AN ACCURATE ANISOTROPIC ADAPTATION METHOD FOR SOLVING THE LEVEL SET ADVECTION EQUATION

CUC BUI[1], CHARLES DAPOGNY[2,3], AND PASCAL FREY[2]

[1] *CEA Saclay, DEN-DM2S-SFME, Gif sur Yvette.*
[2] *UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France.*
[3] *Renault DREAM-DELT'A Guyancourt, France.*

Abstract.
In the present paper, a mesh adaptation process for solving the advection equation is presented, with a particular emphasis on the case it implicitly describes an evolving surface. It mainly relies on a numerical scheme based on the method of characteristics. This low-order scheme is thoroughly analyzed on the theoretical side. An anisotropic error estimate is derived and interpreted in terms of the Hausdorff distance between the exact and approximated surfaces. The computational mesh is then adapted according to the metric supplied by this estimate. The whole process enjoys a good accuracy as far as the interface resolution is concerned. Some numerical features are discussed and several classical examples are presented and commented in two or three dimensions.

Key words. Advection equation, level set method, method of characteristics, anisotropic mesh adaptation.

1. Introduction

Since the seminal work in [28], implicit descriptions of surfaces or interfaces between different domains have become increasingly popular. Roughly speaking, it amounts to considering a surface embedded in $\mathbb{R}^d$ as the zero level set of a scalar function defined on the whole ambient space. The motion of such a surface through a given velocity field turns out to be parameterized by an advection equation for the associated scalar function : see [34] or [29] for various topics around the level set method, or applications to physical problems. On the other hand, numerous methods are available when it comes to solving the advection equation : see [18] for a review on the topic, or [21] for numerical comparisons between several existing methods for the advection equation in the context of level set methods.

In the present paper, we mean to solve the standard advection equation associated to a given velocity field V

$$\frac{\partial u}{\partial t} + V.\nabla u = 0$$

in the particular case the advected scalar function $u(t,x)$ is a level set function associated to an evolving domain $\Omega(t)$, that is, for all $t > 0$, $x \in \mathbb{R}^d$ ($d = 2, 3$ in our examples)

$$\left\{ \begin{array}{lll} u(t,x) < 0 & \text{if} & x \in \Omega(t) \\ u(t,x) = 0 & \text{if} & x \in \partial\Omega(t) \\ u(t,x) > 0 & \text{if} & x \in {}^c\overline{\Omega(t)} \end{array} \right. . \tag{1}$$

Celebrated methods are available in case the numerical approximations are held on a cartesian grid (see e.g. [17] [27]). On the contrary, the whole work of this paper unfolds in the context of fully unstructured background mesh, for we believe it is of independent relevance. For instance, it can be used for tracking an interface which evolves in a computational domain with a complex geometry (as often happens in industrial computations) that is more accurately described by an unstructured mesh. Furthermore, in several applications, the velocity according to which the interface evolves stems from mechanical computations (flow solvers

Date: December 2, 2011.

in computational fluid dynamics, shape-sensitivity analysis in shape optimization,...) that require a mesh of the implicitly-defined domain Ω at each step of the process. It may thus be desirable that the computational mesh used for advection also encloses a discretization of this domain as a submesh, and this is only possible for it is fully unstructured. Actually, the process described in this paper has already been used in such cases in the previous works [11] [3].

We aim at getting a sharp approximation of the domain $\Omega(t)$, hence the choice of conforming finite elements for the discretization of u, although discontinuous Galerkin methods generally prove very efficient in this context (see [26] for instance). Furthermore, the surface under evolution may or may not be related to a physical problem (e.g. fluid mechanics) ; for this purpose, we undertook not to address mass conservation enforcement, however crucial this aspect might prove in several applications. Of course, it is possible to couple the proposed method with a mass conservation, or mass restoration scheme.

Our main goal is to suggest a mesh adaptation process so as to control the accuracy of the computation. To achieve this, we rely on the classical method of characteristics (see [30] for an exhaustive review of its use in many fluid problems) to solve the advection equation. Although it is admittedly low-order as well as very diffusive, it is simple to implement numerically, and we will see it is amenable to an error analysis which yields straightforwardly an anisotropic error estimate for the Hausdorff distance between the evolving interface and its computed approximation. Then, using anisotropic mesh adaptation on the computational mesh according to the obtained estimate brings a close approximation of the advected quantity where it is needed.

The outline of this paper is as follows : in section 2, we briefly recall some basic theoretical facts around the advection equation that will be extensively used. Section 3 is devoted to deriving a numerical method for the considered advection equation : this method is presented in subsection 3.1, then analyzed in both subsection 3.2, where an a priori error estimate is proved, and subsection 3.3, where this estimate is specialized to the case of interest, that is when the advected scalar function is a level-set function associated to an evolving interface. Section 4 tackles the issue of deducing a mesh adaptation process from the previous error estimate. After recalling some classical material about metric-based mesh adaptation in subsection 4.1, an appropriate adaptation method is detailed in subsection 4.2. In section 5, we emphasize on two crucial numerical aspects of the proposed technique : the need for a mesh gradation control in the way the computational mesh is adapted, and the role played by redistancing in our computations. Eventually, in section 6, several numerical examples in 2d and 3d are developped to assess the proposed method.

2. Some theoretical facts around the advection equation

Given an initial function $u^{in} : \mathbb{R}^d \to \mathbb{R}$ and a velocity field $V : \mathbb{R}^d \to \mathbb{R}^d$ defined on the whole space, we consider the Cauchy problem for the *advection equation* : find $u \in \mathcal{C}^1([0,T] \times \mathbb{R}^d)$ such that

$$\left\{ \begin{array}{cc} \dfrac{\partial u}{\partial t}(t,x) + V(t,x).\nabla u(t,x) = 0, & (t,x) \in (0,T) \times \mathbb{R}^d \\ u(0,x) = u^{in}(x), & x \in \mathbb{R}^d \end{array} \right. . \tag{2}$$

Throughout this paper, unless stated otherwise, we will make (at least) the following assumptions on the velocity field $V : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$:

$$V \text{ and } \nabla V \text{are continuous over } [0,T] \times \mathbb{R}^d. \tag{3}$$

$$\text{There exists } \kappa > 0 \text{ such that } ||V(t,x)|| \leq \kappa\,(1 + ||x||) \text{ for all } (t,x) \in [0,T] \times \mathbb{R}^d. \tag{4}$$

We then recall the following classical result ([22]) related to the advection equation.

Theorem 2.1. *Suppose (3) and (4) hold, and let $u^{in} \in \mathcal{C}^1\left(\mathbb{R}^d\right)$. For all $0 \leq t \leq T$ and $x \in \mathbb{R}^d$, denote $s \mapsto X(s,t,x)$ the characteristic curve emerging from point x at time t, solution to the ODE*

$$\left\{ \begin{array}{rcl} \dfrac{dX}{ds}(s,t,x) & = & V(s, X(s,t,x)) \\ X(t,t,x) & = & x \end{array} \right. , \tag{5}$$

then

- *For every point $x \in \mathbb{R}^d$ and every time $t \in [0,T]$, the curve $s \mapsto X(s,t,x)$ is well-defined over $[0,T]$. Furthermore, for every $s,t \in [0,T]$, the application $\mathbb{R}^d \ni x \mapsto X(s,t,x) \in \mathbb{R}^d$ is a $\mathcal{C}^1$, orientation-preserving diffeomorphism.*
- *Cauchy problem (2) has a unique solution $u \in \mathcal{C}^1([0,T] \times \mathbb{R}^d)$, given by*

$$u(t,x) = u^{in}(X(0,t,x)) \tag{6}$$

Note that the above result holds for the advection equation over the whole space $\mathbb{R}^d$, and for an evolution driven by a (smooth) velocity field which satisfies some kind of 'non blow-up' behaviour at infinity. In numerical practice, we restrict ourselves to a (large) bounded computational domain, so that this property will always prove 'numerically true'. However, we will still solve the advection equation as if it were considered over $\mathbb{R}^d$, so that formula (6) will stand, the reason being that in this work, we are mainly interested in the motion of a surface evolving 'far' from the boundary of the computational domain, so that the values at stake do not see the boundary. Actually, Cauchy problem (2) is ill-posed on a bounded domain, and its analysis is far more difficult : one has to introduce boundary conditions in the so-called *reentrant boundary* where the velocity field V requires information from outside the domain. See [6] for a complete study of this general problem.

In particular, Theorem 2.1 has the following interesting consequence in case the scalar quantity at stake stands for a level-set function associated to a smooth surface :

Corollary 2.2. *Under the hypothesis of theorem 2.1, suppose u^{in} is a level-set function associated to a regular domain $\Omega^{in} \subset \mathbb{R}^d$ in the sense that (1) hold, such that for all $x \in \partial\Omega^{in}$, $\nabla u^{in}(x) \neq 0$. Then for all $t \in [0,T]$, $\Omega(t) := \big\{x \in \mathbb{R}^d \backslash u(t,x) < 0\big\}$ is a regular domain with regular boundary $\partial\Omega(t) := \big\{x \in \mathbb{R}^d \backslash u(t,x) = 0\big\}$, and is diffeomorphic to Ω^{in} through the mapping $X(0,t,.)$.*

3. The proposed numerical method, and its error analysis

3.1. **A numerical method for the advection equation based on the method of characteristics.** Suppose the whole space $\mathbb{R}^d$ is endowed with a mesh $\mathcal{T}$, and consider an associated Lagrange finite element space $V_{\mathcal{T}}$ (e.g. $\mathbb{P}^1$ or $\mathbb{P}^2$ finite elements). Given a time step Δt and an interval $\big[T^n, T^{n+1}\big]$, $T^{n+1} = T^n + \Delta t$ (see remark below for a discussion), and an initial state $u(T^n,.)$ which fulfills the hypothesis of Theorem 2.1, we intend to approximate the function $u(T^{n+1},.)$, where $u(t,.)$ is the unique solution to the Cauchy problem

$$\left\{ \begin{array}{cc} \dfrac{\partial u}{\partial t}(t,x) + V(t,x).\nabla u(t,x) = 0, & (t,x) \in (T^n, T^{n+1}) \times \mathbb{R}^d \\ u(T^n,x) = u^{in} & \forall x \in \mathbb{R}^d \end{array} \right. .$$

To this end, we only have an approximation $\widetilde{V}$ of vector field V, as well as a piecewise affine approximation $u^{T^n} \in V_{\mathcal{T}}$ of $u(T^n,.) = u^{in}$, and we also look for an approximation of $u(T^{n+1},.)$ as a function $u^{T^{n+1}} \in V_{\mathcal{T}}$, that is to say we only need to compute the desired approximation $u^{T^{n+1}}(x)$ for every *degree of freedom* x of $\mathcal{T}$. In order to mimic the exact formula (6), as is well-known with the method of characteristics, we carry out two steps :

- In a first step of *time discretization*, we compute an approximation $\widetilde{Y}(T^n, T^{n+1}, x)$ of the solution $Y(T^n, T^{n+1}, x)$ to the *approximated* characteristic curve at time T^n :

$$\left\{ \begin{array}{rcl} \dfrac{dY}{ds}(s, T^{n+1}, x) & = & \widetilde{V}(s, Y(s, T^{n+1}, x)) \\ Y(T^{n+1}, T^{n+1}, x) & = & x \end{array} \right.$$

  To achieve this, any classical method for solving ordinary differential equations can be used (Euler's method, or a more accurate Runge-Kutta method, see [16]). For instance, introducing a subintegration time step $\delta t << \Delta t$ and subdividing $]T^n, T^{n+1}[= \cup_{l=0}^{L}]t^l, t^{l+1}[$, with $t^l := T^n + l\delta t$, $l = 0, ..., L$, Euler's method yields

$$\left\{ \begin{array}{c} \widetilde{Y}(T^{n+1}, T^{n+1}, x) = x \\ \widetilde{Y}(t^l, T^{n+1}, x) = \widetilde{Y}(t^{l+1}, T^{n+1}, x) - \delta t \widetilde{V}(\widetilde{Y}(t^{l+1}, T^{n+1}, x)) \quad \text{for } l = 0, ..., L-1 \end{array} \right.$$

- Then, a *spatial approximation* step is to be performed ; in other words, if K is a simplex of $\mathcal{T}$ such that $\widetilde{Y}(T^n, T^{n+1}, x) \in K$, we have

$$\begin{aligned} u^{in}(X(T^n, T^{n+1}, x)) &\approx u^{in}\left(\widetilde{Y}(T^n, T^{n+1}, x)\right) \\ &\approx u^{T^n}\left(\widetilde{Y}(T^n, T^{n+1}, x)\right) \end{aligned},$$

the latter expression being evaluated on basis of the discrete set of values of u^n at the degrees of freedom belonging to simplex K.

Remarks

(1) *Time-stepping.* The above method uses two different time steps. The first one, $\Delta t = T^{n+1} - T^n$, is a 'large' time step, mainly related to physics : in most interesting problems, the velocity field V used for advection over $\left[T^n, T^{n+1}\right]$ stems from a mechanical computation at time T^n. Time step Δt is then the period of time for which we assume this velocity physically relevant. The second one, δt is a subintegration time step ; it is fictitious and merely involved for the integration of the characteristic curves. It is the only one really involved by the method of characteristics. In the sequel, we will often focus on a generic period of time $[0, T]$, standing for any $\left[T^n, T^{n+1}\right]$.

(2) *About numerical discretization.* The above method amounts to solving an ordinary differential equation at each degree of freedom of the computational mesh, and involves neither matrix inversion, nor quadrature formulas for approximating integrals. Consequently, it is computationally efficient en practice, and as accurate as can be a spatial first-order scheme, as will be seen. However, as is, it is not readily extended to more general problems, such as convection-diffusion-reaction problems. Moreover, it has been observed (see [31]) that simple first-order characteristic-based numerical scheme for treating a convection term are generally very diffusive, which could be unacceptable for several applications (such as fluid dynamics). For these reasons, characteristic-Galerkin numerical schemes are often used in combination with higher-order finite elements. See [8], [9] for an exhaustive presentation of several aspects of general characteristic-Galerkin finite element methods. In this paper, we will overcome this low-order drawback resorting to a mesh adaptation procedure.

3.2. **A priori error analysis of the proposed method.** The goal of this section is to develop an a priori error estimate for the scheme presented in section 3.1. As already mentioned, neither the idea of using the method of characteristics for advection, nor its error analysis is new : see e.g. [30] [32] for considerations in a far more general context. However, to our knowledge, the following *a priori* error estimate, on which relies our adaptation scheme, is not so classical under this form (even if it is actually a variation of existing ones).

In the following, we still consider equation (2) for a generic period of time $[0, T]$ and over the whole space $\mathbb{R}^d$, which is endowed with two separate simplicial meshes $\mathcal{T}$ (which is to carry the approximations of functions $u(t, x)$) and $\mathcal{T}'$, on which we have an approximation $\widetilde{V}$ of a continuous vector field V. We denote $V_{\mathcal{T}}$ (resp. $V_{\mathcal{T}'}$) the space of continuous functions over $\mathbb{R}^d$, whose restriction to every simplex $K \in \mathcal{T}$ (resp. $K' \in \mathcal{T}'$) is a $\mathbb{P}^1$ polynomial function. For every continuous function f over $\mathbb{R}^d$, we denote $\pi_{\mathcal{T}}(f)$ (resp. $\pi_{\mathcal{T}'}(f)$) the $\mathbb{P}^1$-interpolate of f over $\mathcal{T}$ (resp. $\mathcal{T}'$), i.e. the unique function in $V_{\mathcal{T}}$ (resp. $V_{\mathcal{T}'}$) which coincides with f at every node of $\mathcal{T}$ (resp. $\mathcal{T}'$).

Theorem 3.1. *Assume moreover that V is a stationary, uniformly Lipschitz continuous vector field over $\mathbb{R}^d$, with constant k, and u^{in} is a $\mathcal{C}^2$, uniformly Lipschitz continuous function over $\mathbb{R}^d$ with constant k' (so that (3) and (4) are satisfied). Assume as well the approximation $\widetilde{V}$ of V is such that both $\sup_{x\in\mathbb{R}^d} ||V(x) - \widetilde{V}(x)||$ and $\sup_{K\in\mathcal{T}', x\in K} ||\nabla V(x) - \nabla\widetilde{V}|_T||$ are bounded.*
Let δt be a time step, and consider the sequence of times $t^n = n\delta t$, $0 = t^0 < t^1 < ... < t^N = T$. Denote $u^N \in V_{\mathcal{T}}$ the sought approximation of $u(T, .)$ defined as

$$\text{For each node } x \text{ of } \mathcal{T},\ u^N(x) = \pi_{\mathcal{T}}(u^{in})(\widetilde{Y}(0, T, x)), \tag{7}$$

where $\widetilde{Y}(0,T,x)$ is the approximation of the solution $Y(0,T,x)$ to the ODE

$$\left\{ \begin{array}{rcl} \dfrac{dY}{ds}(s,T,x) & = & \widetilde{V}(Y(s,T,x)) \\ Y(T,T,x) & = & x \end{array} \right. , \tag{8}$$

at time $s=0$ by means of Euler method with time step δt. Then there exists a constant C which only depends on V, such that

$$\begin{array}{r} ||u(T,.)-u^N||_{L^\infty(\mathbb{R}^d)} \leq ||u(T,.)-\pi_{\mathcal{T}}u(T,.)||_{L^\infty(\mathbb{R}^d)} + ||u^{in}-\pi_{\mathcal{T}}u^{in}||_{L^\infty(\mathbb{R}^d)} \\ +\dfrac{k'}{k}(e^{kT}-1)||V-\widetilde{V}||_{L^\infty(\mathbb{R}^d)} + \dfrac{k'}{k}CTe^{C\delta t}\delta t \end{array} \tag{9}$$

Proof. The proof will be split into two steps.

Step 1 : time approximation of the integral curves of vector field V. Given $x \in \mathbb{R}^d$, the aim is to estimate the approximation of $X(0,T,x)$ by $Y(0,T,x)$. Here, X stands for the exact integral curve associated to vector field V, solution of (5), and Y is the exact integral curve associated to the approximated vector field $\widetilde{V}$ (solution of (8)), which is itself to be numerically approximated by $\widetilde{Y}(0,T,x)$, obtained with a first-order Euler scheme. First, the use of Lemma 3.2 below allows to quantify the gap between $Y(0,T,x)$ and $\widetilde{Y}(0,T,x)$:

$$||Y(0,T,x)-\widetilde{Y}(0,T,x)|| \leq CTe^{C\delta t}\delta t$$

for a constant C which only depends on $\widetilde{V}$, or alternatively on V, owing to the asumptions made over the approximation of V by $\widetilde{V}$. Then, thanks to a variation of Gronwall's lemma recalled in appendix (see Lemma 7.1) for the approximation of $X(0,T,x)$ by $Y(0,T,x)$, we get

$$\begin{array}{rcl} ||X(0,T,x)-\widetilde{Y}(0,T,x)|| & \leq & ||X(0,T,x)-Y(0,T,x)|| + CTe^{C\delta t}\delta t \\ & = & \frac{(e^{kT}-1)}{k}||V-\pi_{\mathcal{T}'}V||_{L^\infty(\mathbb{R}^d)} + CTe^{C\delta t}\delta t \end{array} . \tag{10}$$

Step 2 : spatial approximation on mesh $\mathcal{T}$. We now turn to the error carried by the definition of the approximation u^N of function $u(T,.)$, for a *node* x of mesh $\mathcal{T}$:

$$\begin{array}{rcl} ||u(T,x)-u^N(x)|| & = & ||u^{in}\left(X(0,T,x))\right) - \pi_{\mathcal{T}}u^{in}\left(\widetilde{Y}(0,T,x)\right)|| \\ & \leq & ||u^{in}\left(X(0,T,x))\right) - u^{in}\left(\widetilde{Y}(0,T,x)\right)|| \\ & & \quad + ||u^{in}\left(\widetilde{Y}(0,T,x)\right) - \pi_{\mathcal{T}}u^{in}\left(\widetilde{Y}(0,T,x)\right)|| \\ & \leq & k'||X(0,T,x)-\widetilde{Y}(0,T,x)|| + ||u^{in}-\pi_{\mathcal{T}}u^{in}||_{L^\infty(\mathbb{R}^d)} \\ & \leq & \dfrac{k'}{k}(e^{kT}-1)||V-\pi_{\mathcal{T}'}V||_{L^\infty(\mathbb{R}^d)} + \dfrac{k'}{k}CTe^{C\delta t}\delta t + ||u^{in}-\pi_{\mathcal{T}}u^{in}||_{L^\infty(\mathbb{R}^d)}. \end{array}$$

Eventually, consider *any* point $x \in \mathbb{R}^d$, and let $K \in \mathcal{T}$ any simplex containing x. Denote $a_0,...,a_d$ its vertices, and $\lambda_0,...,\lambda_d$ the associated barycentric coordinates. It stems from the definition of u^N that

$$\begin{array}{rcl} u(T,x)-u^N(x) & = & u(T,x) - \displaystyle\sum_{i=0}^{d} \lambda_i(x)u^N(a_i) \\ & = & u(T,x) - \pi_{\mathcal{T}}u(T,x) + \displaystyle\sum_{i=0}^{d} \lambda_i(x)\left(u(T,a_i)-u^N(a_i)\right) \end{array} .$$

Finally, we get the following estimate

$$\begin{array}{r} ||u(T,.)-u^N||_{L^\infty(\mathbb{R}^d)} \leq ||u(T,.)-\pi_{\mathcal{T}}u(T,.)||_{L^\infty(\mathbb{R}^d)} + \dfrac{k'}{k}(e^{kT}-1)||V-\widetilde{V}||_{L^\infty(\mathbb{R}^d)} \\ +\dfrac{k'}{k}CTe^{C\delta t}\delta t + ||u^{in}-\pi_{\mathcal{T}}u^{in}||_{L^\infty(\mathbb{R}^d)}. \end{array}$$

□

In the first step of the above proof, we made use of the following lemma which is a version of the well-known estimate for Euler's method for an ODE, in case the velocity field is only continuous and piecewise differentiable, and its proof is a mere variation of the arguments in [16] for instance.

Lemma 3.2. *Let $x \in \mathbb{R}^d$, and $V \in (\mathcal{V}_{\mathcal{T}'})^d$ a continuous and piecewise affine vector field over $\mathbb{R}^d$, such that*

$$C_1 := \sup_{K \in \mathcal{T}'} ||\nabla V|_K|| < +\infty \quad ; \quad C_2 := \sup_{x \in \mathbb{R}^d} ||V(x)|| < +\infty. \tag{11}$$

Denote y the exact solution to the ODE

$$\left\{ \begin{array}{rcl} y'(s) & = & V(y(s)) \\ y(T) & = & x \end{array} \right. , \tag{12}$$

and y^0 its approximation at time $t^0 = 0$ obtained by a simple Euler method over the interval $[0, T]$, with time step δt. Then,

$$||y(0) - y^0|| \leq 2C_1C_2\, Te^{C_1\delta t}\delta t \tag{13}$$

Proof. As the vector field V is continuous and uniformly Lipschitz over $\mathbb{R}^d$ because of assumption (11), the exact solution y to the ODE (12) exists over $[0, T]$, is unique, and differentiable on this interval.

Introduce the sequence $y^0, ..., y^N = x$ of approximated values for y at times $t^0, ..., t^N$ obtained by Euler's method with time step δt. For any $n = 0, ..., N-1$, this means

$$y^n = y^{n+1} - \delta t V(y^{n+1}) \tag{14}$$

Next, subdivide the time interval $\left[t^n, t^{n+1}\right]$ as $t^n = r^0 < r^1 < ... < r^p = t^{n+1}$ for a certain p in such a way that for all $j = 0, ..., p-1$, $y\left(\left[r^j, r^{j+1}\right]\right)$ is included in a single simplex K^j of $\mathcal{T}'$. Then, y being differentiable on each subinterval $\left[r^j, r^{j+1}\right]$,

$$\begin{array}{rcl} y(t^n) & = & y(t^{n+1}) + \displaystyle\sum_{j=0}^{p-1} \left(y(r^j) - y(r^{j+1})\right) \\ & = & y(t^{n+1}) + \displaystyle\sum_{j=0}^{p-1} \left(V(y(r^{j+1}))(r^j - r^{j+1}) + \int_0^1 (1-s) f_j''(s) ds \right) \end{array} , \tag{15}$$

where we introduced $f_j(s) = y\left(r^{j+1} + s(r^j - r^{j+1})\right)$. Substracting (14) to (15) yields

$$\begin{array}{rcl} y(t^n) - y^n & = & y(t^{n+1}) - y^{n+1} + \displaystyle\sum_{j=0}^{p-1} \left(\left(V(y(r^{j+1})) - V(y^{n+1}) \right)(r^j - r^{j+1}) + \int_0^1 (1-s) f_j''(s) ds \right) \\ & = & y(t^{n+1}) - y^{n+1} + \displaystyle\sum_{j=0}^{p-1} \left(\left(V(y(r^{j+1})) - V(y(t^{n+1})) \right)(r^j - r^{j+1}) + \int_0^1 (1-s) f_j''(s) ds \right) \\ & & + \displaystyle\sum_{j=0}^{p-1} \left(V(y(t^{n+1})) - V(y^{n+1}) \right)(r^j - r^{j+1}) \end{array} .$$

Subdividing segment $\left[y(t^{n+1}), y^{n+1}\right]$ into smaller subsegments, each one being included into a single element $K \in \mathcal{T}'$, the last sum is such that

$$||\sum_{j=0}^{p-1} \left(V(y(t^{n+1})) - V(y^{n+1}) \right)(r^j - r^{j+1})|| \leq C_1 \delta t ||y(t^{n+1}) - y^{n+1}|| \tag{16}$$

Furthermore, using the same argument of decomposition over smaller subsegments of each interval $\left[r^j, t^{n+1}\right]$, $j = 0, ..., p-1$, we have, thanks to the hypothesis involving constants C_1, C_2,

$$||\sum_{j=0}^{p-1} \left(V(y(r^{j+1})) - V(y(t^{n+1})) \right)(r^j - r^{j+1})|| \leq C_1 C_2 \delta t^2 \tag{17}$$

Eventually, from the definition of f_j, we infer $f_j''(s) = (r^j - r^{j+1})^2 \nabla V|_{T_j} . V\left(y\left(r^{j+1} + s(r^j - r^{j+1})\right)\right)$. Thus, gathering (16), (17) and the latter, it comes

$$||y(t^n) - y^n|| \leq (1 + C_1\delta t)||y(t^{n+1}) - y^{n+1}|| + 2C_1C_2\delta t^2$$

From the so-called discrete Gronwall lemma (see [16] for instance), we conclude that

$$(18) \qquad \begin{aligned} ||y(0)-y^0|| &\leq e^{C_1T}||y(T)-y^N|| + 2C_1C_2\sum_{j=0}^{N-1} e^{C_1\delta t}\delta t^2 \\ &\leq 2C_1C_2Te^{C_1\delta t}\delta t \end{aligned},$$

which is the expected estimate. □

Remarks

- This estimate is low-order. As pointed out by Pironneau [30] [31], the presented method can be greatly improved by the choice of higher order methods in time, or spatial approximation. Thus, approximation of the integral curves of vector field V with a 4^{th} order Runge-Kutta scheme, or of the functions at stake by means of Lagrange $\mathbb{P}^2$-finite elements tremendously improves the quality of the obtained results. Unfortunately, there is no formal proof of these assertions (note that the previous proof cannot be generalized to those cases), however relevant they are in numerical practice.
- Many variations over this result are available : the vector field V could be analytically prescribed (as will be the case in the examples of section 6), and the same estimate holds, except that, understandably enough, the term $||V-\widetilde{V}||_{L^\infty(\mathbb{R}^d)}$ vanishes ; it could also arise from a finite elements analysis (e.g. solution of Stokes' problem) on mesh $\mathcal{T}'$, in which case $||V-\widetilde{V}||_{L^\infty(\mathbb{R}^d)}$ should be controlled by the interpolation error $||V-\pi_{\mathcal{T}'}V||_{L^\infty(\mathbb{R}^d)}$ (generally, it is the case in other norms, thanks to Cea's lemma).
- Theorem 3.1 holds for a generic period of time $[0,T]$, subdivided into smaller intervals of length δt. Going back to the general case of section 3.1, we have to apply it successively on each interval $\left[T^n, T^{n+1}\right]$; the errors in the right-hand side of (9) will then sum up, and it is not difficult to see that the accumulation of spatial errors will become dominant : understandably enough, the intervals on which we backtrack the characteristic curves should be as large as possible, and follow the time step Δt prescribed by the physics of the studied problems.

3.3. **A priori error estimate in terms of Hausdorff distance in the case of level-set functions.** As already pointed out, we are especially interested in the case when the advected scalar function - which we denoted $u(t,.)$ in the previous section - is a level-set function associated to an evolving (regular) domain $\Omega(t)$. The control conveyed by Theorem 3.1 results in that case - at least formally speaking - in an estimate of the Hausdorff distance (whose definition is recalled below) between the continuous evolving interface $\partial\Omega(t)$ and its approximation as the 0-level set of the approximated level-set function.

Definition 3.1. *Let K_1, K_2 two compact subsets of $\mathbb{R}^d$. For any $x \in \mathbb{R}^d$, denote $d(x,K_1) = \inf_{y\in K_1} d(x,y)$ the Euclidean distance from x to K_1 and :*

$$\rho(K_1,K_2) := \sup_{x\in K_1} d(x,K_2).$$

The Hausdorff distance between K_1 and K_2 , denoted by $d^H(K_1,K_2)$, is the nonnegative real number

$$d^H(K_1,K_2) := \max(\rho(K_1,K_2),\rho(K_2,K_1)).$$

With the notations of the previous section, and under the hypothesis of theorem 3.1, suppose moreover that u^{in} is a level-set function associated to a regular bounded domain $\Omega^{in} \subset \mathbb{R}^d$ in the sense (1) holds, and that u^{in} does not admit any critical point in a vicinity of $\partial\Omega^{in}$. According to the material presented in section 2, it follows that for all $t \in [0,T]$, $\Omega(t) := \left\{x \in \mathbb{R}^d \backslash u(t,x) < 0\right\}$ is a bounded regular domain, with smooth boundary $\partial\Omega(t) := \left\{x \in \mathbb{R}^d \backslash u(t,x) = 0\right\}$, and $u(t,.)$ does not admit any critical point within a vicinity of $\partial\Omega(t)$. We also denote, for $n = 0,...,N$, Ω^n and $\partial\Omega^n$ the piecewise affine reconstructions of $\Omega(t^n)$ and $\partial\Omega(t^n)$ obtained as

$$\Omega^n := \left\{x \in \mathbb{R}^d \backslash u^n(x) < 0\right\} \quad ; \quad \partial\Omega^n := \left\{x \in \mathbb{R}^d \backslash u^n(x) = 0\right\}$$

Recall the previous result from [14] :

Lemma 3.3. *Let $u \in \mathcal{C}^1(\mathbb{R}^d)$, without any critical point within a certain tubular neighbourhood W of $\partial\Omega$, so that $\partial\Omega$ is a submanifold of $\mathbb{R}^d$, and Ω is a bounded subdomain of $\mathbb{R}^d$ with $\mathcal{C}^1$ boundary. For any point $x \in W$ we have the estimate :*

$$d(x,\partial\Omega) \leq \frac{\sup\limits_{z\in W} ||\nabla u(z)||}{\inf\limits_{z\in W} ||\nabla u(z)||^2}|u(x)| \tag{19}$$

A *formal* use of this lemma yields :

$$\begin{aligned}
\rho\left(\partial\Omega(T),\partial\Omega^N\right) &\leq \sup_{x\in\partial\Omega(T)} \frac{\sup\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||}{\inf\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||^2}|u^n(x)| \\
&= \sup_{x\in\partial\Omega(T)} \frac{\sup\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||}{\inf\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||^2}|u^N(x)-u(T,x)| \\
&\leq \frac{\sup\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||}{\inf\limits_{K\in\mathcal{T}} ||\nabla u^N|_K(z)||^2}||u^N-u(T,.)||_{L^\infty(\mathbb{R}^d)}
\end{aligned}$$

And now, symmetrically :

$$d^H\left(\partial\Omega(T),\partial\Omega^N\right) \leq \sup\left(\frac{\sup\limits_{z\in\mathbb{R}^d} ||\nabla u(T,z)||}{\inf\limits_{z\in\mathbb{R}^d} ||\nabla u(T,z)||^2}, \frac{\sup\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||}{\inf\limits_{K\in\mathcal{T}} ||\nabla u^N|_K||^2}\right)||u^N-u(T,.)||_{L^\infty(\mathbb{R}^d)}. \tag{20}$$

Therefore, the estimate provided by Theorem 3.1 allows for a control over the discrepancy, measured in terms of Hausdorff distance, between the interface of interest $\partial\Omega(T)$, and its piecewise affine approximation $\partial\Omega^N$, that is actually computed.

4. Mesh adaptation for the advection equation

4.1. **Metric-based mesh adaptation.** The main goal of mesh adaptation is to alter an initial mesh $\mathcal{T}$ in such a way its elements' size and orientation allow to perform the computation of interest with optimal efficiency -i.e. fewer elements, and an enhanced accuracy. Since [36], the idea of *metric-based* mesh adaptation has been increasingly popular : the local desired size, shape and orientation related information at a node x of mesh $\mathcal{T}$ are stored in a Riemannian metric tensor field $M(x)$, which may arise from various possible preoccupations : a posteriori geometric error estimates, analytic error estimates, etc... (see for instance [2], [4], [23]).

Given a metric tensor field $M(x)$, defined at each point $x \in \mathbb{R}^d$, (notice that in practice, $M(x)$ is defined only at the nodes of $\mathcal{T}$ and then interpolated from these values [19]) we consider respectively the *length* $l_M(\gamma)$ of a curve $\gamma : [0,1] \to \mathbb{R}^d$, the *volume* $V_M(K)$ of a simplex K, and the *distance* $d_M(x,y)$ between two points $x, y \in \mathbb{R}^d$ in the Riemannian space $\left(\mathbb{R}^d, M\right)$:

$$l_M(\gamma) = \int_0^1 \sqrt{\langle M(\gamma(t))\gamma'(t),\gamma'(t)\rangle}dt \ , \ V_M(K) = \int_K \sqrt{det(M(x))}dx,$$
$$d_M(x,y) = \inf_{\substack{\gamma\in\mathcal{C}([0,1],\mathbb{R}^d)\\ \gamma(0)=x,\gamma(1)=y}} l_M(\gamma).$$

We aim at modifying mesh $\mathcal{T}$ so as to make it *quasi-unit* with respect to the metric $M(x)$, that is to say all its simplices K have edges lengths lying in $\left[\frac{1}{\sqrt{2}}, \sqrt{2}\right]$. What is more, we expect the *anisotropic quality measure*:

$$\mathcal{Q}_M(K) := \alpha_d \frac{V_M(K)^2}{\left(\sum_{i=1}^{na} l_M(e_i)^2\right)^d}$$

of all elements of the mesh (where $na = d(d+1)/2$ is the number of edges of a d-dimensional simplex, e_i are the edges of K and α_d is a normalization factor) to be as close to 1 as possible. The underlying geometrical

interpretation is that for any given node x_0 of a quasi-unit mesh $\mathcal{T}$ with respect to $M(x)$, every element $K \in \mathcal{T}$ which shares x_0 as a vertex fits 'at best' in the unit pseudo-ellipsoid

$$\Phi_M(x_0) = \left\{ x \in \mathbb{R}^d \backslash d_M(x, x_0) = 1 \right\},$$

which reduces to a true ellipsoid when M is constant over $\mathbb{R}^d$. In this latter case, the eigenvectors $e_1, ..., e_d$ of M give the directions of the principal axis of this ellipsoid, while the associated eigenvalues $\lambda_1, ..., \lambda_d$ are linked to the principal radii (or characteristic lengths) $h_1, ..., h_d$ in direction $e_1, ..., e_d$ by : $h_i = \frac{1}{\sqrt{\lambda_i}}$, $i = 1, ..., d$.

For several applications (see the next sections), it may be desirable to adapt a mesh *at the same time* to several a priori independent information, supplied by two (or more) metric tensor fields M_1, M_2. This is classically achieved resorting to a so-called *metric intersection* procedure : operating on the simultaneous reductions of $M_1(x)$ and $M_2(x)$ at any point $x \in \mathbb{R}^d$,

$$M_1(x) = {}^tP(x) \begin{pmatrix} \lambda_1(x) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d(x) \end{pmatrix} P(x), M_2 = {}^tP(x) \begin{pmatrix} \mu_1(x) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mu_d(x) \end{pmatrix} P(x),$$

where P is an invertible matrix, $\lambda_1, ..., \lambda_d, \mu_1, ..., \mu_d > 0$, the *intersected metric*

$$M_1 \cap M_2(x) := {}^tP(x) \begin{pmatrix} sup\left(\lambda_1(x), \mu_1(x)\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & sup\left(\lambda_d(x), \mu_d(x)\right) \end{pmatrix} P(x)$$

carries both sets of information in the sense that its unit pseudo-ellipsoid $\Phi_{M_1 \cap M_2}(x)$ is a maximal pseudo-ellipsoid enclosed in both $\Phi_{M_1}(x)$ and $\Phi_{M_2}(x)$, at least if M_1 and M_2 are constant over $\mathbb{R}^d$ (see [19] for details).

Several techniques have been thought up for generating anisotropic meshes according to a metric tensor field, that can be roughly classified into two categories. On the one hand, global methods, such as Delaunay-based methods and advancing-front methods, perform the same kind of operations as in the classical case with adapted notions of length and volume. On the other hand, local mesh modification methods [20] - to which belongs the approach followed in this paper - start from an existing non-adapted mesh and adapt it so that it fulfills at best the above conditions.

4.2. **The proposed adaptation method.** In this section, we use back the framework of sections 3.2 and 3.3 and intend to use the previous error estimates (9) and (20) to infer a mesh adaptation method which allows for a good discrete approximation $\partial\Omega_T$ of $\partial\Omega(T)$. For the sake of clarity we shall now denote u_T the approximation of $u(T, .)$ obtained by the algorithm of section 3.1 and

$$\Omega_T = \left\{ x \in \mathbb{R}^d \backslash u_T(x) < 0 \right\} \quad , \quad \partial\Omega_T = \left\{ x \in \mathbb{R}^d \backslash u_T(x) = 0 \right\}$$

the associated polyhedral domain and surface. From formula (9), the control we get over $||u(T, .) - u_T||_{L^\infty(\mathbb{R}^d)}$ -or indifferently $d^H\left(\partial\Omega(T), \partial\Omega_T\right)$ - consists of three independent contributions :

(1) The first one
$$\left(||u(T, .) - \pi_{\mathcal{T}} u(T, .)||_{L^\infty(\mathbb{R}^d)} + ||u^{in} - \pi_{\mathcal{T}} u^{in}||_{L^\infty(\mathbb{R}^d)} \right)$$
is only linked to the way mesh $\mathcal{T}$ is adapted to the interpolation of functions $u(T, .)$ and u^{in}. The proposed adaptation method focuses on dampening this part of the error.

(2) The second one
$$\frac{k'}{k} C T e^{C\delta t} \delta t$$
is solely related to the time discretization of interval $[0, T]$ with the substep δt, and for this contribution to be decreased, substep δt has to be decreased.

(3) The last part
$$\frac{k'}{k} (e^{kT} - 1) ||V - \widetilde{V}||_{L^\infty(\mathbb{R}^d)}$$

is connected to the quality of the approximation of velocity field V. If this vector field is to be obtained by means of a finite element computation on mesh $\mathcal{T}'$, this has to do with the quality of $\mathcal{T}'$ as regards this computation, and we do not intend to discuss these aspects. As said previously, in all our numerical examples, we only considered analytically prescribed velocity fields, and this contribution actually does not appear.

This leaves us with the objective of adapting mesh $\mathcal{T}$ in such a way *both* interpolation errors $||u(T,.) - \pi_{\mathcal{T}} u(T,.)||_{L^\infty(\mathbb{R}^d)}$ and $||u^{in} - \pi_{\mathcal{T}} u^{in}||_{L^\infty(\mathbb{R}^d)}$ are made small. In other words, in view of the estimates in section 3.3, this means that mesh $\mathcal{T}$ allows for a good approximation of the 0-level sets of both u^{in} and $u(T,.)$ by the 0-level set of their respective $\mathbb{P}^1$-interpolate $\pi_{\mathcal{T}} u^{in}$, $\pi_{\mathcal{T}} u(T,.)$. This is actually very intuitive : neglecting the errors on the knowledge of the velocity field, the accuracy of the process is controlled by how well mesh $\mathcal{T}$ allows for a good knowledge of the initial surface, and how well it fits a description of the final one.

To achieve such a mesh, we recall the classical L^∞ error estimate for the Lagrange finite element $\mathbb{P}^1$-interpolation error of a function u of class $\mathcal{C}^2$ [15] :

Theorem 4.1. *Let $\mathcal{T}$ a simplicial mesh of $\mathbb{R}^d$ (or a polyhedral subset of it) and u a $\mathcal{C}^2$ function on $\mathbb{R}^d$. Then for every simplex $K \in \mathcal{T}$,*

$$||u - \pi_{\mathcal{T}} u||_{L^\infty(K)} \leq \frac{1}{2}\left(\frac{d}{d+1}\right)^2 \max_{x\in K} \max_{y,z\in K} \langle |\mathcal{H}(u)|(x)yz, yz\rangle$$

where $\mathcal{H}(u)$ is the Hessian matrix of u and, for a symmetric matrix $S \in \mathcal{S}_d(\mathbb{R})$ which admits the following diagonal shape in orthonormal basis $S = P\, diag(\{\lambda_i\}_{1\leq i\leq d})\, {}^tP$, we denote $|S| := P\, diag(\{|\lambda_i|\}_{1\leq i\leq d})\, {}^tP$.

According to [2], this theorem expresses the idea that a mesh $\mathcal{T}$ suitable for the Lagrange $\mathbb{P}^1$-interpolation of a smooth function u -i.e. such that $||u - \pi_{\mathcal{T}} u||_{L^\infty(K)} \leq \epsilon$ for a given tolerance $\epsilon > 0$ and every simplex $K \in \mathcal{T}$ - can be roughly obtained as a quasi-unit mesh for the metric M_u defined at each node x of $\mathcal{T}$ by :

$$M_u(x) = {}^tP(x) \begin{pmatrix} \widetilde{\lambda_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \widetilde{\lambda_d} \end{pmatrix} P(x) \tag{21}$$

where

$$\widetilde{\lambda_i} = \min\left(\max\left(\frac{c|\lambda_i|}{\epsilon}, \frac{1}{{h_{max}}^2}\right), \frac{1}{{h_{min}}^2}\right), \quad \widetilde{|\mathcal{H}(u)|}(x) = {}^tP(x) \begin{pmatrix} |\lambda_1| & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & |\lambda_d| \end{pmatrix} P(x)$$

being an approximation of the Hessian of u around node x, written here in diagonal form in an orthonormal basis, h_{min} (resp. h_{max}) being the smallest (resp. largest) size allowed for an element in any direction, and c being the above constant.

Remark Actually, we do not need to adapt the whole mesh $\mathcal{T}$ with respect to metric M_u : as we are mostly interested in an accurate approximation of the zero level set of the considered functions by the piecewise affine zero level set of their Lagrange $\mathbb{P}^1$-interpolate, we only need to have a quasi-unit mesh $\mathcal{T}$ according to M_u in a vicinity of the 0 level set of u. See section 5.1 for a further discussion on this point.

In our applications, we are interested in modifying mesh $\mathcal{T}$ so that it becomes adapted to *both* Lagrange $\mathbb{P}^1$ interpolation of functions u^{in} and $u(T,.)$. Such a mesh is built as a quasi-unit mesh according to the intersection $M_{u^{in}} \cap M_{u(T,.)}$ of metrics $M_{u^{in}}$ and $M_{u(T,.)}$, respectively adapted to u^{in} and $u(T,.)$ (see figure 1 for an example).

In practice, our adaptation procedure is *iterative* : it starts with an initial function u^{in}, on an adapted mesh $\mathcal{T}^{in}$, adapted to metric $M_{u^{in}}$, and in order to compute an approximation u_T of $u(T,.)$, the advection equation is solved up to m times over the period $[0,T]$: the first $m-1$ times are 'virtual', and aimed at

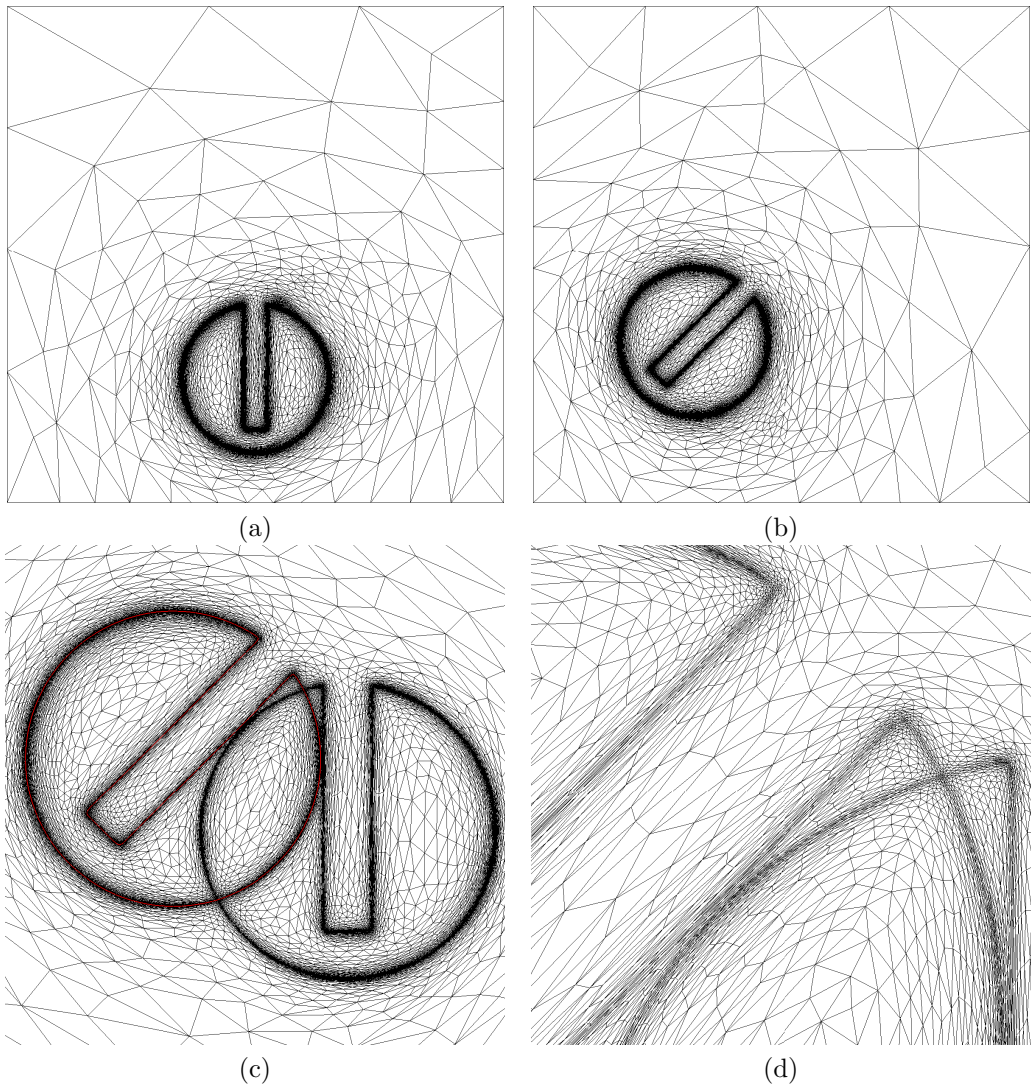

FIGURE 1. *Rotation of Zalesak's slotted disk of angle $\frac{\pi}{4}$: (a) adapted mesh at time $t = \pi$, (b) adapted mesh at time $\frac{5\pi}{8}$, (c) mesh adapted to both interfaces $\partial\Omega^{in}$ and $\partial\Omega(T)$ (displayed in red), (d) zoom on the mesh.*

getting an increasingly accurate approximation $\widetilde{u_T^k}$ $(k = 0, ...m-1)$ of $u(T, .)$, as well as an increasingly well-adapted mesh $\mathcal{T}^k$ to the intersected metric $M_{u^{in}} \cap M_{u(T,.)}$. Eventually, the m-th resolution of the advection equation is carried out on a well-adapted mesh $\mathcal{T}^{M-1}$ and yields a close approximation $u_T = \widetilde{u_T^m}$ of $u(T, .)$. Such an iterative procedure is absolutely crucial, because the time period $[0, T]$ can be very large. Thus, the sought $\partial\Omega(T)$ is likely to be located very far from $\partial\Omega^{in}$, i.e. in an area where the initial mesh is very coarse.

All things considered, the proposed adaptation method for advection equation is summed up in algorithm (1), for a generic time interval $[0, T]$. Of course, in the general case, when several such time periods $\left[T^n, T^{n+1}\right]$ follow one another, this process has to be applied successively to each one of them.

5. Additional numerical features

5.1. **The need for mesh gradation control.** Considering a function u, we are mainly interested in its 0 level set. Thus, we only need to adapt our computational mesh $\mathcal{T}$ with respect to the metric M_u defined in

Algorithm 1 Adaptation method for advection equation over $[0,T]$.

Start with an approximation u_0 (e.g. $\mathbb{P}^1$-interpolate) of function u^{in} on mesh $\mathcal{T}$.
for $k = 0, ..., m-1$ **do**
  if $k = 0$ **then**
    Set $\widetilde{\mathcal{T}^0} = \mathcal{T}$ and $\widetilde{u^{in,0}} = u_0$.
  end if
  Solve the advection equation with velocity field V and initial state $\widetilde{u^{in,k}}$, over $[0,T]$, on $\widetilde{\mathcal{T}^k}$. $\widetilde{u_T^k}$
  Compute the intersected metric $M^k := M_{\widetilde{u_0^k}} \cap M_{\widetilde{u_T^k}}$
  Adapt $\widetilde{\mathcal{T}^k}$ with respect to M^k. $\widetilde{\mathcal{T}^{k+1}}$
  if $k+1 < m$ **then**
    Project u_0 on mesh $\widetilde{\mathcal{T}^{k+1}}$. $\widetilde{u^{in,k+1}}$
  else
    Project $\widetilde{u_T^k}$ on $\widetilde{\mathcal{T}^{k+1}}$. $(\widetilde{\mathcal{T}^m}, \widetilde{u_T^m})$
    Adapt $\widetilde{\mathcal{T}^m}$ with respect to $\widetilde{u_T^m}$, and project this function on the final mesh. $(\mathcal{T}_T, u_T)$
  end if
end for
return $(\mathcal{T}_T, u_T)$

the previous section on a neighborhood of $\partial\Omega := \left\{x \in \mathbb{R}^d \backslash u(x) = 0\right\}$. As is classical in mesh adaptation, we may ask $\mathcal{T}$ to show very large, isotropic elements 'far' from this interface, in such a way $\mathcal{T}$ consists of very few elements, with optimal size and orientation. However, doing so, variations in the size and orientation prescriptions are bound to be very sharp, resulting in a shock of features which can yield severe instabilities during the mesh adaptation process (see figure 5 for an example). To get past this difficulty, we impose a control over mesh gradation near the interface [10]. One possible way to do this consists in, roughly speaking, bounding the allowed ratio between lengths $l_M(e_1)$ and $l_M(e_2)$ of any two edges e_1, e_2 belonging to a common simplex K by a constant value r (in the examples of section 6, we used $r = 4$), i.e.

$$\frac{1}{r} \leq \frac{l_M(e_1)}{l_M(e_2)} \leq r.$$

5.2. **Importance of redistancing.** In the context of level set methods, it has been observed that too steep or too loose variations in the level sets of the function $u(t,.)$ under evolution may jeopardize the accuracy of the computation. To overcome this feature, since [13], a great attention has been paid to maintaining or restoring $u(t,.)$ as the signed distance function to its 0 level set $\partial\Omega(t)$ at least near $\partial\Omega(t)$ (so that $||\nabla u(t,.)|| = 1$ in a vicinity of $\partial\Omega(t)$), even though, doing so, the handled interface may inevitably end up perturbed : see for instance [35], [5] for mass-preserving approaches, or [7] for a smoothing redistancing process. Here we apply the previous study of [14], merely replacing the computed approximation u_T of $u(T,.)$ by the signed distance function $\widetilde{u_T}$ to $\partial\Omega_T$. Given a small time step dt, $\widetilde{u_T}$ is computed as the steady state of the sequence of $\mathbb{P}^1$-finite element functions u^n defined in algorithm 2, which is based on the properties of the *unsteady Eikonal equation*.

As discussed in [14], time step dt must be chosen small enough so that the updating of $u^n(x)$ with the formulae of algorithm 2 does not require values of u^{n-1} lying 'too far' on the other side of the boundary. For this reason, dt should be taken of the order of the local mesh size near the interface $\partial\Omega_T$. However, this time step can be steadily increased, as values of the sequence u^n converge near the interface. What is more, we actually need $\widetilde{u_T}$ to enjoy the distance property only in a neighborhood of $\partial\Omega_T$. For this reason, in practice, we limit ourselves to performing a fixed number of the iterations of this algorithm.

Algorithm 2 Redistancing of the level-set function

Initialize function u^0 with :

$$\begin{cases} u^0(x) & = & \textit{approximation of the signed distance function to } \Omega_T \textit{ if } x \\ & & \textit{belongs to a simplex of } \mathcal{T} \textit{ intersecting } \partial\Omega_T \\ u^0(x) & = & \pm u_{MAX} \textit{ otherwise} \end{cases}$$

for $n = 1, ...$ until convergence **do**
  $u^n(x) = u^{n-1}(x)$ for each node x of $\mathcal{T}$
  for each node x of $\mathcal{T}$ *which does not belong to a simplex intersecting* $\partial\Omega_T$ **do**
    if $x \notin \Omega$ **then**

$$u^n(x) = \min\left(u^{n-1}(x), \min_{K\in\mathcal{T}\ s.t.\ x \text{ is a node of } K} u^{n-1}\left(x - \frac{\nabla(u^{n-1}|_K)}{||\nabla(u^{n-1}|_K)||}dt\right) + dt\right),$$

    else

$$u^n(x) = \max\left(u^{n-1}(x), \max_{K\in\mathcal{T}\ s.t.\ x \text{ is a node of } K} u^{n-1}\left(x + \frac{\nabla(u^{n-1}|_K)}{||\nabla(u^{n-1}|_K)||}dt\right) - dt\right).$$

    end if
  end for
end for
return u^n

The study carried out in [14] showed that doing so yields formally an error in terms of Hausdorff distance between $\partial\Omega_T$ and the new interface $\widetilde{\partial\Omega_T}$:

$$d^H\left(\partial\Omega_T, \widetilde{\partial\Omega_T}\right) \leq \sup\left(\frac{\sup_{K\in\mathcal{T}} ||\nabla u_T|_K||}{\inf_{K\in\mathcal{T}} ||\nabla u_T|_K||^2}, \frac{\sup_{K\in\mathcal{T}} ||\nabla \widetilde{u_T}|_K||}{\inf_{K\in\mathcal{T}} ||\nabla \widetilde{u_T}|_K||^2}\right) \max_{K\in\mathcal{T}} \max_{y,z\in K} \langle |\mathcal{H}(\widetilde{u_T})| yz, yz\rangle. \tag{22}$$

Consequently, given mesh $\mathcal{T}$ is adapted to $|\mathcal{H}(u_T)| \approx |\mathcal{H}(\widetilde{u_T})|$ in the sense of section 4.2, this error is actually very small, controlled by the fixed precision parameter ϵ.

Note that in the proposed context, the redistancing process is twice as important as in the case of level set methods performed on a fixed mesh Indeed, the narrow band on which we impose our mesh to be adapted with respect to a metric M_u as in section 4.2 is numerically identified according to the values of u. Hence, very stretched level sets of u would also cause mesh adaptation to be performed on an ill-identified narrow band around $\partial\Omega$.

6. Numerical examples

In this section, we present several numerical examples in two or three dimensions in order to assess the validity of the proposed adaptation method for the level set advection equation. Each one of them consists in letting evolve an initial interface $\partial\Omega^{in}$ submitted to a more or less deforming velocity field V from time 0 to a final time T, cooked up in such a way the final surface $\partial\Omega(T)$ coincides with $\partial\Omega^{in}$.

As presented in the analysis of section 3.2, all functions are approximated by $\mathbb{P}^1$ Lagrange finite element functions, and we discretized the ODE (5) with a 4-th order Runge-Kutta method. As hinted at previously, the use of higher order spatial approximation (e.g. resorting to $\mathbb{P}^k$ Lagrange finite element functions) would improve the method. It is also worth mentionning that we used the aforementioned redistancing procedure at each step, even for the cases (e.g. rigid-body motions) that do not theoretically bring about any distortion of the level sets of the evolving function.

The accuracy of the computation is evaluated in terms of the Hausdorff distance between the numerical initial interface associated to $\partial\Omega^{in}$ and the computed final interface $\partial\Omega_T$, which is computed with a brute-force approach (in $2d$ only). We believe this is the relevant way to measure the error entailed by the method, inasmuch as it is the quantity we mean to control with our mesh adaptation procedure, through estimate (20). We are however well aware this is not so classical an error measure, and propose also more standard

error measures. Although no particular attention has been paid to mass conservation during this work, we also display the loss of mass between initial and final step for the sake of completeness.

All our $2d$ examples were run on a MacBook Pro, 2.66 Ghz, (4 Go), and our $3d$ examples on an OPTERON 2.1 Ghz.

Rotation of Zalesak's slotted disk. As initially proposed in [38], consider a unit square as a computational domain, in which lies a disk of radius 0.15 centered at $(0.5, 0.75)$, with a slot of length 0.25, submitted to a uniform rotation of center $(0.5, 0.5)$, corresponding to an evolution under velocity field :

$$V(t,x,y) = \begin{pmatrix} -(y-0.5) \\ (x-0.5) \end{pmatrix}.$$

between $0 \leq t \leq T = 2\pi$, so that the final interface should theoretically overlay the initial one. This test-case classically allows for an assessment of the well-preservation of the sharp features of an interface through an evolution. We subdivide the time interval into 8 time periods on which algorithm 1 is successively applied. We perform two computations with different parameters as regards the mesh size prescription so as to test the scaling of the method. Figure 2 displays a comparison between both interfaces. In table 1, we provide the associated parameters (precision parameter ϵ, minimum size parameter h_{min}), along with their translation in terms of mesh size : maximum number np of points of a mesh adapted to a single interface (the number of points of a mesh adapted to both interfaces being approximately equal to twice this number), as well as the error estimates of interest : we compute the Hausdorff distance between the initial interface, as well as two error measures that are more classical in the literature [1] [12] [27] [24] : the measure of the symmetric difference between the initial and final domains Ω^{in} and Ω_T, (or sometimes referred to as L^1-*error measure*)

$$E_{sd}(\Omega^{in}, \Omega_T) := |\, (\Omega_{in} \cup \Omega_T) \setminus (\Omega_{in} \cap \Omega_T)\,|$$

and the L^∞-*error measure* between both numerically obtained corresponding level-set functions u^{in} and u_T :

$$E_\infty(u^{in}, u_T) := \sup_{K \in \mathcal{K}} ||u^{in} - u_T||_{L^\infty(K)},$$

$\mathcal{K} \subset \mathcal{T}$ being the set of simplices crossed by the 0-level set of either u^{in} or u_T. The whole computation of the first test takes roughly 5 minutes, while it takes about 8 minutes for the second.

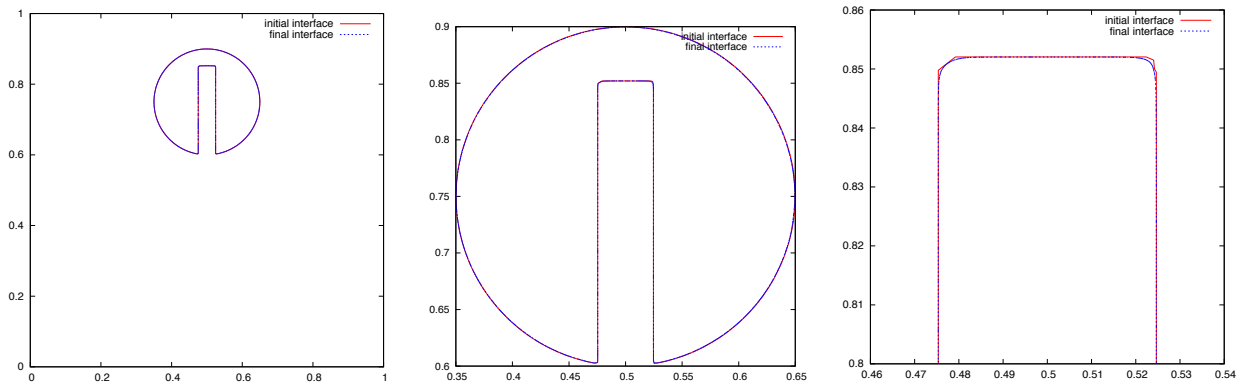

FIGURE 2. *(left) Superposition of the slotted disk after one complete revolution (blue line) over the initial one (red line), and zoom on the surfaces (middle and right).*

We notice that a very nice accuracy can be obtained with a very small number of points. The sharp features of the slotted disk are well preserved, and the final interface presents no oscillation whatsoever. However, it seems difficult to say much more than that, especially when it comes to comparison with other methods, or order computation of the process : we use a numerical scheme for the advection equation that is admittedly

	ϵ	h_{min}	np	Volume loss (% initial vol.)	$d^H(\partial\Omega^{in}, \partial\Omega_T)$	$E_{sd}(\Omega^{in}, \Omega_T)$	$E_\infty(u^{in}, u_T)$
Test (1)	$1e^{-3}$	$1e^{-3}$	5699	-0.889	$2.08e^{-3}$	$3.36e^{-3}$	$2.60e^{-3}$
Test (2)	$1e^{-4}$	$1e^{-4}$	14477	-0.299	$8.61e^{-4}$	$2.18e^{-3}$	$1.34e^{-3}$

TABLE 1. Details on the two-dimensional test of Zalesak's slotted disk.

low-order, and the accuracy of the process stems from the mesh adaptation process. The number of points (or triangles) of the meshes at hand can vary considerably from one iteration to the other depending on the 'wildness' of the interface to be captured ; they also depend in an unclear way of the precision parameters ϵ and h_{min}.

Time-reversed vortex flow. The second test-case, proposed in [25], is a very good opportunity to investigate the behaviour of our method when dealing with velocity fields entailing serious deformations of the initial shape. In a unit square domain, consider a disk of radius 0.15, centered at $(0.5, 0.75)$, evolving according to the velocity field

$$V(t,x,y) = \begin{pmatrix} -sin^2(\pi x)sin(2\pi y)cos(\frac{\pi t}{T}) \\ sin^2(\pi y)sin(2\pi x)cos(\frac{\pi t}{T}) \end{pmatrix}.$$

for $0 \leq t \leq T = 8$. Because of the vorticity of the velocity field, some parts of the initial disk are compressed, while some others become very stretched. What is more, the shape tends to become a more and more thin filament, and very small (or, in our case, very elongated) elements have to be put so that mesh resolution allows for a description of these parts. Because of the modulation in time, the shape reaches its most distorted state at time $t = T/2$, and has returned to its initial state at time T. We performed 20 intermediate time steps on this test-case, and the whole computation takes about 40 minutes. Figure 3 shows four steps of the computation. Comparison between the initial and final state is reported in figure 4, and subsequent details and error measures on this test-case can be found in table 2.

In order to emphasize the importance of mesh gradation, as discussed in section 5, we report in figure 5 the comparison between the first step of the presented computation, and the first step of the same computation, performed with the same parameters, but without any mesh gradation. Note that the 0 level set of the evolving function has not been represented in the latter case for it has been utterly 'lost' ! This shows that, even with a correct minimum size allowed, and with relevant precision parameters, the mesh adaptation process needs such a gradation in the mesh, for the sake of robustness : indeed, at the beginning of each iteration, the 0-level set of the evolving function is advected towards an area where the computational mesh is likely to be coarse, then refined with the m internal iterations, as expressed in algorithm 1. If no gradation in the mesh is enforced, this first 'virtual' advection may be too rough for capturing small details, that will be missed by the subsequent iterations.

ϵ	h_{min}	np	Volume loss (% initial vol.)	$d^H(\partial\Omega^{in}, \partial\Omega_T)$	$E_{sd}(\Omega^{in}, \Omega_T)$	$E_\infty(u^{in}, u_T)$
$1e^{-4}$	$3e^{-4}$	34862	-0.380	$1.81e^{-3}$	$8.56e^{-4}$	$1.83e^{-3}$

TABLE 2. Details on the two-dimensional time-reversed vortex flow test-case.

Remark We hinted at the fact that, during a single iteration of the process, the 0 level set of the considered scalar function may be advected from an area where the mesh is suitably refined, towards an area where it is dramatically under-sampled (this is particularly likely to happen if the physical time step Δt is chosen very large). One could wonder whether it could prove beneficial to make a first refinement of the 'landing area',

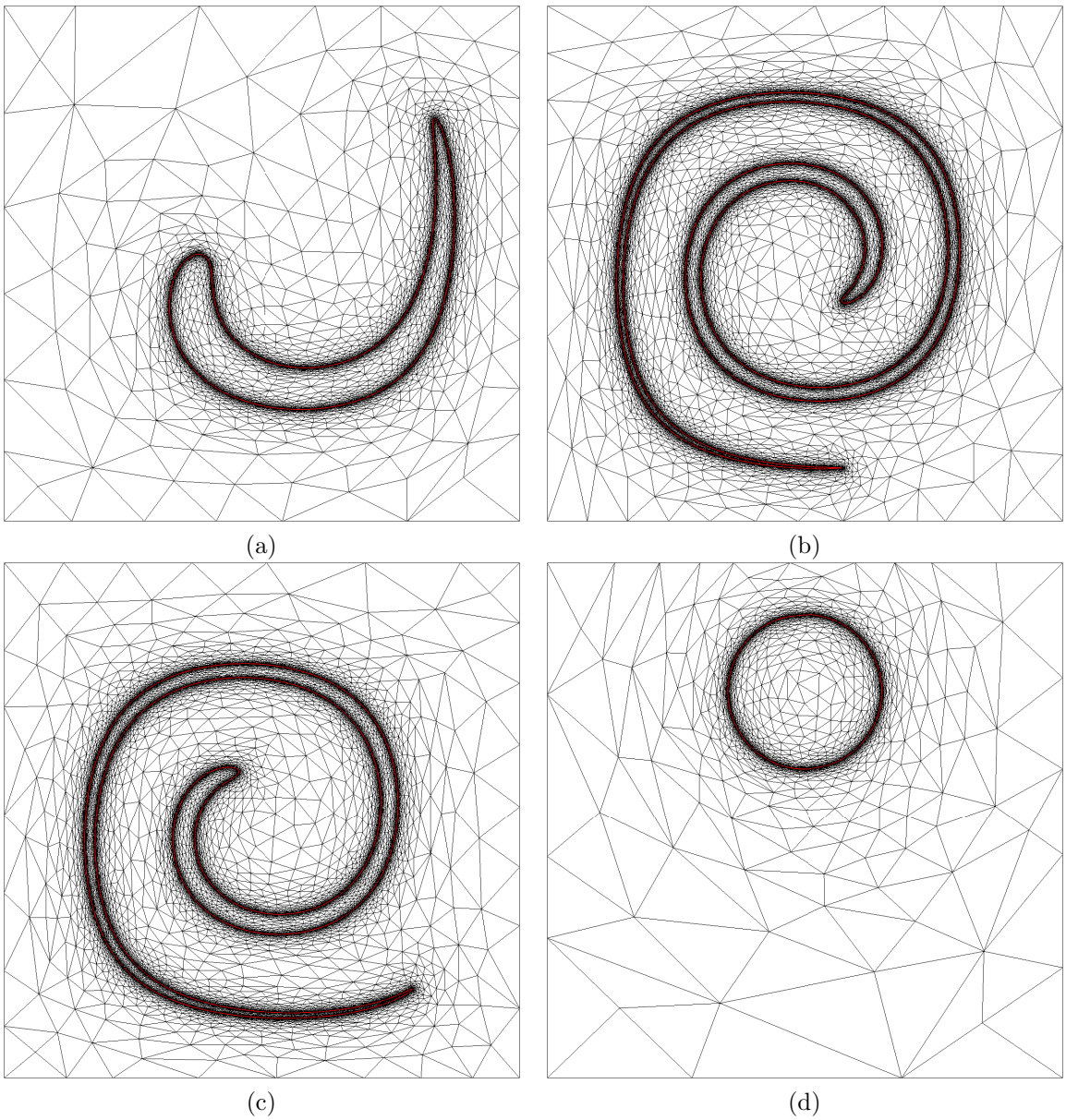

FIGURE 3. *Three steps of the computation for the time-reversed vortex flow, (a-b)* $t = 0.8$*, (c-d)* $t = 4$*, (e-f)* $t = 5.6$ *together with the corresponding isovalues*

which could be achieved by applying the numerical scheme for advection to the size prescription (which is computationally unexpensive), before turning to the first internal iteration, just so as to get a well-sampled landing area. This merely amounts to roughly adding some degrees of freedom where we know the next 0 level set of interest will be located. Actually, this process is rather easy to implement numerically, at least when it comes to transporting the sole size prescription ; things grow more tedious if we are to advect both size and orientation prescriptions. However, this yields disappointing numerical results : almost no improvement on the method has been observed when resorting to this technique.

Deformation test flow. An even more serious test case when it comes to shape distortion has been proposed by [33]. In a unit square domain, a disk of radius 0.15, centered at $(0.5, 0.75)$ is evolved according

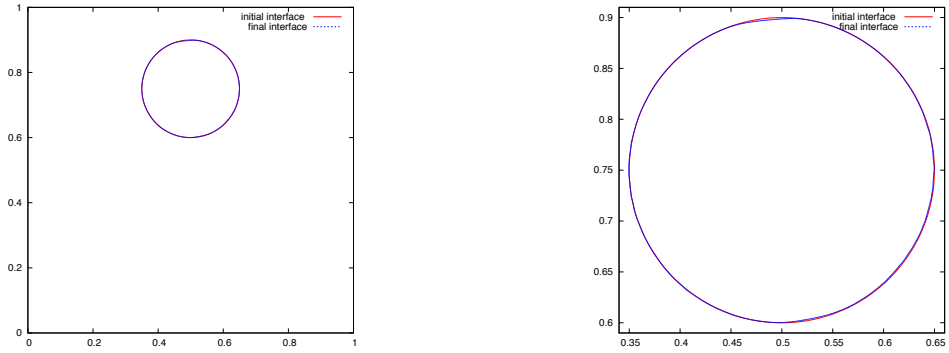

FIGURE 4. *Superposition of the final disk (blue line) over the initial one (red line), and zoom on the comparison (right)*

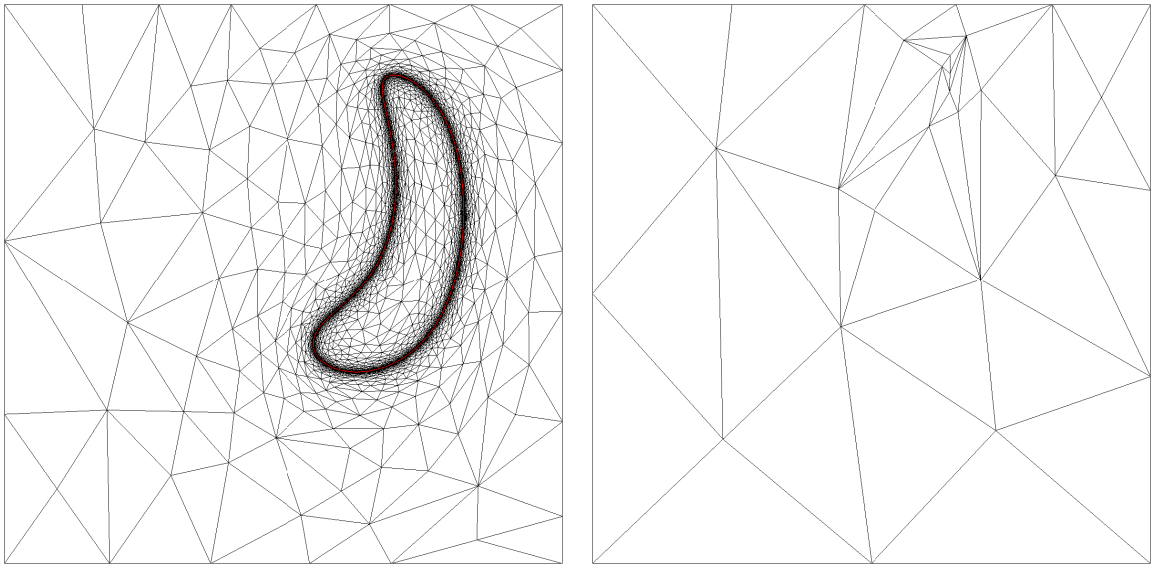

FIGURE 5. *Evolving surface at time $t = 0.4$ with (left) and without (right) mesh gradation control.*

to the following velocity field :

$$V(t,x,y) = \begin{pmatrix} sin(4\pi(x+\frac{1}{2}))sin(4\pi(y+\frac{1}{2}))cos(\frac{\pi t}{T}) \\ cos(4\pi(x+\frac{1}{2}))cos(4\pi(y+\frac{1}{2}))cos(\frac{\pi t}{T}) \end{pmatrix}.$$

for $0 \leq t \leq T = 3$. Periodicity of the domain with respect to the top and bottom sides is enforced. Roughly speaking, this velocity field makes the domain composed of 16 vortices, and several parts of the disk are dragged by different ones, literally tearing the shape into pieces. See figure 6 for some illustrations of this test case, and figure 7 for a comparison between the initial and final interfaces. The whole computation takes around 70 minutes, and details as well as error measures regarding this test-case are to be found in table 3.

Rotation of Zalesak's sphere. Very similarly to the first two-dimensional example comes the rotation of Zalesak's slotted sphere in 3 dimensions. In a unit cube, a sphere of radius 0.15 with a slot of width 0.15 is initially centered at $(0.5, 0.5, 0.25)$, and undergoes a uniform rotation with respect to the x-axis,

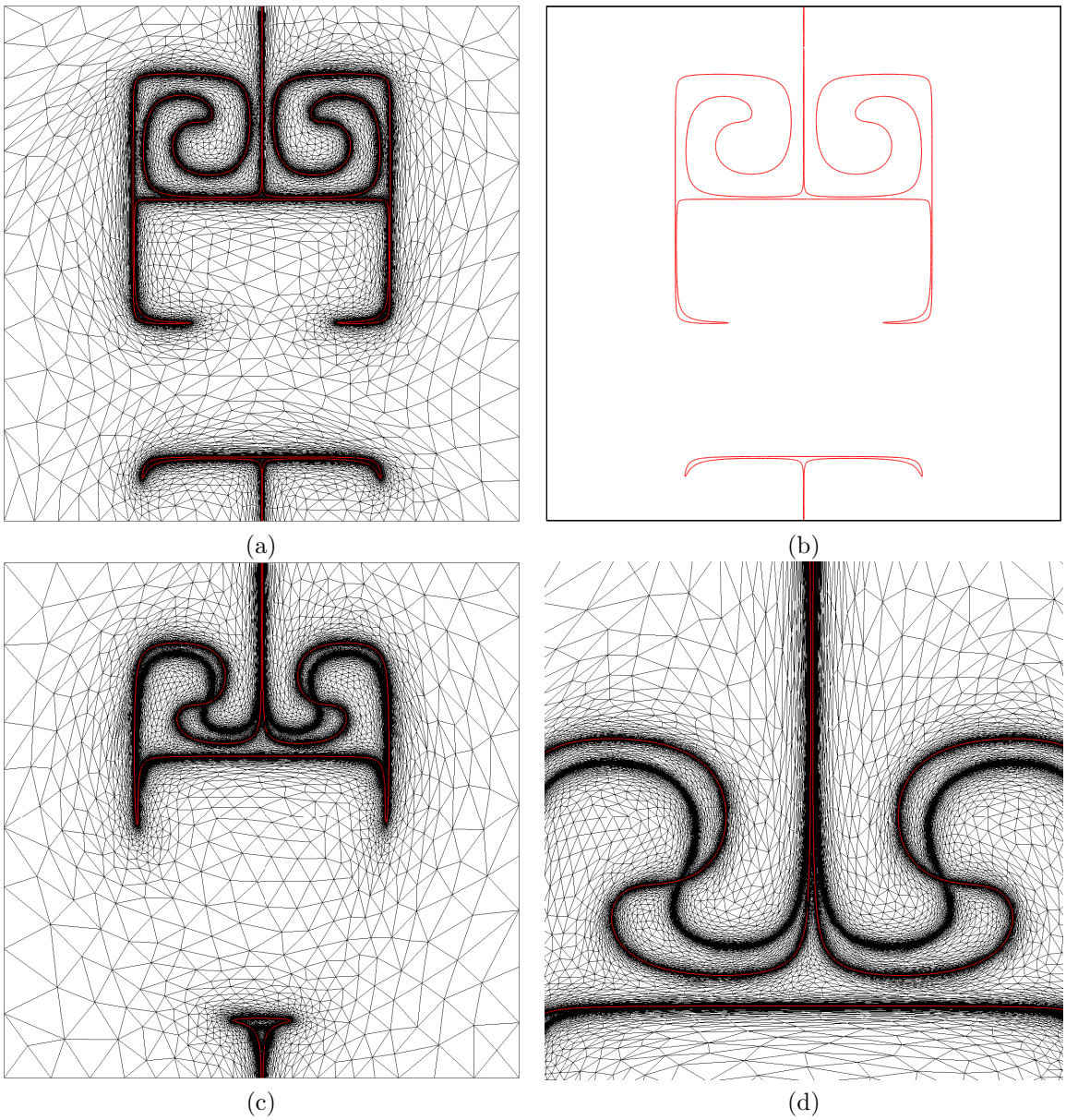

FIGURE 6. *(a) Maximum elongation step ($t = T/2$), (b) corresponding 0-level set, (c) intersected mesh, adapted to both steps $t = 0.9$ and $t = 1.2$, (d) zoom on the intersected mesh (the surface associated to $t = 1.2$ is displayed in red).*

ϵ	h_{min}	np	Volume loss (% initial vol.)	$d^H(\partial\Omega^{in}, \partial\Omega_T)$	$E_{sd}(\Omega^{in}, \Omega_T)$	$E_\infty(u^{in}, u_T)$
$1e^{-3}$	$5e^{-5}$	56536	-0.097	$4.81e^{-3}$	$6.77e^{-3}$	$4.09e^{-3}$

TABLE 3. Details on the two-dimensional deformation flow test-case.

corresponding to a velocity field :

$$V(t,x,y,z) = \begin{pmatrix} 0 \\ -(z-0.5) \\ (y-0.5) \end{pmatrix}.$$

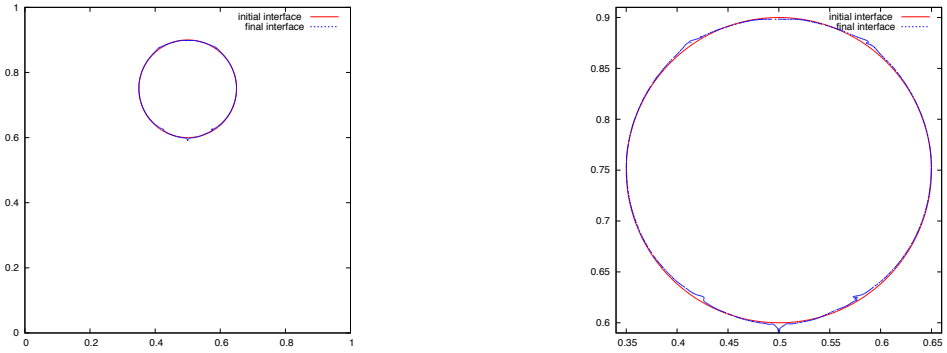

FIGURE 7. *Superposition of the final disk (blue line) over the initial one (red line), and zoom on the comparison (right)*

over $0 \leq t \leq 2\pi$. This sphere shows both ridges and triple points, that are naturally the most difficult features to preserve throughout the advection process.

We split the time interval into 8 subperiods, and work with parameters $\epsilon = 0.005$, $h_{min} = 0.005$, in such a way each computational mesh has about $200,000$ points ($\approx 1,200,000$ tetrahedra).The sequence of obtained surfaces is displayed on figure 8, while a zoom on both initial and final states is to be found on figure 9, and several cuts into an 'intersected mesh' are to be found on figure 10. The whole computation process takes about 100 minutes. Comparison of the initial and final interfaces demonstrate a good accuracy of the method, even though, of course, the ridges and triple points of the surface have been a little smeared.

Three-dimensional deformation test case. Eventually, we turn to yet another test case proposed in [25]. In a unit cube, a sphere of radius 0.15, centered at $(0.35, 0.35, 0.35)$ is made evolved according to the following velocity field :

$$V(t,x,y,z) = \begin{pmatrix} 2sin^2(\pi x)sin(2\pi y)sin(2\pi z)cos(\frac{\pi t}{T}) \\ -sin(2\pi x)sin^2(\pi y)sin(2\pi z)cos(\frac{\pi t}{T}) \\ -sin(2\pi x)sin(2\pi y)sin^2(\pi z)cos(\frac{\pi t}{T}) \end{pmatrix}.$$

for $0 \leq t \leq T = 3$. We split the time interval into 10 subperiods, and the results are reported in figure 11. Figure 12 displays two cuts in the most stretched interface of the evolution (the one at time $t = 1.5$), while figure 13 displays two cuts in two different adapted meshes (one is anisotropic, the other is isotropic) to the latter interface. The results presented in figure 11, which are the best obtained among different tests carried out with different parameters, are those corresponding to a computation held with isotropic adaptation with a minimum size parameter $h_{min} = 0.002$ (which amounts to anisotropic adaptation with very small precision parameters). The largest mesh of the computation is worth $3,763,497$ vertices, and the whole computation took about 21 hours.

Taking a close look at the displayed sequence, one realizes that the final interface is not exactly as smooth as the initial one, notably near its horizontal diameter ; actually, this area corresponds to the most stretched zone at the maximum elongation time $t = T/2$, and is the most difficult to track accurately (see the results in e.g. [24] or [5] for similar behavior) ; of course, this effect vanishes with enhanced resolution.

FIGURE 8. *Rotation of Zalesak's sphere : sequence of computed surfaces.*

The proposed method for solving the level set advection equation brings into play various tools, and it is interesting to wonder which parts exactly take most of the computational expense. For each example, we reported in table 4, the details of the longest iteration of the process, that is, the total time Δt_{it} of the iteration, and the percentage of it which has been spent in each main step of the process (note that the interpolation steps have been neglected). We notice that, understandably enough, the remeshing procedure is by far the most costly in every case, which is quite understandable since it is the point in where lies the main complexity of the method.

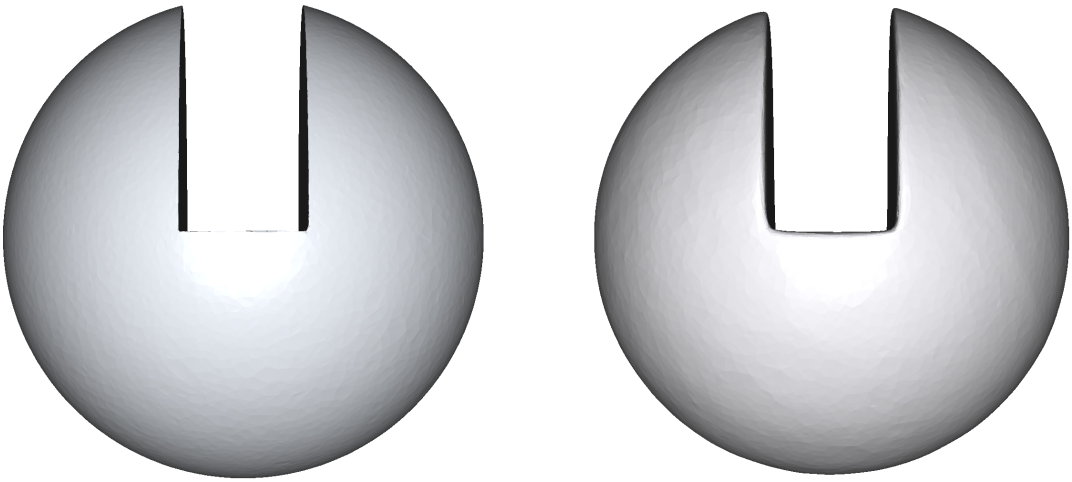

FIGURE 9. *Zoom on initial Zalesak's sphere (left) and on Zalesak's sphere after a whole rotation (right)*

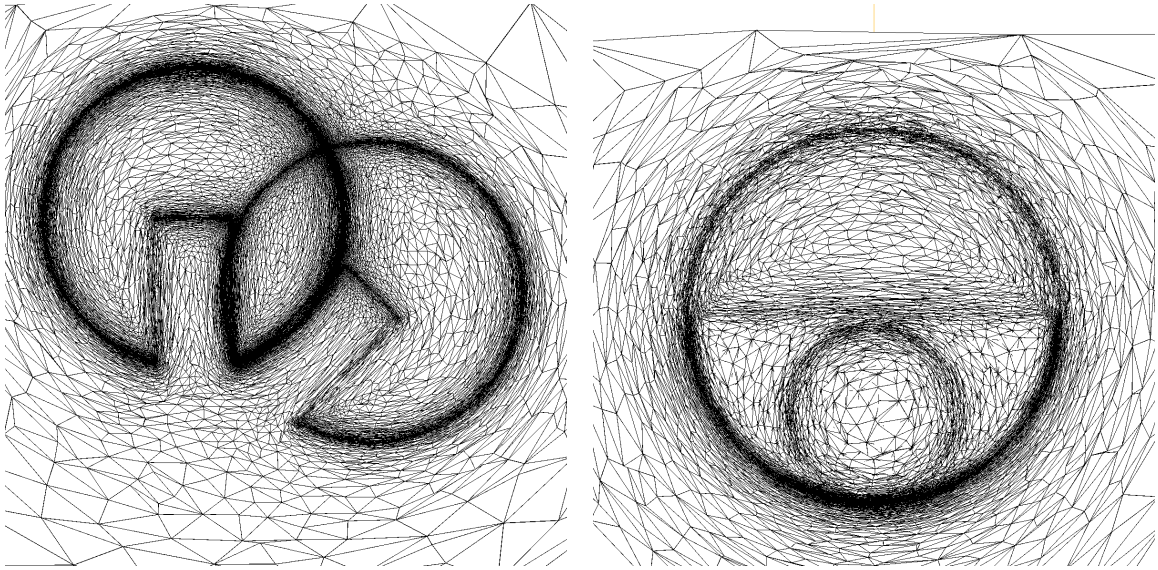

FIGURE 10. *Cut on a mesh adapted with respect to both surfaces at $t = T/2$ and $t = 5T/8$, following a plane $x = cste$ (left) and following a plane $y = cste$ (right)*

7. Appendix

Here is a variation of classical Gronwall's lemma for the estimation of the discrepancy between the solutions of two ODEs associated to different vector fields (see [37] for proof).

Lemma 7.1. *Let $V_1, V_2 : \mathbb{R}^{d+1} \to \mathbb{R}^d$ two vector fields, V_1 being Lipschitz continuous in the second variable with constant k, and such that there exists $\epsilon > 0$ with, for all $x \in \mathbb{R}^d$, $||V_1(x) - V_2(x)|| \leq \epsilon$. Let $x_1, x_2 : \mathbb{R}_+ \to \mathbb{R}^d$ some respective solutions to :*

$$\left\{ \begin{array}{rcl} x_1'(t) & = & V_1(t, x_1(t)) \\ x_1(0) & = & u_1 \end{array} \right. , \quad \left\{ \begin{array}{rcl} x_2'(t) & = & V_2(t, x_2(t)) \\ x_2(0) & = & u_2 \end{array} \right.$$

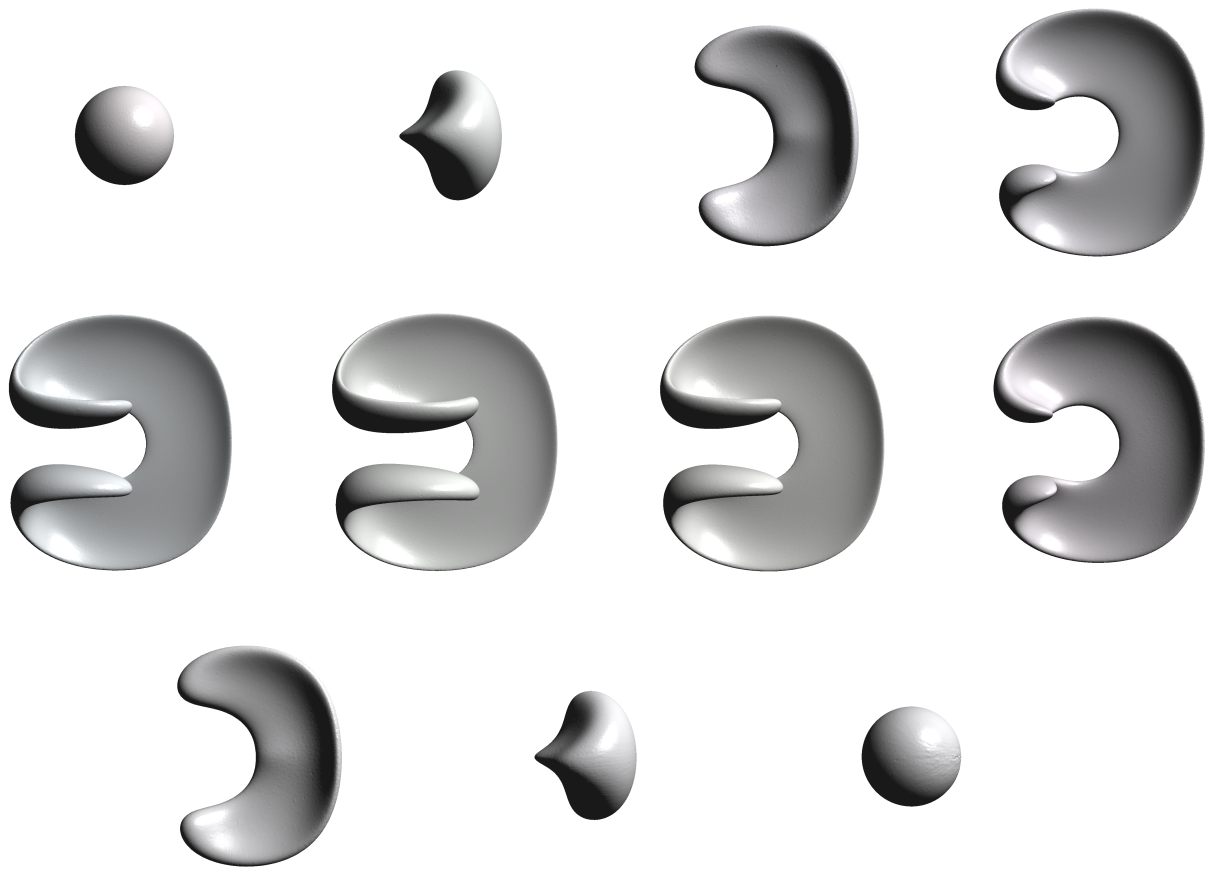

FIGURE 11. 3*d deformation test case : sequence of computed surfaces.*

	Δt_{it} (s)	cost of advection ($\%\Delta t_{it}$)	cost of remeshing ($\%\Delta t_{it}$)	cost of redistancing ($\%\Delta t_{it}$)	cost of metric computations ($\%\Delta t_{it}$)
Zalesak's disk (Test 2)	62.1	4.0	90.6	3.1	2.3
2d time-reversed vortex flow	142.6	5.4	83.0	9.0	2.6
2d deformation flow	253.1	4.7	87.4	7.9	3.0
Zalesak's sphere	1416	9.8	71.2	4.2	14.8
3d deformation	12417	16.9	51.2	26.1	5.8

TABLE 4. Costs of the steps of the proposed algorithm.

Then the following estimate holds

$$\forall t \in \mathbb{R}_+ , \ ||x_1(t) - x_2(t)|| \leq ||u_1 - u_2||e^{kt} + \frac{\epsilon}{k}(e^{kt} - 1).$$

REFERENCES

[1] H.T. AHN AND M. SHASHKOV, *Adaptive Moment-of-Fluid Method*, Submitted to Journal of Scientific Computing,(2010).

[2] F. ALAUZET AND P. FREY, *Estimateur d' erreur géométrique et métriques anisotropes pour l' adaptation de maillage. Partie I : aspects théoriques*, INRIA : Technical Report, 4759 (2003).

[3] G. ALLAIRE AND C. DAPOGNY AND P. FREY, *Topology and Geometry Optimization of Elastic Structures by Exact Deformation of Simplicial Mesh*, C. R. Acad. Sci. Paris, Ser. I (2011), doi : 10.1016/j.crma.2011.08.012.

[4] T. APEL, *Anisotropic Finite Elements : Local Estimates and Applications*, B.G. Teubner Stuttgart,Leipzig, Series of Advances in Numerical Mathematics, (1999).

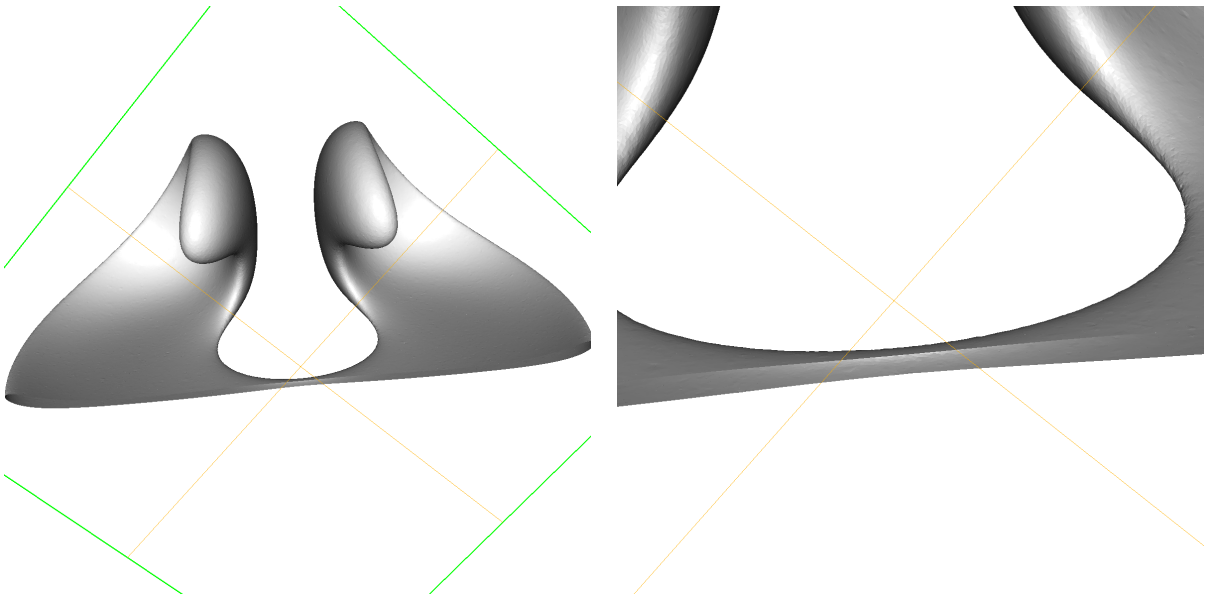

FIGURE 12. *A cut in the most stretched interface, at time $t = 1.5$ (left) ; a zoom on the cut (right).*

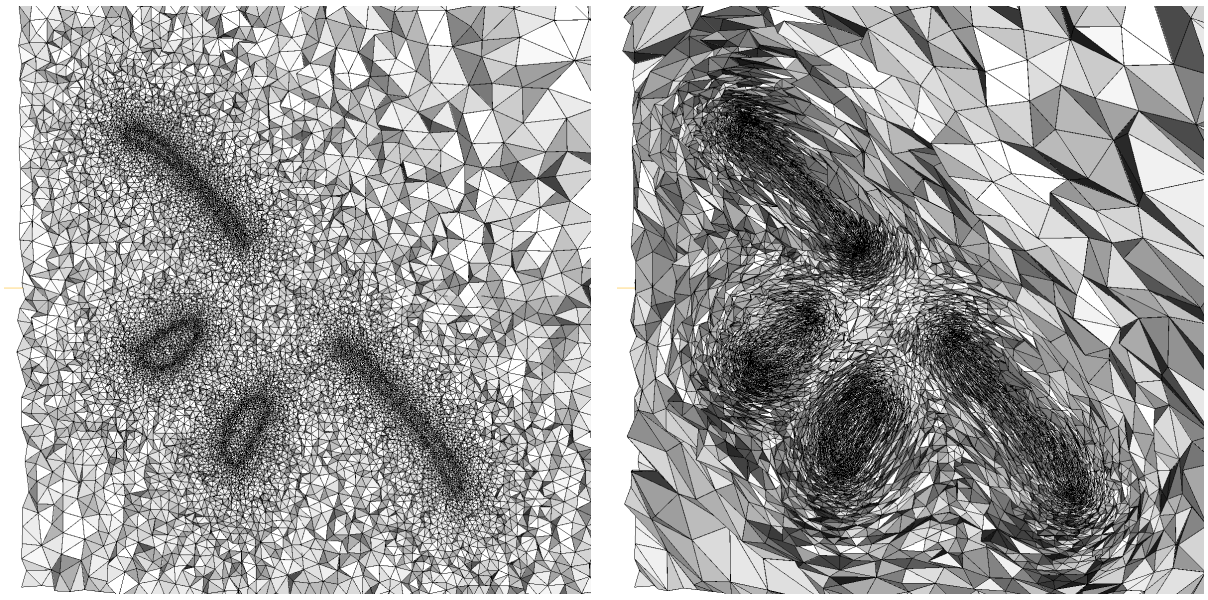

FIGURE 13. *(left) A cut in an mesh adapted to the evolving interface at time $t = 1.5$ using isotropic mesh adaptation ($\approx 1,500,000$ points) and (right) anisotropic mesh adaptation ($\approx 700,000$ points).*

[5] R. Ausas, E. Dari and G. Buscaglia, *A geometric mass-preserving redistancing scheme for the level set function.* , Int. J. Numer. Methods in Fluids, 65 (2011), pp. 989–1010.

[6] C. Bardos, *Problèmes aux limites pour les équations aux dérivées partielles du premier ordre à coefficients réels ; théorèmes d'approximation ; application à l'équation de transport*, Annales scientifiques de l'E.N.S., 4ème série, tome 3, (1970), pp. 185-233.

[7] L. Battaglia and M.A. Storti and J. D'Elia, *Bounded renormalization with continuous penalization for level set interface-capturing methods*, International Journal for Numerical Methods in Engineering, vol 84, *nb.*7 (2010), pp. 830-848.

[8] A. Bermúdez and M.R. Nogueiras and C. Vásquez, *Numerical analysis of convection-diffusion-reaction problems with higher order characteristics/finite elements. Part I : time discretization* , SIAM J. NUMER. ANAL, Vol. 44, No 5 (2006), pp. 1854-1876.
[9] A. Bermúdez and M.R. Nogueiras and C. Vásquez, *Numerical analysis of convection-diffusion-reaction problems with higher order characteristics/finite elements. Part II : fully discretized scheme and quadrature formulas* , SIAM J. NUMER. ANAL, Vol. 44, No 5 (2006), pp. 1829-1853.
[10] H. Borouchaki, P.Frey and F.Hecht, *Mesh gradation control*, Int. J. Numer. Methods in Engineering, 43 (1998), pp. 1143–1165.
[11] T.T.C. Bui, P. Frey and B. Maury, *A coupling strategy for solving two-fluid flows*, Int. J. Numer. Methods in Fluids, to appear (2010).
[12] A. Cervone and S. Manservisi and R. Scardovelli and S. Zaleski, *A geometrical predictor - corrector advection scheme and its application to the volume fraction function*, Journal of Scientific Computing, vol 228,(2009), pp. 406-419.
[13] D. Chopp, *Computing minimal surfaces via level-set curvature flow*, Journal of Computational Physics, 106 (1993), pp. 77–91.
[14] C. Dapogny and P. Frey, *Computation of the signed distance function to a discrete contour on adapted triangulation*, submitted (2010).
[15] E.F. D'Azevedo and R.B. Simpson, *On Optimal Triangular Meshes for Minimizing the Gradient Error*, Numerische Mathematik, 59 (1991), pp. 321–348.
[16] J.P. Demailly, *Analyse numérique et équations différentielles.*, EDP Sciences (2006).
[17] D. Enright and R. Fedkiw and J. Ferziger and I. Mitchell, *A Hybrid Particle Level Set Method for Improved Interface Capturing*, Journal of Computational Physics, 183 (2002), pp. 83–116.
[18] R.E. Ewing and H. Wang, *A Summary of Numerical Methods for Time-Dependent Advection-Dominated Partial Differential Equations*, Num. Anal., VII (2000), pp. 423-445.
[19] P.J. Frey and P.L. George, *Mesh Generation : Application to Finite Elements*, Wiley, 2nd Edition, (2008).
[20] C. Dobrzynski and P. Frey, *Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations*, Proc. of the 17th IMR, (2008), pp. 177–194.
[21] S. Ghosh Moulic and A. Salih, *Some numerical studies of interface advection properties of level set method*, Sādhanā, 34 (2009), pp. 271–298.
[22] F. Golse, *Distributions, analyse de Fourier, équations aux dérivées partielles*, Cours de l'Ecole Polytechnique (2010).
[23] W. Huang, *Metric Tensors for Anisotropic Mesh Generation*, Journal of Computational Physics, 204 (2005), pp. 633–665.
[24] H. Kim and M.S. Liou, *Accurate adaptive level set method and sharpening technique for three dimensional deforming interfaces*, Computer and Fluids, vol 44,(2011), pp. 111-129.
[25] R.J. Leveque, *High-Resolution Conservative Algorithms for Advection in Incompressible Flow.*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.
[26] E. Marchandise and J.-F. Remacle, *A stabilized finite element method using a discontinuous level set approach for solving two phase incompressible flows*, J. Comput. Phys., 219 (2006), pp. 780–800.
[27] C. Min and F. Gibou, *A Second Order Accurate Level Set Method on Non-graded Adaptive Cartesian Grids.*, Journal of Computational Physics, 225 (2007), pp. 300–321.
[28] S.J. Osher and J.A. Sethian,*Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[29] S. J. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, (2002).
[30] O. Pironneau, *The finite element methods for fluids.*, Wiley (1989).
[31] O. Pironneau, *Finite Element Characteristic Methods Requiring no Quadrature*, Journal of Scientific Computing, vol 43, *nb.*3 (2010), pp. 402-415.
[32] O. Pironneau, *On the Transport-Diffusion Algorithm and its Applications to the Navier-Stockes Equations*, Numerische Mathematik, vol 38 (1982), pp. 309-332.
[33] P.K. Smolarkiewicz, *The Multi-Dimensional Crowley Advection Scheme*, Monthly Weather Review, vol 110 (1982), pp. 1968-1983.
[34] J.A. Sethian, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry,Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).
[35] M. Sussman and E. Fatemi, *An Efficient, Interface-Preserving Level Set Redistancing algorithm and its Applications to Interfacial Incompressible Fluid Flow*, SIAM J. Sc. Comp., 20 (1999), pp. 1165–1191.
[36] M.G. Vallet, F. Hecht and B. Mantel, *Anisotropic Control of Mesh Generation Based upon a Voronoi Type Method*, Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, (1991).
[37] C. Viterbo, *Systèmes dynamiques et équations différentielles*, Cours de l'Ecole Polytechnique (2009).
[38] S.T. Zalesak, *Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids*, Journal of Computational Physics, 31 (1979), pp. 335–362.