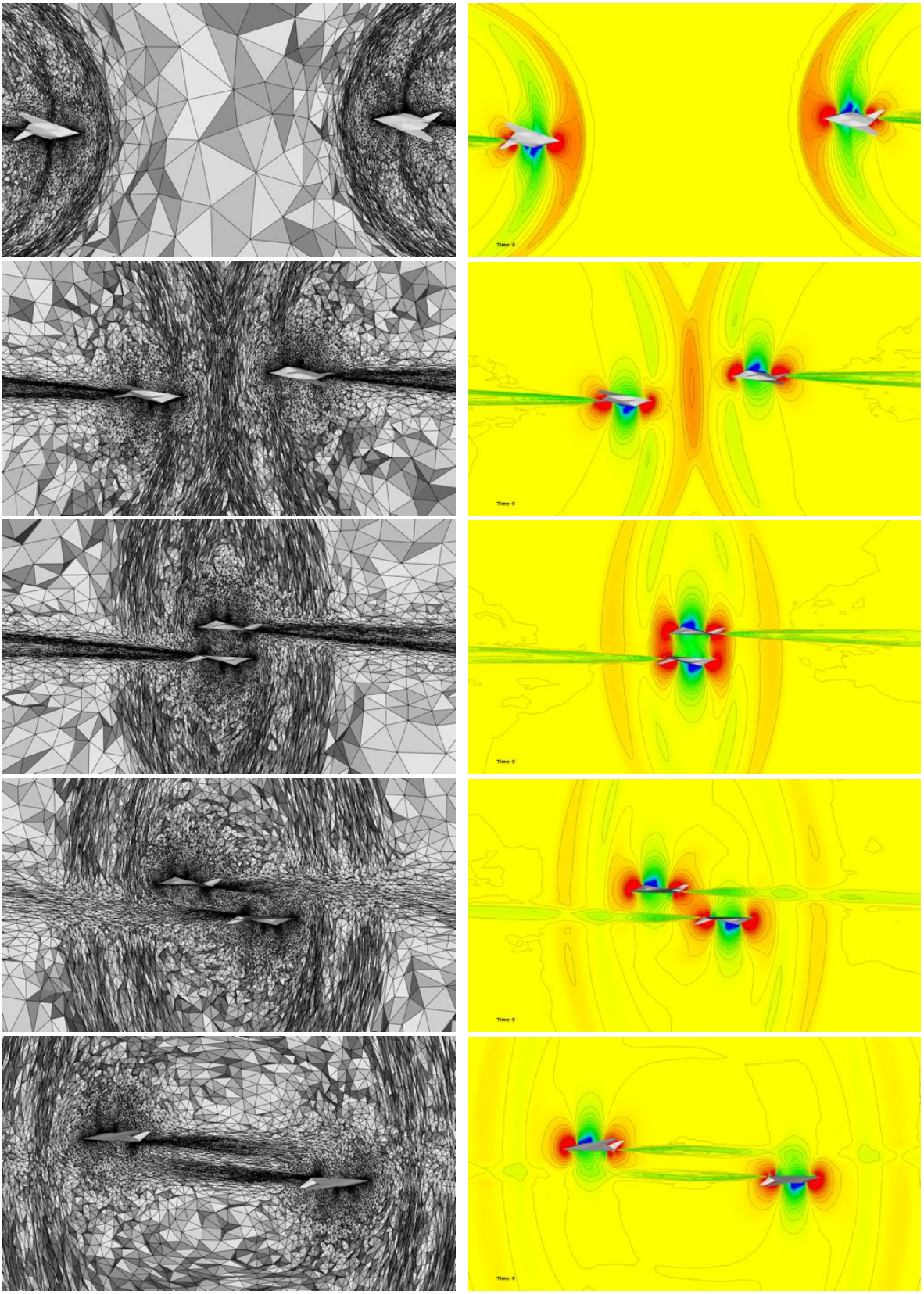

Figure 11. Adapted meshes and density solutions for the two F117s test case at different time steps. Cuts into the volume mesh are made along the plane $y = 0$.