ESAIM: PROCEEDINGS AND SURVEYS, March 2015, Vol. 50, p. 169-188
Franck BOYER, Thierry GALLOUET, Raphaèle HERBIN and Florence HUBERT Editors

3D PARALLEL ANISOTROPIC UNSTEADY MESH ADAPTATION FOR THE HIGH-FIDELITY PREDICTION OF BUBBLE MOTION

Victorien Menier[1]

Abstract. Anisotropic unsteady mesh adaptation is applied to the high-fidelity prediction of bubble motion, which has applications in the framework of safety evaluations for nuclear reactors. A prescribed advection of the bubble is performed, which lacks any physical sense but is representative of the reality and makes it possible to precisely measure the diffusion caused by the numerical model. The model is described, and results are presented in 2D and 3D, with comparisons in terms of mesh convergence, CPU time, and propagation of the interface.

Résumé. L'adaptation de maillage instationnaire anisotropique est appliquée à la prédiction haute-fidélité de la propagation de l'interface d'une bulle, dont les applications existent notamment dans le cadre des évaluations de sureté des réacteurs nucléaires. Une advection de la bulle est prescrite, qui n'est pas physique mais représente bien la réalité tout en permettant une mesure précise de la diffusion causée par le modéle numérique. Ce dernier est décrit et des résultats sont présentés en 2D et 3D avec comparaisons des convergences en maillage, des temps CPU et de la propagation de l'interface de la bulle.

Introduction

In this paper, the high-fidelity prediction of the propagation of an extremely thin interface is addressed from the meshing point of view. Applications exist in the framework of safety evaluations for nuclear reactors, in which gas bubbles may appear in the liquid phase. In this context, the meshing strategy used must deal with the discontinuities of most variables through the bubble's interface. Here, the contribution of anisotropic unsteady mesh adaptation to this issue is discussed, which aims at increasing the accuracy of the solution while decreasing the CPU time of the simulation, by dividing the physical time frame considered into sub-intervals for each of which an anisotropic mesh is generated according to size and directional constraints.

In order to measure how well the numerical model predicts bubble motion, the Kothe-Rider test [1] is performed. An initial sphere is linearly advected, governed by a velocity field $\vec{v}(x, y, z, t)$ of period T, and starting from $t = 0$. Due to the periodicity of the velocity field, the bubble is expected to recover its original position (i.e. the sphere) at each $t \in \frac{T}{2}\mathbb{N}$. Although this bubble advection lacks any physical sense, it is representative of the reality, and makes it possible to precisely estimate the numerical error due to the meshing strategy at each $t \in \frac{T}{2}\mathbb{N}$.

[1] Inria Paris-Rocquencourt, 78153 Le Chesnay, France.

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201550009

Several studies of numerical methods for propagating an extremely thin interface have been carried out. Adaptive Mesh Refinement (AMR) techniques have been used [2, 3] as well as level set methods coupled with anisotropic mesh adaptation [4, 5].

1. Numerical Model

This Section describes the numerical model used for performing the Kothe-Rider test, including the advection solver, as well as the steady and unsteady mesh adaptation algorithms.

Using anisotropic mesh adaptation for predicting bubble motion is motivated by the features of this physical phenomena, which (i) is concentrated in a small area of the computational domain, (ii) is anisotropic, and (iii) is time-dependent. Therefore, uniform meshes - i.e. meshes whose edges size is constant in the domain - are not optimal in terms of both sizes and directions. Mesh adaptation, however, provides a way to control the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. For instance, unstructured Hessian-based mesh adaptation has already proved its efficiency to improve the solution accuracy while decreasing the problem complexity (i.e. the number of degrees of freedom) [6–9].

1.1. **Advection Solver**

In this Section, the advection solver used for performing the Kothe-Rider test case is described. It is based on `Wolf` [10], our in-house flow solver developped in the Gamma3 team at Inria. The advection equation is the following:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0, \tag{1}$$

where ρ is the density (see Figure 1) and $\vec{v}(x, y, z, t)$ a velocity field.

The spatial discretization of Eq. 1 is based on a vertex-centered finite volume formulation on unstructured meshes. Let $\mathcal{H} = (K_i)$ be a mesh of a domain Ω, the vertex-centered finite volume formulation consists in associating to each vertex P_i of the mesh a control volume or finite volume cell, denoted C_i. The discretized domain Ω_h can be written as the union of the mesh elements or the union of the finite volume cells:

$$\Omega_h = \bigcup_{i=1}^{N_T} K_i = \bigcup_{i=1}^{N_S} C_i \,.$$

The dual finite volume cells used for the bubble motion are the classical median cells, as depicted in Figure 2.

Second order space accuracy is achieved through a piecewise linear interpolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure with a particular edge-based formulation with upwind elements. The advection flux through an edge $\overrightarrow{P_iP_j} = \vec{e}_{ij}$ (see Figure 2) is given by:

$$\Phi_{ij}^{\texttt{adv}} = \begin{cases} v_{moy}.\rho_i.\,\|\vec{e}_{ij}\| & \text{if} \quad v_{moy} > 0 \\ v_{moy}.\rho_j.\,\|\vec{e}_{ij}\| & \text{else.} \end{cases} \tag{2}$$

where ρ_i is the solution at the vertex P_i,

$$v_{moy} = \frac{1}{2}(v_i + v_j),$$

$$\begin{cases} v_i &= \vec{v}(P_i, t).\vec{n} \\ v_j &= \vec{v}(P_j, t).\vec{n} \end{cases},$$

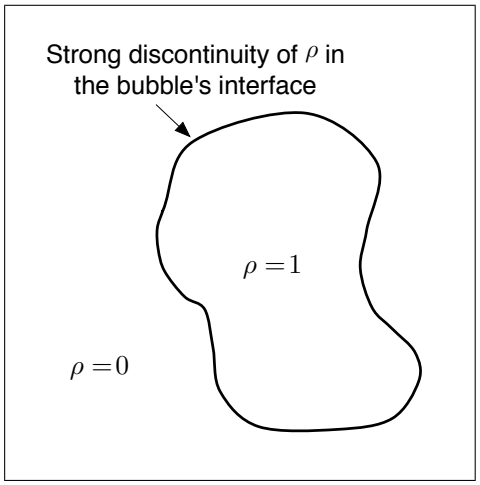


FIGURE 1. The density ρ is discontinued through the bubble's interface.

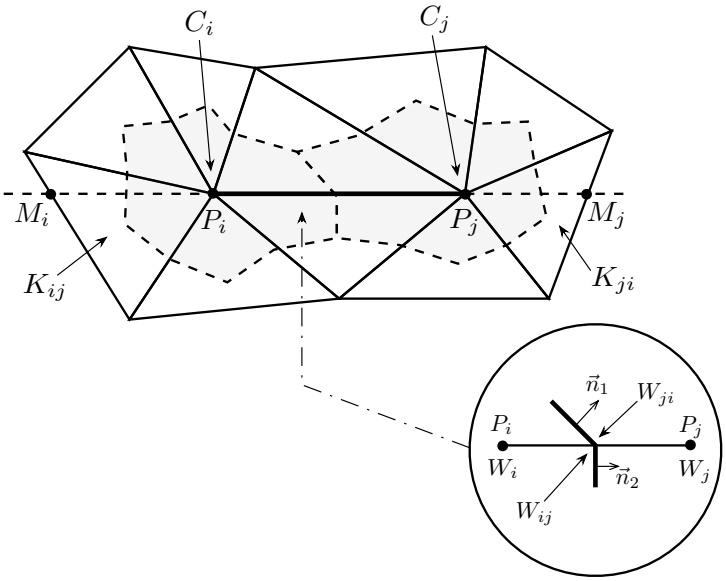


FIGURE 2. Illustration of two finite volume control cells C_i and C_j around two vertices P_i and P_j.

and $\vec{n}$ is the edge's normal vector.

The MUSCL type reconstruction method is used in order to increase the order of accuracy of the scheme [11]. The idea is to use extrapolated values ρ_{ij} and ρ_{ji} of ρ at the interface ∂C_{ij} to evaluate the flux. The following approximation is performed:

$$\Phi_{ij} = \Phi_{ij}(\rho_{ij}, \rho_{ji}, \mathbf{n}_{ij}) \,,$$

ρ_{ij} and ρ_{ji} which are linearly interpolated as:

$$\rho_{ij} \;=\; \rho_i + \frac{1}{2}\,(\nabla\rho)_{ij} \cdot \overrightarrow{P_iP_j} \qquad \text{and} \qquad \rho_{ji} \;=\; \rho_j + \frac{1}{2}\,(\nabla\rho)_{ji} \cdot \overrightarrow{P_jP_i} \,,$$

where, in contrast to the original MUSCL approach, the approximate "slopes" $(\nabla\rho)_{ij}$ and $(\nabla\rho)_{ji}$ are defined for any edge and obtained using a combination of centered, upwind and nodal gradients.

The centered gradient related to an edge P_iP_j, is defined as:

$$(\nabla\rho)^C_{ij} \cdot \overrightarrow{P_iP_j} = \rho_j - \rho_i \,.$$

Upwind and downwind gradients are computed according to the definition of upstream and downstream tetrahedra of an edge P_iP_j. These tetrahedra are respectively denoted K_{ij} and K_{ji}. K_{ij} (resp. K_{ji}) is the unique tetrahedron of the ball of P_i (resp. P_j) the opposite face of which is crossed by the line defined by the edge P_iP_j. Upwind and downwind gradients are then defined for vertices P_i and P_j as:

$$(\nabla\rho)^U_{ij} = (\nabla\rho)|_{K_{ij}} \quad \text{and} \quad (\nabla\rho)^D_{ij} = (\nabla\rho)|_{K_{ji}} \,.$$

where $(\nabla\rho)|_K = \sum_{P\in K} \rho_P \nabla\phi_P|_K$ is the P_1-Galerkin gradient on tetrahedron K. Parametrized nodal gradients are built using the β-scheme:

$$\begin{aligned}(\nabla\rho)_{ij} &= (1-\beta)(\nabla\rho)^C_{ij} + \beta\,(\nabla\rho)^U_{ij}\\ (\nabla\rho)_{ji} &= (1-\beta)(\nabla\rho)^C_{ij} + \beta\,(\nabla\rho)^D_{ij}\,,\end{aligned}$$

where $\beta \in [0,1]$ is a parameter controlling the amount of upwinding. For instance, the scheme is centered for $\beta = 0$ and fully upwind for $\beta = 1$.

Bubble motion is predicted using a V4-scheme, obtained for $\beta = 1/3$. It can be demonstrated that this scheme is third-order for the two-dimensional linear advection on structured triangular meshes. On unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained. High-order gradients are given by:

$$\begin{aligned}(\nabla\rho)^{V4}_{ij} &= \frac{2}{3}(\nabla\rho)^C_{ij} + \frac{1}{3}(\nabla\rho)^U_{ij}\\ (\nabla\rho)^{V4}_{ji} &= \frac{2}{3}(\nabla\rho)^C_{ji} + \frac{1}{3}(\nabla\rho)^D_{ij}\,.\end{aligned}$$

The parallelization of the solver is based on posix standard threads (pthreads) taking advantage of multi-core chips and shared memory architectures supported by most platforms. Loops running over tables and structures featuring direct or indirect memory accesses take up a large part of the total CPU time when dealing with meshes, and are easily parallelized with pthreads [12].

1.2. **Steady Mesh Adaptation**

The general idea of mesh adaptation is to modify the discretization of the computational domain according to size and directional constraints, in order to minimize a given error criterion, and thus improve the adequation with the underlying physics. Mesh adaptation has proved its efficiency in improving the tradeoff between computational time and accuracy of the solution. Figure 3 presents the example of mesh adaptation for the recovery of a bubble's interface in the steady case. The adaptation is performed on the density variable ($\rho = 1$ inside the bubble and 0 outside), so that the mesh is refined close to the interface and coarsened elsewhere. As the solution varies dramatically in the normal direction to the interface and does not vary in the tangential direction, stretched elements aligned to the direction of anisotropy are created.

Isotropic mesh adaptation simply relies on the prescription of a scalar size field. Anisotropic mesh adaptation, however, must control the sizes along prescribed directions. To this end, we use the unit-mesh concept [13] in the continuous mesh framework [14]. The main idea is to generate a uniform mesh with respect to a Riemannian metric space rather than to the Euclidian space. According to the continuous mesh framework, any mesh can be represented by a continuous Riemannian metric field $\mathcal{M}$. The link between a continuous and a discrete mesh is based on the concept of unit-mesh: a mesh is unit according to $\mathcal{M}$, if all its edges have a length $l_{\mathcal{M}}(e)$ in the metric approximately equal to 1 and if its elements K have a volume $|K|_{\mathcal{M}}$ in the metric approximately equal to $\sqrt{2}/12$. More formally, a metric tensor $\mathcal{M}$ in $\mathbb{R}^n$ is a $n \times n$ symmetric definite positive matrix. The scalar product of two vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ according to $\mathcal{M}$ is defined as:

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

So, the associated norm of a vector in $\mathbb{R}^n$ is defined as:

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

which measures the length of the vector $\vec{u}$ in the metric $\mathcal{M}$. Thus, the length of an edge $e = AB$ according to $\mathcal{M}$ is defined:

$$l_{\mathcal{M}}(e) = \int_0^1 \sqrt{{}^t AB \mathcal{M}((1-t)A + tB)AB}. \tag{3}$$

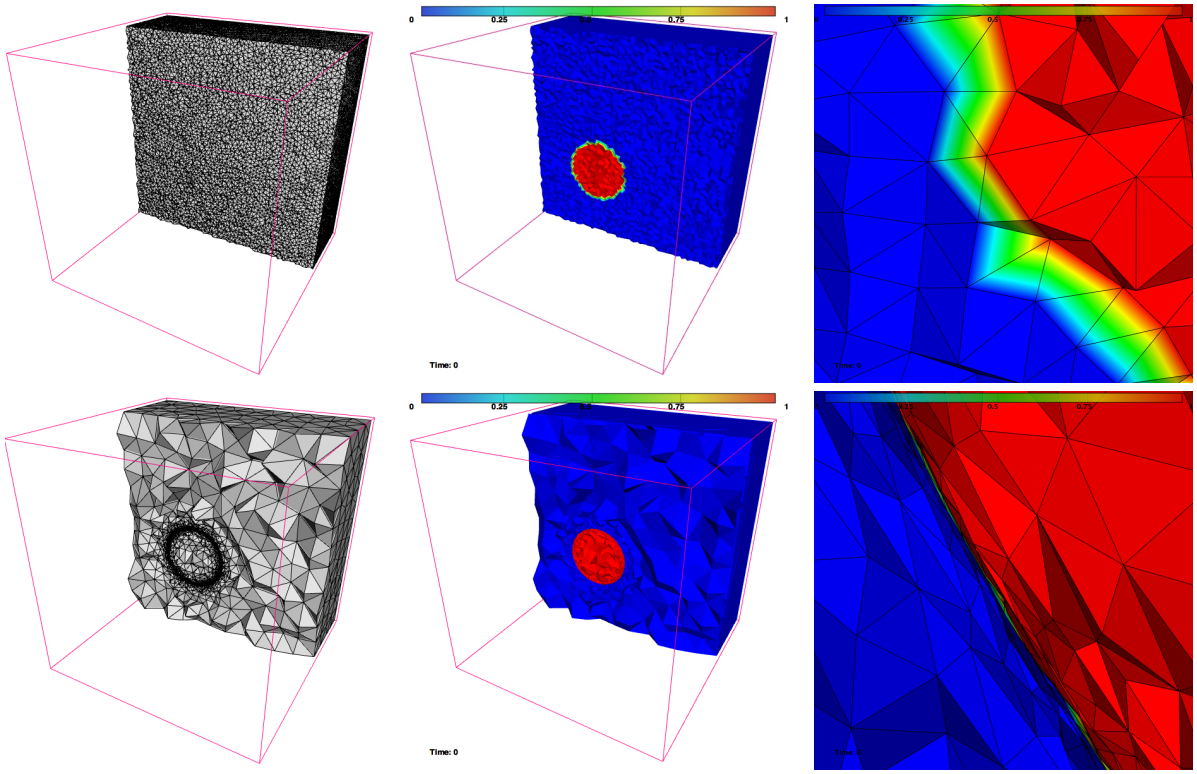


FIGURE 3. Steady mesh adaptation at time $t = 0$. Top: cuts in the initial uniform mesh. Bottom: cuts in the final adapted mesh which is refined at the bubble's interface.

Mesh adaptation consists in generating a mesh that is unit according to a Riemanninan metric field obtained from an error estimation of the solution. We now describe the mesh adaptation process in the steady case. We start from an initial (coarse and non-adapted) mesh $\mathcal{H}_0$. The four main stages of the process are the following: (i) solution computation on $\mathcal{H}_0$ ($\mathcal{S}_0^0$), (ii) metric computation ($\mathcal{M}_0$), (iii) mesh generation ($\mathcal{H}_1$) using the sizes and directions provided by $\mathcal{M}_0$, (iv) solution interpolation to $\mathcal{H}_1$.

As presented in Figure 4, these four stages are repeated several times and at each iteration, the complexity of the generated meshes is increased. It makes it possible to converge both the mesh and the solution to an optimal state and to capture accurately physical phenomena. The classical steady mesh adaptation scheme is a fixed point algorithm: the algorithm stops when there is no variation of the couple mesh/solution from one iteration to the next. More details on each stage are now provided.

(i) Solution computation. We use the advection solver described in Section 1.1 to compute the density variable ρ at each vertex of the mesh.

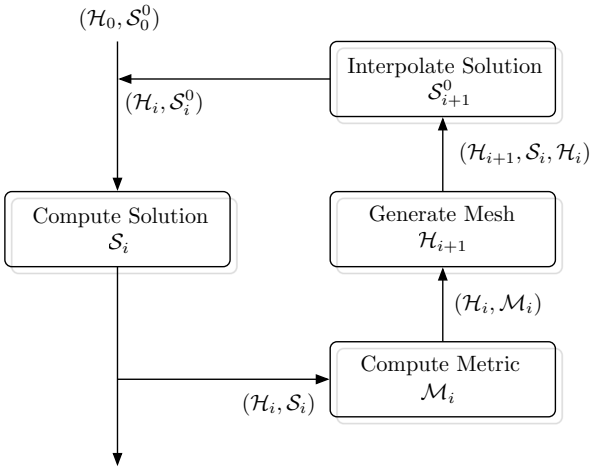


FIGURE 4. The mesh adaptation loop

(ii) Metric computation. We build the metric $\mathcal{M}$ according to an error estimate. In the case of bubble motion, we chose to minimize the $\mathbb{P}^1$ interpolation error $\rho - \Pi_h\rho$, which can be expressed in terms of second derivatives of ρ, i.e., the Hessian H_ρ of ρ, and of the metric $\mathcal{M}$:

$$|\rho - \Pi_h\rho| \equiv |\rho - \pi_{\mathcal{M}}\rho| = \mathtt{trace}(\mathcal{M}^{-\frac{1}{2}}\,|H_\rho|\,\mathcal{M}^{-\frac{1}{2}})\,, \tag{4}$$

where $|H_\rho|$ is derived from H_ρ by taking the absolute value of the eigenvalues and $\pi_{\mathcal{M}}\rho$ is the continuous interpolate defined in [14] and [15]. Minimizing $\|\rho - \Pi_h\rho\|_{L^p(\Omega)}$ for a given number N of vertices can be recast in the continuous setting as minimizing $\|\rho - \pi_{\mathcal{M}}\rho\|_{L^p(\Omega)}$ for a complexity $C(\mathcal{M}) = N$, which is the continuous counter part of the number of vertices. This optimization problem is solved to compute the optimal metric $\mathcal{M}_{L^1}^{opt}$:

$$\mathcal{M}_{L^1}^{opt}(\rho) = \arg\min_{C(\mathcal{M})=N} \mathtt{trace}(\mathcal{M}^{-\frac{1}{2}}\,|H_\rho|\,\mathcal{M}^{-\frac{1}{2}})\,. \tag{5}$$

The expression of the optimal continuous mesh is:

$$\mathcal{M}_{L^1}^{opt}(\rho) = \frac{N}{\int_\Omega (\mathtt{det}|H_\rho|)^{1/3}} (\mathtt{det}|H_\rho|)^{-1/3}|H_\rho|\,. \tag{6}$$

(iii) Mesh generation. Generating an anisotropic unit mesh $\mathcal{H}$ with respect to $\mathcal{M}$ requires to use any anisotropic mesh generator [16–21]. The results presented in this paper were achieved using our in-house remesher [22]. The general idea is to perform iteratively simple mesh modifications such as vertex insertions/removal, edge swaps/collapses etc., in order to generate unit mesh elements. All the aforementioned operations are performed using a single mesh operator based on cavity remeshing [22].

(iv) Solution interpolation. The interpolation operator used [23] verifies the properties of mass conservation, $\mathbb{P}^1$-exactness (order 2) and maximum principle, which are achieved through local mesh intersections and quadrature formulae.

1.3. **Unsteady Mesh Adaptation**

The random progression of the bubble's interface in the computational domain is a time-dependent problem, which makes the steady mesh adaptation algorithm inadequate. Indeed, the mesh generated for the time $t = 0$

would lead to a strong error in both time and space for the rest of the time frame. A non-optimal approach would be to perform a steady mesh adaptation at each time step, but this would be too costly in terms of CPU. In order to minimize the number of mesh adaptations, an unsteady mesh adaptation algorithm is used [24], which consists in dividing the physical time frame considered into sub-intervals and generating an adapted mesh for each one of them.

The unsteady mesh adaptation scheme is derived from the classical steady mesh adaptation algorithm. It consists of two steps: the main (classical) adaptation loop, and an internal loop in which a transient fixed point problem is solved (see Figure 5). Let us consider the simulation of bubble motion from time $t = 0$ to $t = T$. First, the time period $[0, T]$ is divided in N sub-intervals $[t, t + \Delta t]$. At each iteration of the main adaptation loop is considered a time period $[t, t+\Delta t]$ in which the solution evolves. For instance during the i-th ($i \in [1, N]$) main iteration, a mesh $\mathcal{H}_i$ is generated that is suitable for times $t \in [(i-1)\Delta t, i\Delta t]$. This sub-interval mesh is generated via the internal loop: at each internal iteration j, a metric $\mathcal{M}_{(i,j)}$ is computed that takes into account the solution progression in the sub-interval and a mesh $\mathcal{H}_{(i,j+1)}$ is generated according to $\mathcal{M}_{(i,j)}$. The final solution $\mathcal{S}_{(i,j+1)}$ of the period (i.e. at time $t = i\Delta t$) is computed and compared to the solution of the previous internal iteration $\mathcal{S}_{(i,j)}$ in order to assess the convergence of the internal loop. Let ϵ be a given parameter, the internal transient fixed point algorithm is iterated until:

$$\frac{\left\|\mathcal{S}_{(i,j+1)} - \mathcal{S}_{(i,j)}\right\|_{L^1(\Omega)}}{\left\|\mathcal{S}_{(i,j+1)}\right\|_{L^1(\Omega)}} \leq \epsilon \quad ,$$

where Ω is the computational domain.

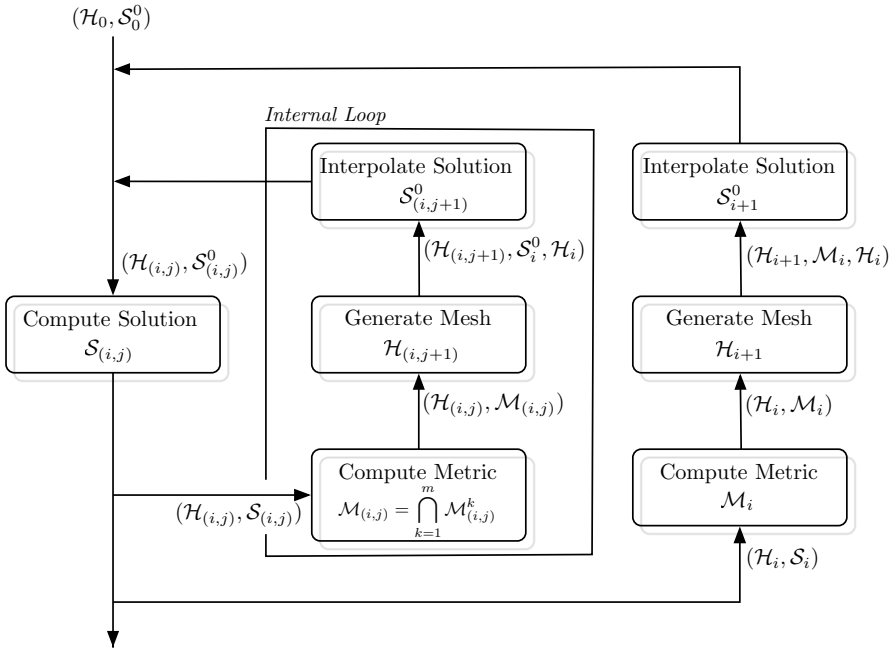


FIGURE 5. The mesh adaptation loop

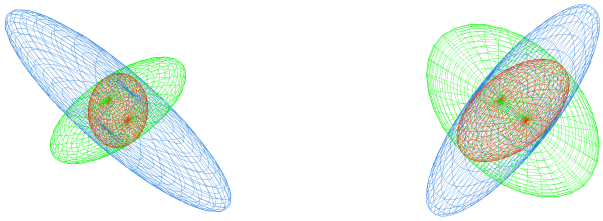


FIGURE 6. Two illustrations of the metric intersection procedure. $\mathcal{M}_1$ (in blue) is intersected with $\mathcal{M}_2$ (in blue). The resulting metric $\mathcal{M}_1 \cap \mathcal{M}_2$ is in red.

Computing $\mathcal{M}_{(i,j)}$. At the jth internal iteration of the main iteration i, a metric intersection in time procedure is used to compute $\mathcal{M}_{(i,j)}$, the metric field that takes into account the evolution of the solution in the ith sub-interval $[t, t+\Delta t]$ (i.e. $[(i-1)\Delta t, i\Delta t]$). All the intermediate solutions between $(i-1)\Delta t$ and $i\Delta t$ must be considered to mesh suitably all this region so as to control the error of the solution throughout the time sub-interval. So, $\mathcal{M}_{(i,j)}$ is the intersection of m intermediate metrics:

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

where $\cap$ is the metric intersection defined above and $\mathcal{M}^k_{(i,j)}$ is the kth intermediate metric of the sub-interval $[(i-1)\Delta t, i\Delta t]$. The number of intermediate metrics m is given as an input of the algorithm.

Definition of metric intersection $\cap$. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two metrics of eigenvalues (λ_i) and (μ_i) resp. $(i = 1, 3)$. Let $\mathcal{P} = (e_1, e_2, e_3)$ be the matrix whose columns are formed by the eigenvectors of $\mathcal{N} = \mathcal{M}_1^{-1}\mathcal{M}_2$. The intersection of two metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ is given by:

$$\mathcal{M}_1 \cap \mathcal{M}_2 = {}^t\, \mathcal{M}_1^{1/2} \overline{\mathcal{M}_{1\cap 2}} \mathcal{M}_1^{1/2}$$

where

$$\overline{\mathcal{M}_{1\cap 2}} = \mathcal{P} \begin{pmatrix} \max(\lambda_1, 1) & 0 & 0 \\ 0 & \max(\lambda_2, 1) & 0 \\ 0 & 0 & \max(\lambda_3, 1) \end{pmatrix} {}^t\mathcal{P}$$

Geometrically speaking, a metric intersection is depicted in Figure 6.

2. Numerical Results

The Kothe-Rider test [1] is performed in order to measure the impact of the meshing strategy in accurately predicting bubble motion. In 3D, an initial sphere is linearly advected in a cubic computational domain according to a periodic velocity field of period T. Due to this periodicity, the bubble moves forward from $t = 0$ to $t = \frac{T}{4}$, and backward from $t = \frac{T}{4}$ to $t = \frac{T}{2}$. So, it is expected to recover its original position (i.e. the sphere) at each $t \in \frac{T}{2}\mathbb{N}$ (see Figure 7). Although the Kothe-Rider test case lacks any physical sense, it is representative of the reality and makes it possible to measure the diffusion due to the numerical model by estimating the spatial error at each $t \in \frac{T}{2}\mathbb{N}$ (see Figure 7).

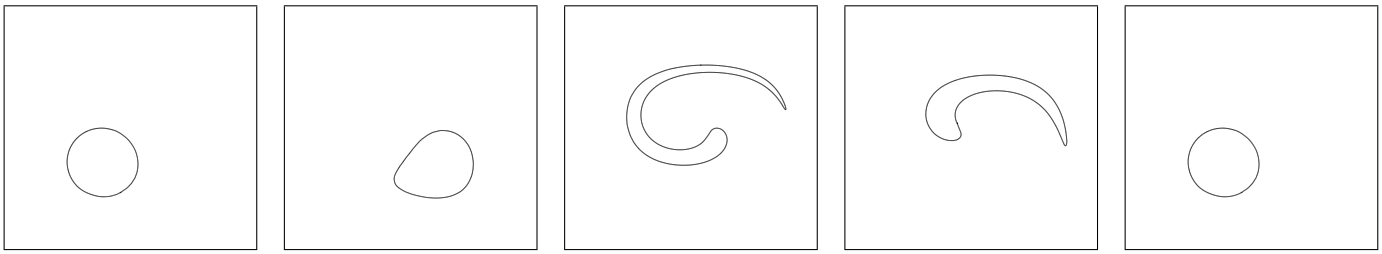


FIGURE 7. The 2D bubble's interface at several times from $t = 0$ to $t = \frac{T}{2}$ (from left to right: $t = 0, \frac{T}{16}, \frac{T}{4}, \frac{3T}{8}, \frac{T}{2}$). A spatial error is computed at each $t \in \frac{T}{2}\mathbb{N}$ to evaluate the diffusion due to the numerical model.

2.1. **Results in 2D**

In 2D, the computational domain is $\Omega = [0,1] \times [0,1]$ and the velocity field is the following:

$$\begin{cases} u(x,y,t) &= -\sin^2(\pi x)\sin(2\pi y)\cos(2\pi\frac{t}{T}) \\ v(x,y,t) &= \sin(2\pi x)\sin^2(\pi y)\cos(2\pi\frac{t}{T}) \end{cases} \tag{7}$$

The initial bubble's radius is $R = 0.15$ and it is centered in $(x_0, y_0) = (0.50, 0.75)$. The velocity field period is $T = 6$ (see Eq. 7). The bubble was chosen to be advected from time $t = 0$ to $t = 10\frac{T}{2}$ and a spatial error is computed at each $t \in \frac{T}{2}\mathbb{N}$:

$$\epsilon = \sum_i |C_i| \, |\rho_{i,\texttt{exact}} - \rho_{i,h}|$$

where $|C_i|$ is the area of the finite volume cell associated to vertex P_i, $\rho_{i,\texttt{exact}}$ is the exact solution at vertex P_i and $\rho_{i,h}$ is the computed solution.

A total of 9 simulations were run: 5 using uniform meshes and 4 unsteady mesh adaptations. These 9 simulations are summarized in Table 2.1. The final error at time $t = 10\frac{T}{2}$ was computed for each simulation, see Figure 11. It shows that a 2nd order mesh convergence is achieved for adapted meshes (1st order for uniform meshes). The total CPU time of each simulation presented in Figure 12 shows that the final spatial error observed in 6 hours using a uniform mesh of 1 Million vertices can be achieved in 20 minutes using mesh adaptation. Moreover, it would take 12 days and 22 hours for an uniform mesh to reach the final spatial error observed after the mesh adaptation which took 2 hours and 30 minutes.

All the 2D simulations were run on a two 2.93 GHz Quad-Core Intel Xeon chips with 24Gb of RAM. The time frame $[0, 10\frac{T}{2}]$ was divided into 20 sub-intervals. At each main iteration in the unsteady adaptation loop, the 20 mesh generations were performed in parallel (8 cores, one process per core at the time).

Adapted meshes alongside with the corresponding solutions are depicted in Figure 9 and close-up views of the meshes in Figure 10. Mappings of the density for several meshes and at several physical times are depicted in Figure 13. A comparison of the final bubble's interface at time $t = 10\frac{T}{2}$ is given in Figure 8.

2.2. **Results in 3D**

In 3D, the bubble is advected in a cubic computational domain $\Omega = [0,1] \times [0,1] \times [0,1]$. The advection is governed by the following time-periodic velocity field:

# ver	type	#procs	Total CPU	Spatial L^1 error at $t = 10\frac{T}{2}$
10k	uniform	4	57s	8.50e-02
50k	uniform	8	6m5s	4.74e-02
100k	uniform	8	13m58s	3.68e-02
500k	uniform	8	2h11m	1.99e-02
1M	uniform	8	6h0m	1.50e-02
24k	adapted	8	6m39s	2.73e-02
37k	adapted	8	16m35s	1.57e-02
50k	adapted	8	50m42s	1.05e-02
92k	adapted	8	2h30m	4.89e-03

TABLE 1. Summary of the 9 simulations for the Kothe-Rider test in 2D. For the mesh adaptations, the given number of vertices corresponds to the mesh for time $t = 10\frac{T}{2}$.

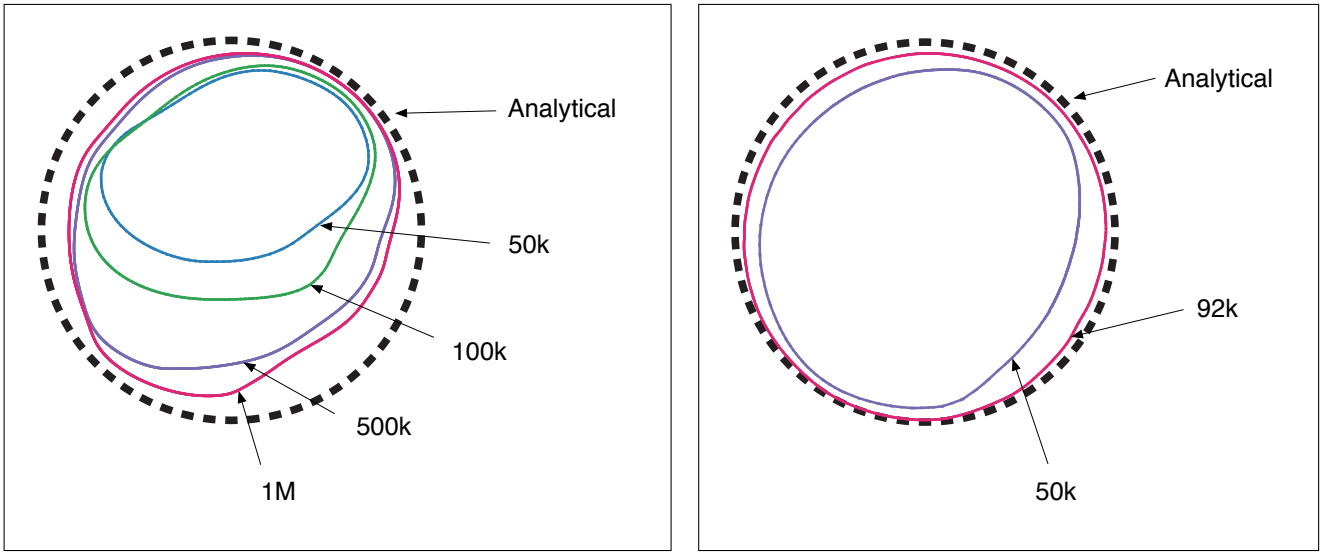


FIGURE 8. Comparison of the bubble's interface at $t = 10\frac{T}{2}$. Left: uniform meshes. Right: unsteady mesh adaptation.

$$\begin{cases} u(x,y,z,t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(2\pi\frac{t}{T}) \\ v(x,y,z,t) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(2\pi\frac{t}{T}) \\ w(x,y,z,t) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(2\pi\frac{t}{T}) \end{cases} \tag{8}$$

The initial bubble's radius is $R = 0.15$ and it is centered in $(x_0, y_0, z_0) = (0.35, 0.35, 0.35)$. The velocity field's period is $T = 6$. The bubble was chosen to be advected from time $t = 0$ to $t = 2\frac{T}{2}$ and a spatial error is computed at $t = \frac{T}{2}$ and $t = T$.

Four simulations were run using uniform meshes containing from 125k to 32M vertices. One mesh adaptation was run with an increasing mesh complexity at each iteration in the main loop (see Section 1.3): from ∼50k vertices for the first iteration to ∼350k vertices for the last one. A summary of the simulations in 3D is presented in Table 2.2. Note that the CPU timing given for i-th iteration of the main loop includes the timings of the

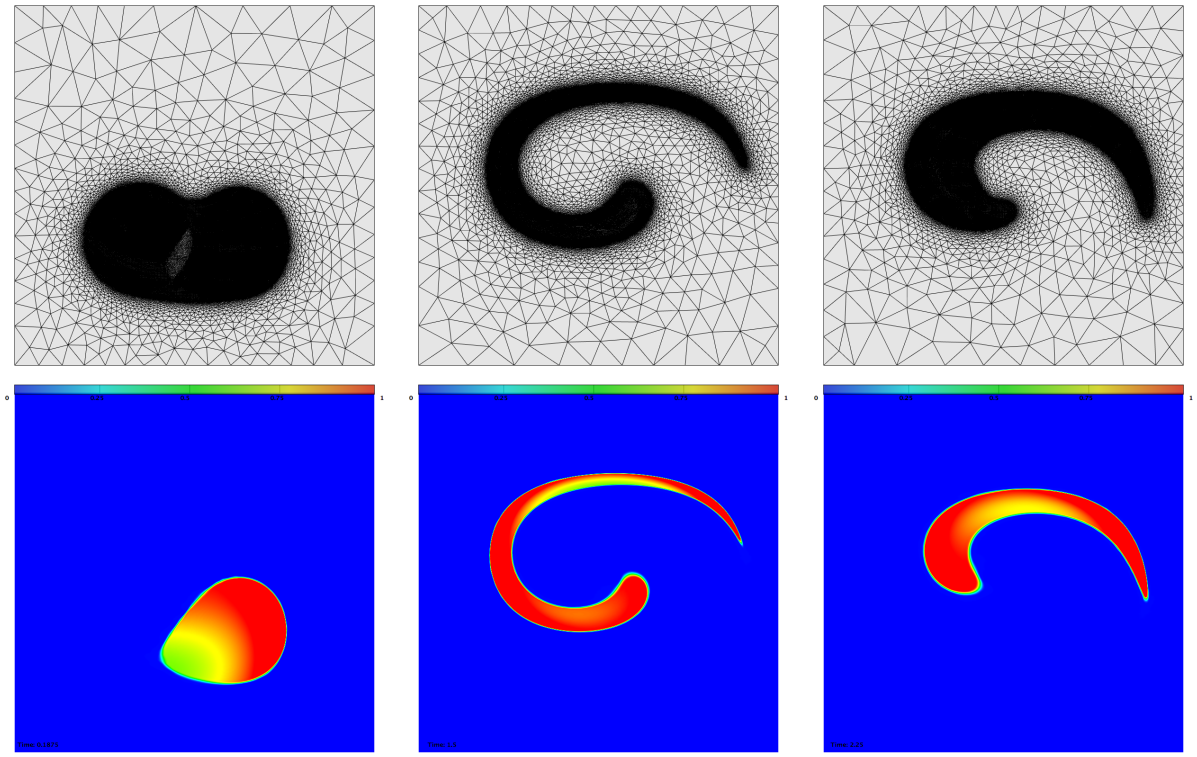


FIGURE 9. Adapted meshes and the corresponding densities at times $t = \frac{T}{16}, \frac{T}{4}$ and $\frac{3T}{8}$.

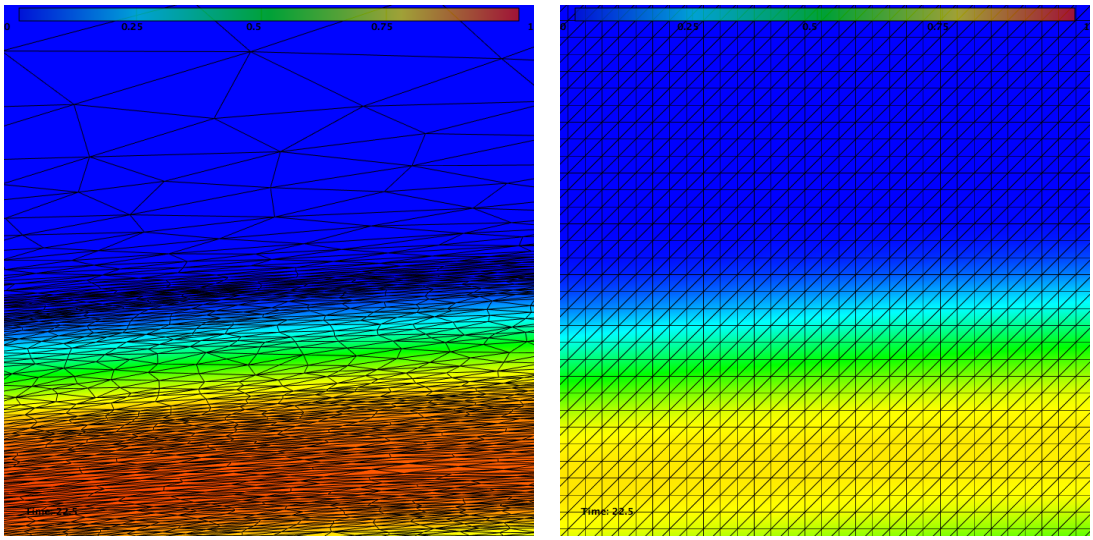


FIGURE 10. Close-up views of meshes around the bubble's interface at time $t = 7\frac{T}{2} + \frac{T}{4}$. Left: adapted anisotropic mesh, 50k vertices. Right: uniform mesh, 1M vertices.

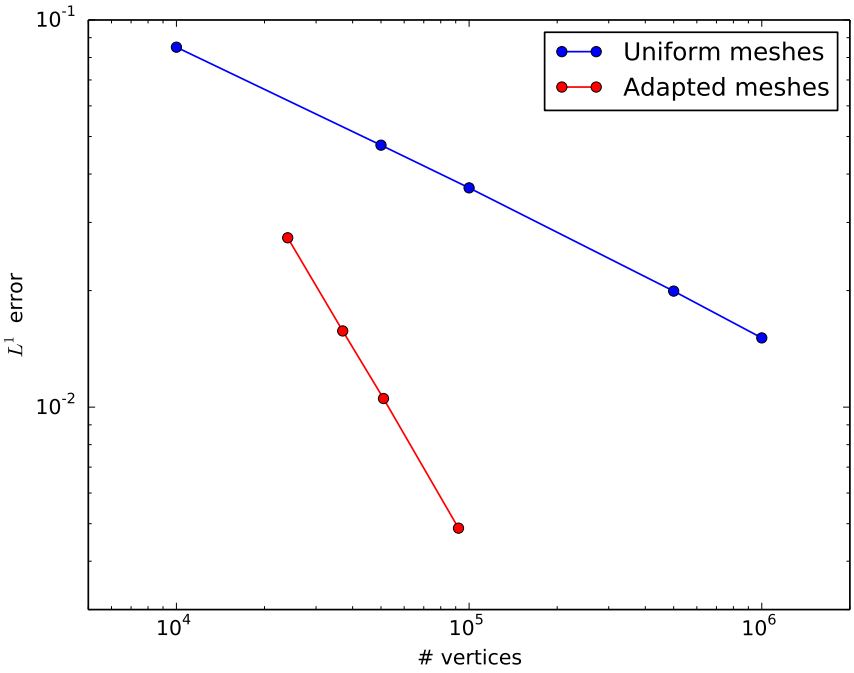


FIGURE 11. Comparison of the spatial L^1 error in 2D at time $t = 10\frac{T}{2}$ vs number of vertices.

iterations from the first to the ith.

The 3D simulations were run on four 2.00GHz ten-core Intel Xeon chips with 3Tb of RAM. The time frame $[0, T]$ was divided into 64 sub-intervals and mesh generations were performed on 32 cores (one process per core at the time).

A spatial L^1 error ϵ was computed at time $t = T$:

$$\epsilon = \sum_i |C_i| \, |\rho_{i,\texttt{exact}} - \rho_{i,h}|$$

where $|C_i|$ is the volume of the finite volume cell associated to vertex P_i, $\rho_{i,\texttt{exact}}$ is the exact solution at vertex P_i and $\rho_{i,h}$ is the computed solution. The mesh convergence is presented in Figure 16, and the CPU timings in Figure 17. One would need a uniform mesh of approximately 10^{12} vertices to reach the final spatial error obtained using unsteady mesh adaptation (349k vertices), which would require years of computation using this numerical model.

The bubble's interface (iso-value $\rho = 0.95$) is depicted for several physical times in Figure 14 for the uniform case (32 Million vertices), and in Figure 15 for the adapted case (349k vertices). The interface's conservation is significantly improved using mesh adaptation. Several views of the adapted meshes are given in Figure 18. Views of the uniform mesh containing 4 Million vertices are depicted in Figure 19.

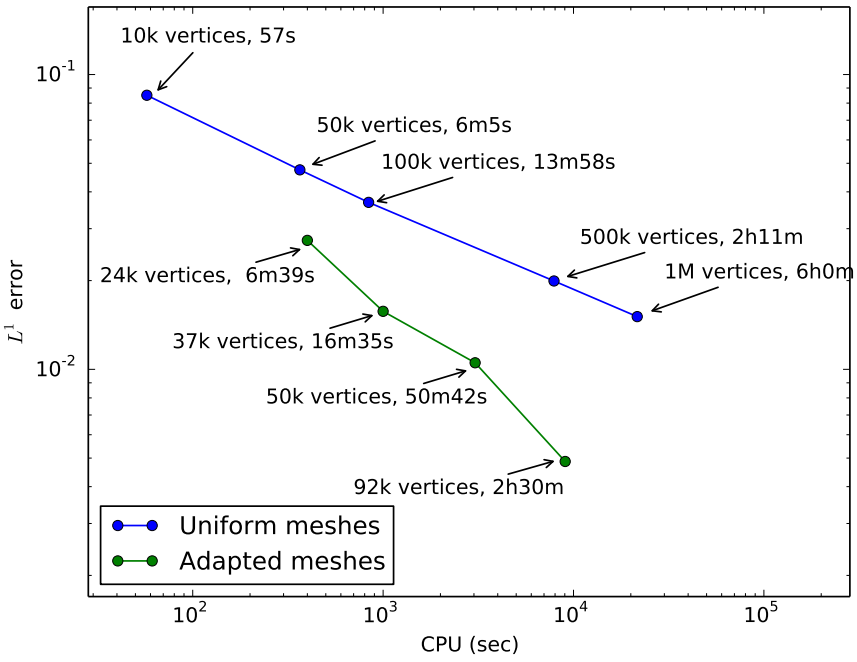


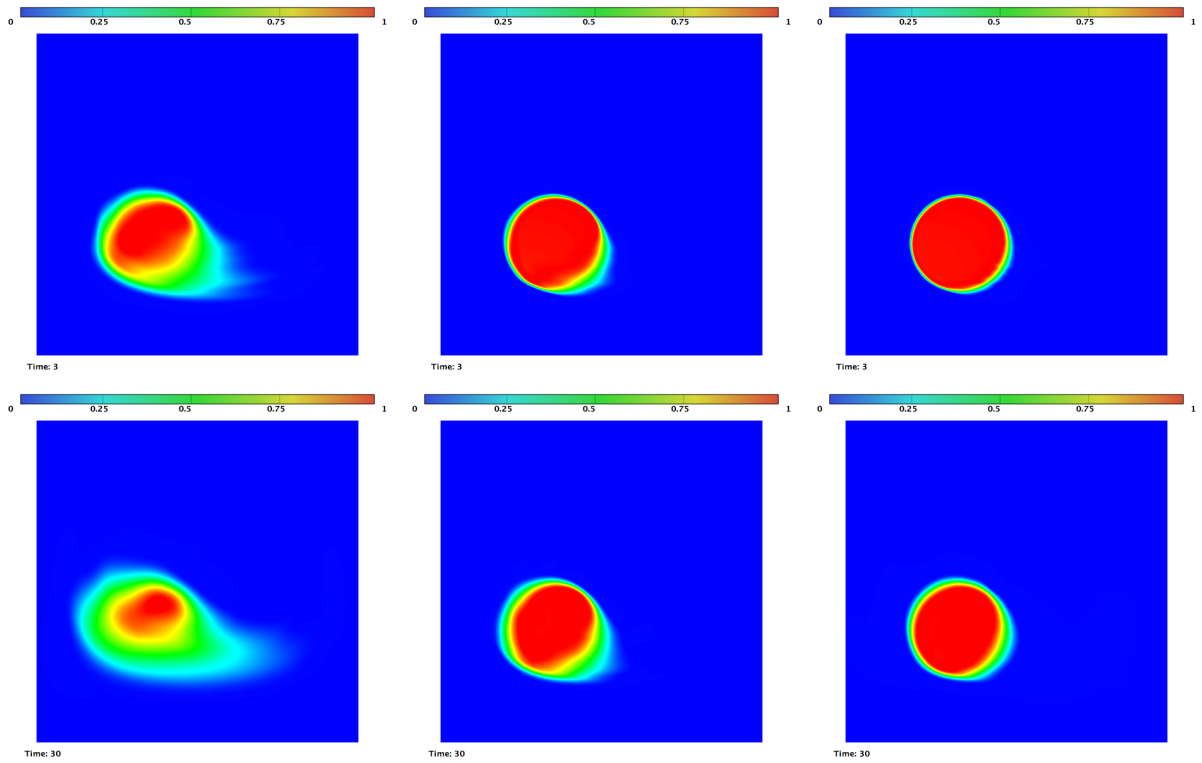
FIGURE 12. Spatial L^1 error in 2D at time $t = 10\frac{T}{2}$ vs the total CPU time of each simulation.

# ver	type	#procs	Total CPU	L^1 error ($t = T$)
125k	uniform	8	87s	2.43e-02
500k	uniform	16	7m55s	2.11e-02
4M	uniform	32	1h23m	1.49e-02
32M	uniform	32	13h0m	9.92e-03
51k	adapted (1st ite)	32	36m	1.60e-02
93k	adapted (2nd ite)	32	2h15m	7.86e-03
187k	adapted (3rd ite)	32	7h8m	4.74e-03
234k	adapted (4th ite)	32	15h50m	3.17e-03
292k	adapted (5th ite)	32	1d3h13m	2.41e-03
349k	adapted (6th ite)	32	1d6h48m	2.01e-03

TABLE 2. Summary of the simulations for the Kothe-Rider test in 3D. For the mesh adaptations, the given number of vertices corresponds to the mesh for the time $t = T$.

3. CONCLUSION

Anisotropic mesh adaptation was compared to uniform meshes for the high-fidelity prediction of bubble motion. Mesh convergence is dramatically improved for the Kothe-Rider test case using mesh adaptation. In 2D, the final spatial error observed in 6 hours with an uniform mesh of 1M vertices can be achieved in 20 minutes using mesh adaptation. Moreover, it would take 12 days and 22 hours for uniform meshes to reach the final spatial error observed after the mesh adaptation which took 2 hours and 30 minutes. In 3D, a uniform mesh of 10^{12} vertices would give the same accuracy as obtained in 1 day and 6 hours using an adapted mesh of 349k



$50k$ vertices (uniform) $1000k$ vertices (uniform) $50k$ vertices (adapted)

FIGURE 13. Comparison of the mapping of the density at times $t = \frac{T}{2}$ (top row) and $t = 10\frac{T}{2}$ (bottow row). Three computations are compared: two uniform meshes of 50k vertices and 1M vertices (1st and 2nd column resp.) and adapted meshes ($\sim 50k$ vertices each, 3rd column).

vertices, which would then take years of computation using the same numerical model.

The results obtained with mesh adaptation could be improved in several ways, including (i) using a level-set method, (ii) generating metric-aligned adapted meshes [25] in order to better handle strong anisotropy in the interface, and (iii) optimizing the number of sub-intervals during the adaptation as well as the prescribed number of vertices, which can dramatically impact the total CPU time of the simulation.

Acknowledgments. Many thanks to Adrien Loseille for his tips on the unsteady algorithm, to Arthur Talpaert for providing the test case's description, to Nicolas Barral for his work on the advection module in `Wolf`, and to Frédéric Alauzet and Richard James for the proofreading.

REFERENCES

[1] W. J. Rider and D. B. Kothe, "Reconstructing volume tracking," *J. Comput. Phys.*, vol. 141, pp. 112–152, Apr. 1998.
[2] Y. Penel, A. Mekkas, S. Dellacherie, J. Ryan, and M. Borrel, "Application of an amr strategy to an abstract bubble vibration model," *19th AIAA Comp. Fluid Dyn. Conf. Proc*, 2009.
[3] A. Talpaert, "Analysis of the efficiency and relevance of the berger-rigoutsos and the livne cluster creation algorithms for patch-based amr in the case of thin flagged areas," in *Poster presented at the 42nd CANUM*, 2014.
[4] R. Claisse, V. Ducrot, and P. Frey, "Level sets and anisotropic mesh adaptation," 2008.

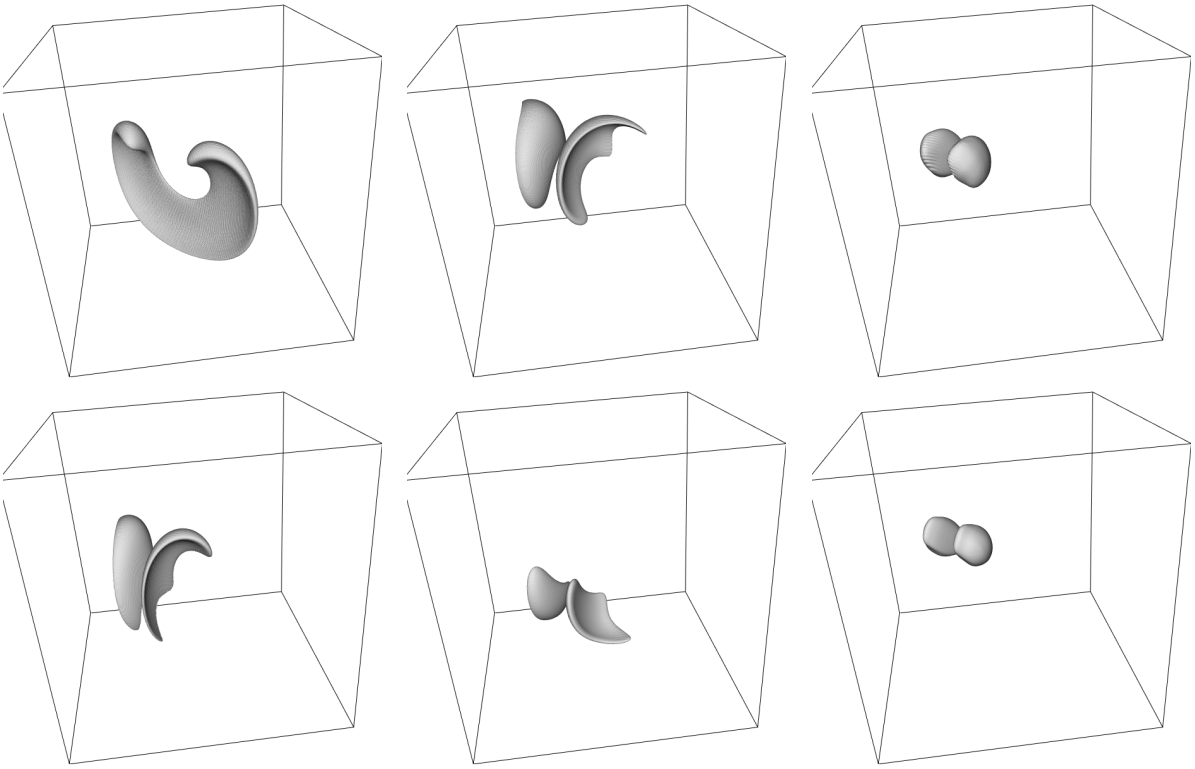


FIGURE 14. Result of the computation on the uniform mesh containing 32 Million vertices. The bubble's interface (iso-surface $\rho = 0.95$) at times $t = \frac{T}{8}, \frac{T}{4}, \frac{T}{2}, \frac{3T}{4}, \frac{7T}{8}$ and T.

[5] C. Bui, C. Dapogny, and P. Frey, "An accurate anisotropic adaptation method for solving the level set advection equation," *International Journal for Numerical Methods in Fluids*, vol. 70, no. 7, pp. 899–922, 2012.

[6] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau, "Anisotropic unstructured mesh adaptation for flow simulations," *Int. J. Numer. Meth. Fluids*, vol. 25, pp. 475–491, 1997.

[7] P. J. Frey and F. Alauzet, "Anisotropic mesh adaptation for CFD computations," *Comput. Methods Appl. Mech. Engrg.*, vol. 194, no. 48-49, pp. 5068–5082, 2005.

[8] C. Gruau and T. Coupez, "3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric," *Comput. Methods Appl. Mech. Engrg.*, vol. 194, no. 48-49, pp. 4951–4976, 2005.

[9] X. L. Li, M. S. Shephard, and M. W. Beall, "3D anisotropic mesh adaptation by mesh modification," *Comput. Methods Appl. Mech. Engrg.*, vol. 194, no. 48-49, pp. 4915–4950, 2005.

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

[11] B. V. Leer, "Towards the ultimate conservative difference scheme i. The quest of monotonicity," *Lecture notes in physics*, vol. 18, pp. 163–168, 1973.

[12] 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*, pp. 337–357, Springer, 2009.

[13] P. J. Frey and P.-L. George, *Mesh generation. Application to finite elements*. Paris, Oxford: Hermès Science, 2000.

[14] A. Loseille and F. Alauzet, "Continuous mesh framework. Part I: well-posed continuous interpolation error," *SIAM J. Numer. Anal.*, vol. 49, no. 1, pp. 38–60, 2011.

[15] A. Loseille and F. Alauzet, "Continuous mesh framework. Part II: validations and applications," *SIAM J. Numer. Anal.*, vol. 49, no. 1, pp. 61–86, 2011.

[16] C. Bottasso, "Anisotropic mesh adaption by metric-driven optimization," *Int. J. Numer. Meth. Engng*, vol. 60, pp. 597–639, 2004.

[17] G. Compère, J.-F. Remacle, J. Jansson, and J. Hoffman, "A mesh adaptation framework for dealing with large deforming meshes," *Int. J. Numer. Meth. Engng*, vol. 82, pp. 843–867, 2010.

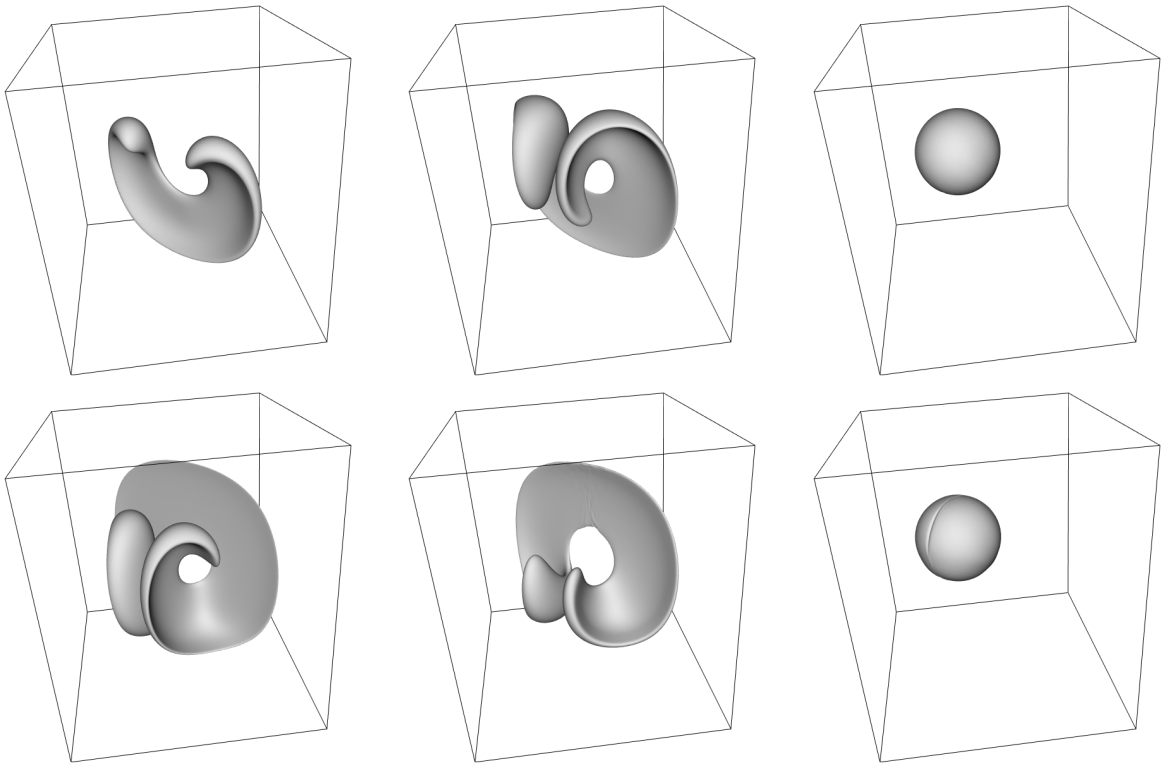


FIGURE 15. Result of the final 3D mesh adaptation. The bubble's interface (iso-surface $\rho = 0.95$) at times $t = \frac{T}{8}, \frac{T}{4}, \frac{T}{2}, \frac{3T}{4}, \frac{7T}{8}$ and T.

[18] C. Dobrzynski and P. J. Frey, "Anisotropic delaunay mesh adaptation for unsteady simulations," in *Proc. of 17th Int. Meshing Rountable*, pp. 177–194, Springer, 2008.

[19] P.-L. George, "`Gamanic3d`, adaptive anisotropic tetrahedral mesh generator," technical Note, INRIA, 2003.

[20] W. Jones, E. Nielsen, and M. 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.

[21] 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.

[22] A. Loseille and V. Menier, "Serial and parallel mesh modification through a unique cavity-based primitive," in *Proceedings of the 22nd International Meshing Roundtable* (X. Jiao and J.-C. Weill, eds.), 2013.

[23] F. Alauzet and M. Mehrenberger, "P1-conservative solution interpolation on unstructured triangular meshes," Research Report RR-6804, 2009.

[24] 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.*, vol. 222, pp. 592–623, 2007.

[25] A. Loseille, "Metric-orthogonal anisotropic mesh generation," in *Proceedings of the 23rd International Meshing Roundtable*, 2014.

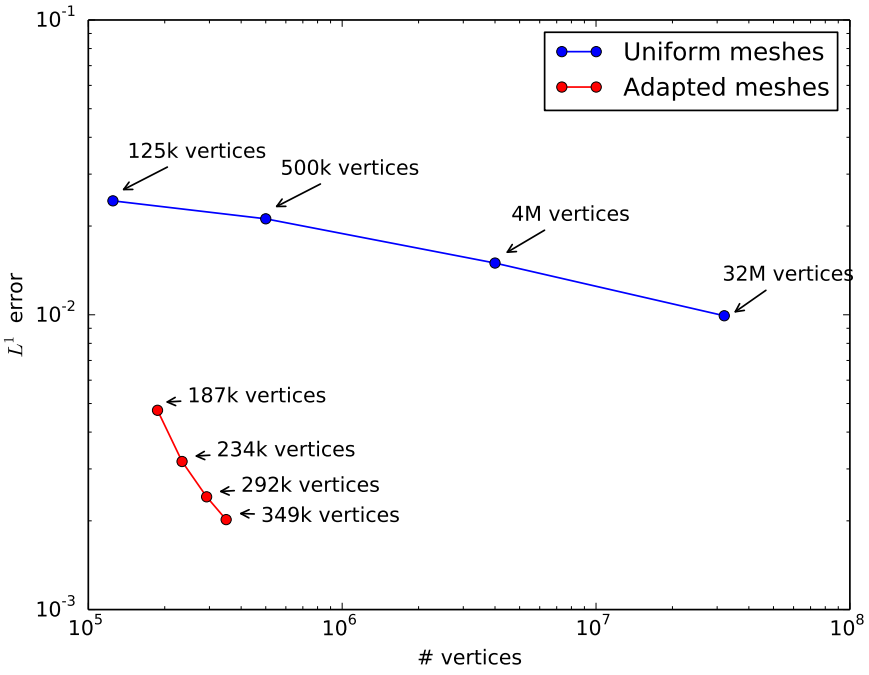


FIGURE 16. Comparison of the spatial L^1 error in 3D at time $t = T$ vs number of vertices.

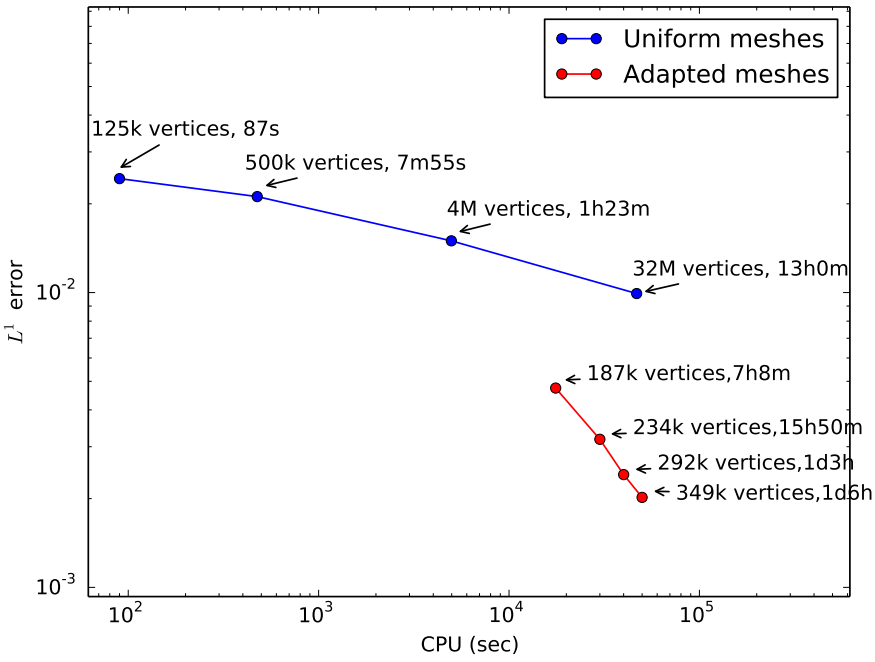


FIGURE 17. Spatial L^1 error in 3D at time $t = T$ vs the total CPU time of each simulation.

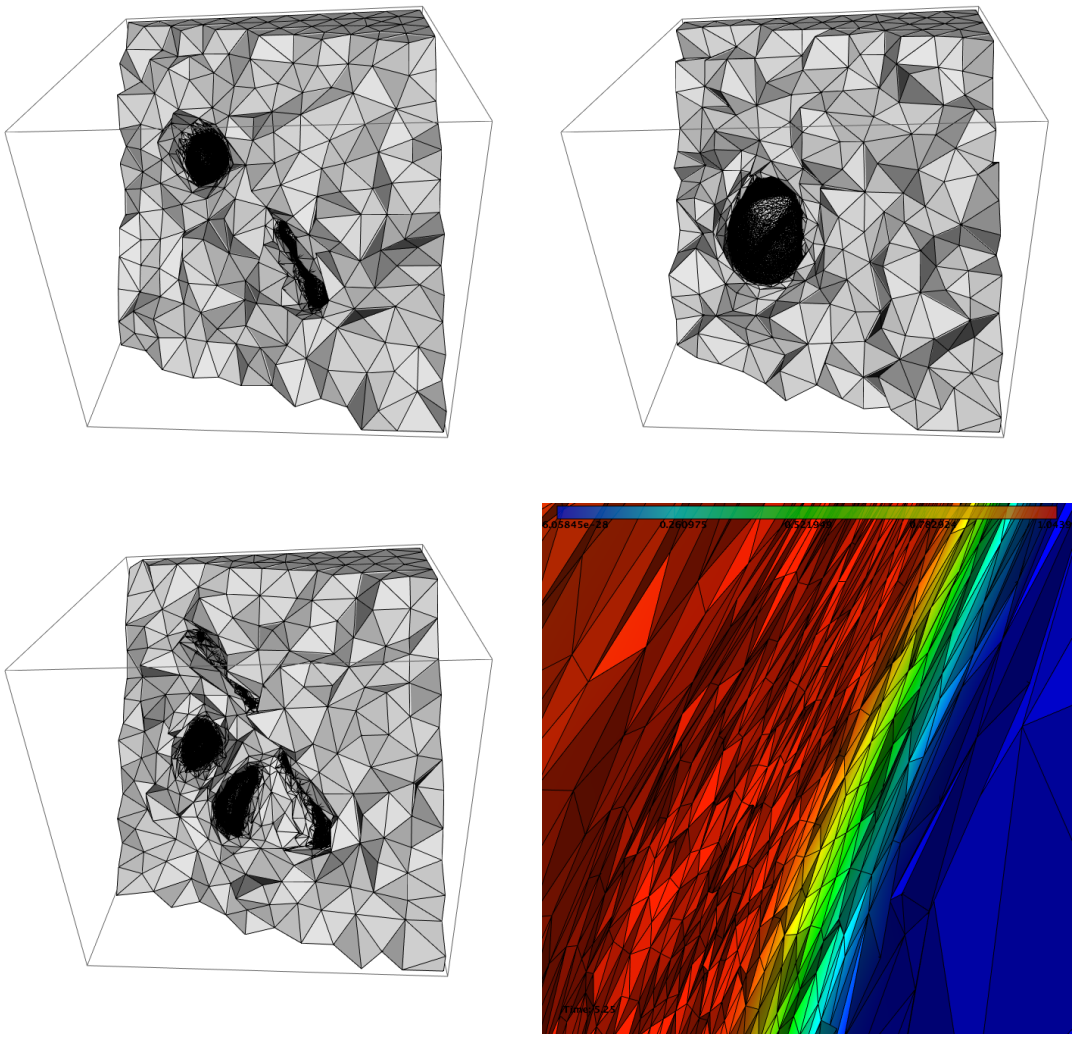


FIGURE 18. Adapted meshes for the 3d bubble motion. Top: cuts in the volume at times $t = \frac{T}{8}$ and $\frac{T}{2}$. Bottom: cuts in the volume at time $t = \frac{3T}{4}$ (right: close-up view of the bubble's interface).

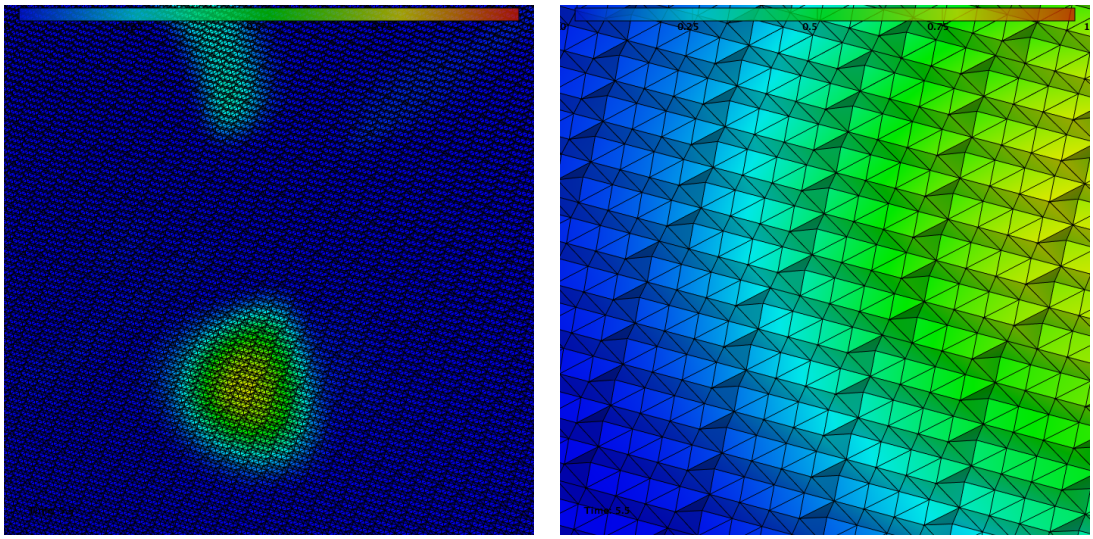


FIGURE 19. Cut in the volume of the 4M vertices uniform mesh with its solution at time $t = 0.91T$ (two close-up views).