A geometric mass-preserving redistancing scheme for the level set function

Roberto F. Ausas[1,*,†], Enzo A. Dari[1] and Gustavo C. Buscaglia[2]

[1]*Centro Atómico Bariloche e Instituto Balseiro, 8400 Bariloche, Argentina*
[2]*Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 13560-970 São Carlos, Brazil*

SUMMARY

In this paper we describe and evaluate a geometric mass-preserving redistancing procedure for the level set function on general structured grids. The proposed algorithm is adapted from a recent finite element-based method and preserves the mass by means of a localized mass correction. A salient feature of the scheme is the absence of adjustable parameters. The algorithm is tested in two and three spatial dimensions and compared with the widely used partial differential equation (PDE)-based redistancing method using structured Cartesian grids. Through the use of quantitative error measures of interest in level set methods, we show that the overall performance of the proposed geometric procedure is better than PDE-based reinitialization schemes, since it is more robust with comparable accuracy. We also show that the algorithm is well-suited for the highly stretched curvilinear grids used in CFD simulations. Copyright © 2010 John Wiley & Sons, Ltd.

1. INTRODUCTION

The level set method, introduced by Osher and Sethian in 1988 [1], has been extensively used in the past few years to treat problems involving free surfaces, basically due to its simplicity to deal with the complex topological changes that interfaces might undergo along their transport in a general situation. Additionally, quantities such as the curvature of the interface and other related information can be extracted from the level set function making it a very attractive and powerful tool for problems in two and three spatial dimensions.

As is well known, one of the main drawbacks of this method for free surface problems involving incompressible flows is the lack of mass conservation and excessive diffusion, which leads to unphysical motions of the interface that severely deteriorate the accuracy and stability of the results. These difficulties have been addressed in basically three different ways:

- by improving the numerical algorithms used to transport the level set function;
- by combining the level set method with other computational techniques;
- by trying to keep the level set function as regular as possible, using the so-called reinitialization or redistancing procedures.

---

*Correspondence to: Roberto F. Ausas, Centro Atómico Bariloche e Instituto Balseiro, 8400 Bariloche, Argentina.
†E-mail: rfausas@gmail.com

Regarding the first alternative, there are many methods to solve the level set equation, such as finite volume and finite difference methods which combine total variation diminishing (TVD) schemes in time, introduced by Shu and Osher [2, 3], and essentially non-oscillatory (ENO) schemes in space, based on the ideas first proposed by Harten *et al.* [4, 5] to solve Hamilton–Jacobi type equations. In this area, the TVD Runge–Kutta and Hamilton–Jacobi weighted-ENO schemes developed in [6] are considered to be state of the art for solving the level set equation within the framework of Eulerian methods [7]. In this case, curvilinear coordinates can be used to deal with complex geometries (see for instance [8] and [9]). It should be mentioned that TVD schemes can also be used in space as flux limiter methods, as done for instance by Olsson and Kreiss [10]. Stabilized finite elements and discontinuous Galerkin methods are used as well for treating the level set equation. In this case, unstructured meshes can be employed and local grid refinement becomes an easy task. A comparison of such methods is done in [11]. In [12] a discontinuous Galerkin method is proposed and compared with several other methods, including the ENO/RK(3) scheme presented in [13] and the HJ-WENO(5)/RK(3) scheme used in [14, 15]. Finally, semi-Lagrangian schemes, which can be implemented in very simple and efficient ways, are also used in level set methods (see [14, 16, 17]).

With respect to the second alternative in the bulleted list above, hybrid methods that combine the level set method either with Lagrangian particles or with the VOF (volume of fluid) method have been developed. The first option consists in moving massless particles forward in time to redefine the level set function by means of some procedure at the end of each time step or with a predefined frequency, see for instance [14, 15] and [18]. The other option uses the VOF method (see e.g. [19]), another surface capturing method for free surface flows, to correct the level set function so as to locally enforce mass conservation as done by Sussman in [20] and [21] for example.

In this paper we focus on the redistancing procedure. Its purpose is to ensure that the level set function remains smooth close to the interface. This is achieved by periodically redefining it, while trying to maintain the interface intact. The reinitialization improves the robustness of the computations by keeping the distortion of the level set function under control. A natural choice to reinitialize the level set function is the signed distance function to the interface. The idea of reinitialization was first presented in the work of Chopp [22]. Later, Tsitsiklis [23] proposed in a completely different framework, a method that can be used for the construction of distance functions, and therefore for reinitialization, known as the fast marching method, further popularized by Sethian (see [24, 25]). A second paper of Chopp [26], which introduces some improvements to the method, can also be mentioned in this context. It should be pointed out that reinitialization can be avoided using a velocity extension that preserves the signed distance function (see [27, 28]). On the other hand, several partial differential equation (PDE)-based methods have been devised for the purpose of redistancing, which solve the so-called reinitialization equation as originally proposed by Sussman *et al.* [13, 29, 30].

It should be pointed out that, in general, it is not possible to reinitialize the level set function maintaining the interface intact in a discrete problem. In fact, the space of level set functions that share the same given interface is extremely small [31] and it is likely that none of its elements provides a reasonable approximation to the distance function. The consequence of this is that each reinitialization distorts the interface to some extent, implying a local numerical creation/destruction of fluid mass. However, this distortion is not explicit in PDE-based methods but embedded in the discretization adopted for the reinitialization equation. The use of high-order schemes, together with *ad hoc* correction terms, are needed to minimize the interface distortion during reinitialization, which otherwise completely destroys the accuracy of the computations [7]. Though improved versions of PDE-based reinitialization exist for Cartesian grids (see e.g. [32, 33]), they are not well-suited for highly stretched curvilinear grids.

In this paper we adapt the reinitialization scheme of Mut *et al.* [34] to the case of structured, curvilinear finite difference grids. The scheme was originally developed for unstructured meshes of linear finite elements, and a simple subdivision of each quadrilateral (or of each hexahedron in 3D) is used to build a mesh suitable for applying it. The advantages of the proposed reinitialization

scheme are its simplicity, its flexibility to handle arbitrarily distorted meshes, and the absence of adjustable parameters.

We begin by showing the necessity of redistancing in Section 2. In Section 3 we describe the proposed method, together with the TVD Runge–Kutta third-order ENO scheme that is used to evolve the level set equation, for which we use a finite volume implementation very similar to that presented in [8]. As the proposed method is based on a piecewise-linear representation of the level set function, concerns may arise as to its accuracy. Section 3 also contains a brief summary of a widely used PDE-based method, the HJWENO-RK scheme of Jiang and Peng [6]. This method will be compared to the proposed one in the numerical examples.

In Section 4, numerical experiments are shown in two and three spatial dimensions, including the rigid rotation of Zalesak's disk, the deformation of a circle under a swirling flow vortex, and the deformation of a 3D sphere, which are classical benchmark cases in level set methods. Specific measures of error of interest in free surface problems are used to evaluate the results. Finally, to illustrate the versatility of the proposed geometric redistancing scheme, we couple it with a finite difference upwind method in curvilinear coordinates that uses a second-order TVD van Albada scheme as flux limiter for the transport of the level set function, similar to the one used in CFDShip-Iowa [9]. We present numerical examples using curvilinear grids that have appreciable distortion in order to be able to test the mass-preserving scheme in relevant situations that might appear in real CFD computations. We draw some conclusions in Section 6.

2. MOTIVATION

It is not obvious whether the periodic reinitialization of the level set function is convenient or not in computations. It strongly depends, among other things, on the particular case considered, the method used to transport the level set function ϕ, the level of discretization used and of course on the reinitialization algorithm itself. However, as we know, most level set methods assume that ϕ has to be reinitialized periodically for robustness of computations. To illustrate this point, a two-dimensional academic problem is shown below, in which we use a simple semi-Lagrangian algorithm for the transport of the level set function, in combination with the mass-preserving redistancing scheme we propose (details about the scheme will be given later in Section 3).

For the case considered, the initial value of the function ϕ has as zero-level set a circle with radius a centered at the origin, i.e.

$$\phi(x, y, t=0) = a - (x^2 + y^2)^{1/2}, \quad (x, y) \in \Omega,$$

where $\Omega = [-0.75, 0.75] \times [-0.75, 0.75]$.

Then, we consider the flow field corresponding to a doublet with intensity Λ plus an uniform stream of magnitude $U_0 = \Lambda/a^2$ and with an additional circulation $\Gamma = 4\pi k a U_0$, where k is a constant. Using polar coordinates (r, θ), the velocity field reads as

$$u_r = \cos(\theta) \left[U_0 - \frac{\Lambda}{r^2} \eta(r) \right],$$

$$u_\theta = -\sin(\theta) \left[U_0 + \frac{\Lambda}{r^2} \eta(r) \right] + \frac{\Gamma}{2\pi r} \eta(r),$$

where $\eta(r)$ is a regularization function defined as

$$\eta(r) = \begin{cases} \frac{1}{2} \left[1 - \cos\left(\frac{4\pi r}{a}\right) \right], & \text{if } r < a/4, \\ 1, & \text{if } r \geqslant a/4. \end{cases} \tag{1}$$

The regularization function avoids the singularity at the origin. The presence of $\eta(r)$ makes the velocity field non divergence free for $r < a/4$, however, semi-Lagrangian schemes are perfectly capable of handling such velocity fields without any additional effort.

As the radial component of the velocity u_r is identically zero for $r = a$, in the exact problem the interface remains unaltered. However, this will not necessarily be true in the discretized problem, and our aim is to study the role of the redistancing procedure in keeping the interface as accurate as possible.

This case is specially tailored so that changing the parameters k and Γ different behaviors can be observed:

- $k = 1 \Rightarrow \Gamma = 4\pi a U_0$
    This case has a stagnation point at $(r, \theta) = (a, -\frac{\pi}{2})$.
- $k > 1 \Rightarrow \Gamma > 4\pi a U_0$
    In this case the stagnation point occurs at some point $(r, \theta) = (r_1, -\frac{\pi}{2}), r_1 > a$.

The cases corresponding to $k < 1$, with two stagnation points at $(r, \theta) = (a, -\frac{\pi}{2} - \theta_s)$ and $(r, \theta) = (a, -\frac{\pi}{2} + \theta_s)$, behave similar to the case with $k = 1$. Therefore, we will consider three different values for k: 1, 2, and 4 and for all of them, a mesh consisting of 70×70 cells and a time step $\delta t = 0.1$ will be used to evolve the level set equation with the semi-Lagrangian scheme. We use an RK4 scheme with a time step $\delta t / 50$ to go back along the characteristics, so that the error in the computations is dominated by the interpolation error.

The results can be seen in Figure 1, where the flow field and the level set at $t = 4.4$ corresponding to each case are drawn (please note that in some cases, the resulting interface when the redistancing procedure is included cannot be distinguished from the exact interface).

The first case corresponding to $k = 1$, at the top, clearly shows the benefits of including the redistancing step. Without it the interface severely distorts (see the spurious bulge on the right bottom quadrant) and loses its regularity (see the detail).

The second case, corresponding to $k = 2$, in the middle, still shows a significant difference between including the redistancing step and not including it (see the bottom part of the interface). Again, the interface loses its regularity in some regions as shown in the detail.

Finally, for $k = 4$, at the bottom, the effect of including the redistancing step is much less noticeable.

The results above show that in some situations the reinitialization of the level set function can be advantageous in the numerical simulation of free surface flows. We will thus assume that reinitialization *is* performed, and concentrate on *how* to perform it economically and accurately on general meshes.

3. NUMERICAL FORMULATION

3.1. Level set method

We adopt the conservation form of the level set equation for a divergence-free velocity field, i.e.

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\vec{u} \phi) = 0, \tag{2}$$

where ϕ is the level set function whose zero isocontour represents the interface and $\vec{u} = (u_x, u_y, u_z)$ is the velocity field. Both the level set function and the velocity field are functions of $(\vec{x}, t), \vec{x} \in \Omega, t > 0$.

We use a finite volume method similar to that adopted in [8], in which the level set equation is convected by means of a TVD Runge–Kutta scheme. The value of ϕ at time level $n + 1$ is obtained as follows:

$$\phi^{(1)} = \phi^n - \delta t \mathcal{L}(\phi^n, t^n),$$

$$\phi^{(2)} = \frac{3}{4}\phi^n + \frac{1}{4}\phi^{(1)} - \frac{1}{4}\delta t \mathcal{L}(\phi^{(1)}, t^n + \delta t), \tag{3}$$

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\phi^{(2)} - \frac{2}{3}\delta t \mathcal{L}\left(\phi^{(2)}, t^n + \frac{1}{2}\delta t\right),$$
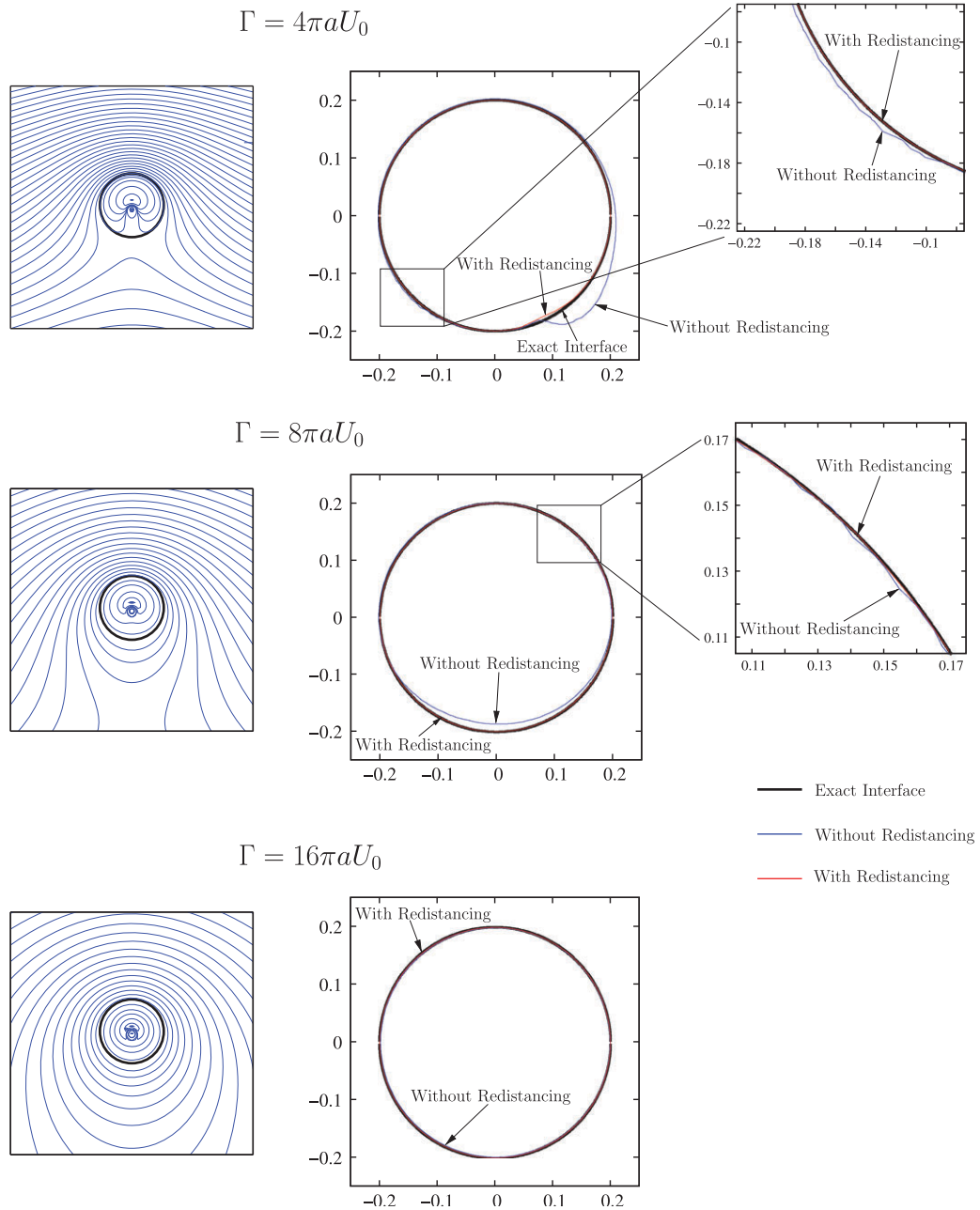
Figure 1. Flow fields and level sets after several time steps for all the cases considered. Top: $k=1$, middle: $k=2$, bottom: $k=4$. In the plot are shown the exact interface with a thick line and drawn with thiner lines, the corresponding results for the cases with and without the redistancing procedure.

where ϕ^n is the value of ϕ at the time level n, δt is the time step, and $\mathscr{L}(\phi, t)$ is the spatial operator in Equation (2), i.e.

$$\mathscr{L}(\phi, t) = \nabla \cdot (\vec{u}\phi). \tag{4}$$

Now, in order to obtain a fully discrete method, which for the sake of simplicity is presented here in two spatial dimensions, we subdivide the computational domain $\Omega = [0, L_x] \times [0, L_y]$ into I and J uniform cells in the x and y directions, respectively, such that the grid spacing will be

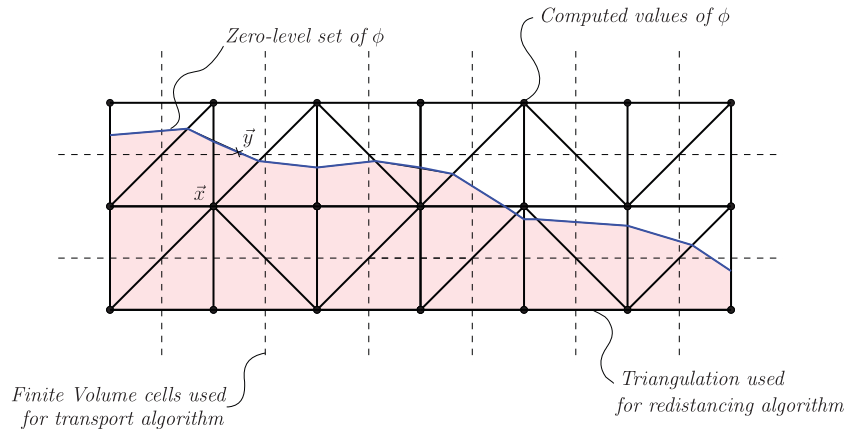R. F. AUSAS, E. A. DARI AND G. C. BUSCAGLIA

Figure 2. Schematic showing the finite volume discretization cells and the corresponding triangulation $\mathcal{T}_h$ for the redistancing algorithm.

given by δx and δy. Then, the discrete form of (4) for the control volume (i, j) will be simply given by

$$\mathcal{L}(\phi) = \frac{(u_x\phi)_{i+1/2,j} - (u_x\phi)_{i-1/2,j}}{\delta x} + \frac{(u_y\phi)_{i,j+1/2} - (u_y\phi)_{i,j-1/2}}{\delta y}. \tag{5}$$

The extension to three spatial dimensions is straightforward. In this scheme, u_x and u_y are computed at the cell faces, but ϕ is given at the cell center of the control volume, from which, the cell face values of ϕ ($\phi_{i+1/2,j}, \phi_{i,j+1/2}, \ldots$) are built by using a third-order accurate ENO interpolation. Details for the construction of the corresponding stencils can be found in [3] or [8].

3.2. Geometric mass-preserving redistancing scheme

The geometric mass-preserving reinitialization algorithm proposed here was originally devised to be used within the finite element framework [34], in which the level set function is linearly interpolated over each simplex of an arbitrary triangulation $\mathcal{T}_h$ (triangles in 2D and tetrahedra in 3D).

To adapt it to finite volume-structured meshes we thus define a finite element partition $\mathcal{T}_h$ of Ω and assign the values of ϕ, computed at the center of gravity of the finite volume cells, as nodal values on $\mathcal{T}_h$. In 2D, the triangulation $\mathcal{T}_h$ is obtained by dividing each cell into two triangles as shown in Figure 2, whereas in 3D each hexahedral cell is divided into six tetrahedra. Therefore, the number of simplices in this partition will be two (respectively, six) times the number of cells used for the finite volume discretization in the 2D (respectively, 3D) case. Although this partition is not unique, it has been checked that all such partitions lead to essentially the same results.

We now proceed to describe the geometric redistancing algorithm. Let V_h be the space of continuous functions that are linear inside each simplex of $\mathcal{T}_h$. Let $\phi_h \in V_h$ be a function, and let $\mathcal{S}_h$ be its zero-level set. Our aim is to find a function $\tilde{\phi}_h \in V_h$ which approximates the signed distance function d to $\mathcal{S}_h$ defined by

$$d(\vec{x}) = \text{sign}[\phi_h(\vec{x})] \min_{\vec{y} \in \mathcal{S}_h} \|\vec{x} - \vec{y}\|, \tag{6}$$

noting that, in general, d does not belong to V_h. As an example, consider the problem of computing the distance to a square as sketched in Figure 3. In this simple case, we can clearly see that the exact distance d to the interface, for any point such as $\vec{x}$ (see Figure 3), will not be a function that belongs to V_h as indicated by the contours of d (continuous red lines).

The algorithm is divided into two different stages

1. Reinitialization of nodes that belong to interface simplices (**First Neighbors of $\mathcal{S}_h$**).
2. Reinitialization of nodes not belonging to interface simplices (**Rest of the mesh**).
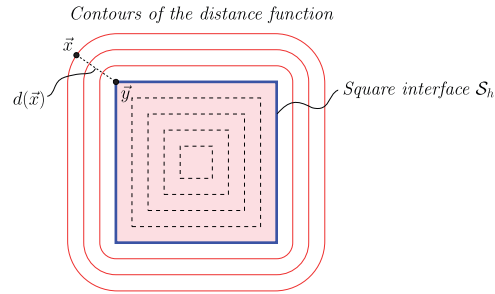
Figure 3. Contours of the distance function d to a square. Example showing that the distance to the interface from outside the square region (contours drawn with continuous lines) does not belong to the space V_h.

3.2.1. Reinitialization of first neighbors. Let $\mathcal{P}$ be the set of nodal points that are adjacent to the zero-level set of ϕ_h, in the sense that they are vertices of simplices inside which ϕ_h changes sign.

Step 1: We begin by computing

$$\phi_h^*(\vec{X}) = d(\vec{X}), \quad \forall \vec{X} \in \mathcal{P}, \tag{7}$$

so that the nodal values of the intermediate function ϕ_h^* coincide with the exact (signed) distance d.

Let us define $\mathcal{K}$ as the set of simplices in which ϕ_h changes sign, so that $\mathcal{S}_h \subset \mathcal{K}$. Notice that the nodes in $\mathcal{P}$ are the vertices of the simplices in $\mathcal{K}$. The 'brute force' approach to compute the exact distance in this case simply consists in the reconstruction of $\mathcal{S}_h$ inside each element in $\mathcal{K}$ (a segment in 2D, a triangle or a quadrilateral in 3D) followed by the computation of the corresponding geometrical distance from each node in $\mathcal{P}$ to the reconstructed interface. This is effectively implemented by using auxiliary data structures such as quad/oct-trees to search for nearby simplices and amounts to a small fraction of the overall cost, since the number of nodes in $\mathcal{P}$ is small.

Once ϕ_h^* is known, we must introduce a correction, *since the volume enclosed by its zero-level set is different from that enclosed by $\mathcal{S}_h$, leading to a mass loss (or gain) that is unacceptable for practical purposes.* We now describe how to compute the correction function ψ_h such that the final function

$$\tilde{\phi}_h = \phi_h^* + \psi_h, \tag{8}$$

is the desired *reinitialized level-set function*. The zero-level set of $\tilde{\phi}_h$, in particular, encloses the same volume as that of ϕ_h.

Let us introduce a notation for the difference in the volumes defined by ϕ_h and ϕ_h^*,

$$\Delta V(\phi_h, \phi_h^*) = \int_{\mathcal{K}} [\mathscr{H}(\phi_h(\vec{x})) - \mathscr{H}(\phi_h^*(\vec{x}))]\, d\vec{x}, \tag{9}$$

where $\mathscr{H}$ is the Heaviside function ($\mathscr{H}(s) = 1$ if $s > 0$, $\mathscr{H}(s) = 0$ otherwise), so that our objective is to determine ψ_h such that $\Delta V(\phi_h, \phi_h^* + \psi_h) = 0$.

For this purpose, we first notice that ΔV is the sum of contributions of the simplices $K \in \mathcal{K}$, namely,

$$\Delta V(\phi_h, \phi_h^*) = \sum_{K \in \mathcal{K}} \Delta V_K(\phi_h, \phi_h^*) = \sum_{K \in \mathcal{K}} \int_K [\mathscr{H}(\phi_h(\vec{x})) - \mathscr{H}(\phi_h^*(\vec{x}))]\, d\vec{x}, \tag{10}$$

leading us to the second step.

Step 2: Determine the *piecewise constant* function η_h, with constant value η_K inside each $K \in \mathcal{K}$ such that

$$\Delta V_K(\phi_h, \phi_h^* + \eta_K) = 0, \tag{11}$$

Notice that Equation (11) is a nonlinear equation for η_K, which is solved independently for each K using a simple secant procedure that converges in very few iterations.

The piecewise-constant function η_h computed in this way contains the information about how much volume loss or gain is contributed by each simplex in $\mathcal{K}$. It is not possible, however, to define $\tilde{\phi}_h$ as $\phi_h^* + \eta_h$, because η_h is a discontinuous function and is thus multiply valued at the nodes in $\mathcal{P}$.

Step 3: We now compute a continuous function ξ_h as an approximate L^2-orthogonal projection of η_h onto the space of piecewise continuous functions in $\mathcal{K}$. This is implemented in practice by simply computing the nodal values of ξ_h averaging over the simplices that share a node. Let I be a node in $\mathcal{P}$, and let N_I be the number of simplices in $\mathcal{K}$ that contain I, then we define

$$\xi_h(\vec{X}_I) = \frac{1}{N_I} \sum_{\substack{K \in \mathcal{K} \\ I \in K}} \eta_K. \tag{12}$$

Step 4: Finally, the correction ψ_h is computed on $\mathcal{P}$ as

$$\psi_h = C \xi_h, \tag{13}$$

where C is the constant that globally preserves volume; i.e. C satisfies

$$\Delta V(\phi_h, \phi_h^* + C\xi_h) = 0. \tag{14}$$

This nonlinear equation for C is again solved by a simple secant method and converges in very few iterations. From the description above it is evident that there are no adjustable parameters in the scheme, except for the numerical tolerance in the secant algorithms, which does not play any significant role since convergence to machine precision takes place quickly. Steps 1–4 are summarized in Table I.

The main advantage of the algorithm, as compared to previous ones, is that ξ_h localizes the correction in those regions where the mass loss/gain produced by ϕ_h^* is the largest. Correcting ϕ_h^* by a constant, as done by other authors [35], corresponds to taking $\xi_h = 1$ and unphysically distributes the correction uniformly over the interface simplices.

3.2.2. Reinitialization of the rest of the mesh.

As discussed by Carrica *et al.* [36], the most critical part of the reinitialization procedure is the reinitialization of first neighbors. Once ϕ_h^* is known on $\mathcal{P}$, these values are used as boundary conditions for the reinitialization of the rest of the mesh points. This can be done using a PDE-based scheme, as the one described in the following section. We however adopt the geometric scheme introduced by Mut *et al.* [34], for which the mesh is subdivided into simplices as in the previous section. We briefly recall the procedure below.

We will describe the calculation of $\tilde{\phi}_h$ only on the positive side of $\mathcal{S}_h$; i.e. for the set of nodes $\mathcal{R}$ at which ϕ_h is positive and that do not belong to $\mathcal{P}$. We assume that $\tilde{\phi}_h$ is already known in $\mathcal{P}$.

Step 5 (Initialization): Let I be a node in $\mathcal{R}$, and let C_I be the set of nodes connected to I, I not included (notice that $C_I \subset (\mathcal{P} \cup \mathcal{R})$). The initial guess we use for $\tilde{\phi}_h$ is a distance-along-edges approximation, i.e. the unique function satisfying

$$\tilde{\phi}_h(\vec{X}_I) = \min_{J \in C_I} [\tilde{\phi}_h(\vec{X}_J) + |\vec{X}_I - \vec{X}_J|].$$

In the process of initializing $\tilde{\phi}_h$ with this option, the elements can be ordered so as to render the algorithm more effective. In addition, if one wants to calculate $\tilde{\phi}_h$ up to a distance δ from $\mathcal{S}_h$, one simply initializes $\tilde{\phi}_h$ as equal to δ over $\mathcal{R}$.

Step 6 (Evaluation): The simplices in the mesh, except those in $\mathcal{K}$, are swept until $\tilde{\phi}_h$ no longer changes. For each simplex, and for each node J of the simplex (coordinates denoted by $\vec{X}_J$), $\tilde{\phi}_h$

Table I. **Steps 1–4** for the reinitialization of ***first neighbors of*** $\mathscr{S}_h$: computation of the exact distance followed by the mass correction.

Step 1: Compute the exact distance to $\mathscr{S}_h$ (Brute force)

- Set $\tilde{d}(\vec{X}_n) = +\infty$ **for** $n = 1, 2, \ldots, N_{\mathrm{nod}}^{\mathscr{P}}$

do $(K = 1, N_{\mathrm{el}}^{\mathscr{K}})$
- Find $\mathscr{S}_K$, the reconstruction of $\mathscr{S}_h$ in K using ϕ_h

  do $(I = 1, N_{\mathrm{nod}}^{\mathscr{P}})$
  - Compute d_I s.t. $d_I = \min_{\vec{x} \in \mathscr{S}_K} |\vec{X}_I - \vec{x}|$
  - Set $\tilde{d}(\vec{X}_I) = d_I$ **if** $(\tilde{d}(\vec{X}_I) > d_I)$
  end do
end do
- Set $\phi_h^*(\vec{X}_n) = \tilde{d}(\vec{X}_n)$ **for** $n = 1, 2, \ldots, N_{\mathrm{nod}}^{\mathscr{P}}$

Step 2: Find η_h, a piecewise constant function

do $(K = 1, N_{\mathrm{el}}^{\mathscr{K}})$
- Set $\delta V_K = \Delta V_K(\phi_h, \phi_h^*)$

  do while $(|\delta V_K| > 10^{-15})$
  - Find $\mathscr{S}_K$, the reconstruction of $\mathscr{S}_h$ in K using $\phi_h^* + \eta_K$
  - Set $\eta_K = -\delta V_K / \mathrm{size}(\mathscr{S}_K)$
  - Set $\delta V_K = \Delta V_K(\phi_h, \phi_h^* + \eta_K)$
  end do while
end do

Step 3: Find ξ_h, the orthogonal projection of η_h

do $(I = 1, N_{\mathrm{nod}}^{\mathscr{P}})$
- Set $\xi_h(\vec{X}_I) = 0$

  do $(K = 1, N_I)$
  - Set $\xi_h(\vec{X}_I) \leftarrow \xi_h(\vec{X}_I) + \eta_K / N_I$
  end do
end do

Step 4: Find $\psi_h = \phi_h^* + C\xi_h$

- Initialize $\delta V^{(i)}, C^{(i)}$ **for** $i = 1, 2$
- Set $i = 3$

do while $(|\delta V^{(i)}| > 10^{-15})$
- Set $m^{(i)} = (C^{(i-1)} - C^{(i-2)}) / (\delta V^{(i-1)} - \delta V^{(i-2)})$
- Set $C^{(i)} = C^{(i-2)} - m^{(i)} \delta V^{(i-2)}$
- Set $\delta V^{(i)} = \Delta V(\phi_h, \phi_h^* + C^{(i)} \xi_h)$
- Set $i \leftarrow i + 1$
end do while
- Set $C = C^{(i)}$

N_I is the number of simplices in $\mathscr{K}$ that contain node I.

is interpolated linearly on the opposite face F_J, using the current values at the nodes. Then, a tentative new value η_J of $\tilde{\phi}_h$ at node J is calculated as

$$\eta_J = \min_{\vec{x} \in F_J} [\tilde{\phi}_h(\vec{x}) + |\vec{X}_J - \vec{x}|]. \tag{15}$$

Finally, $\tilde{\phi}_h(\vec{X}_J)$ is updated to the value η_J *if the current value is greater than* η_J. This is illustrated in Figure 4 and in Table II we summarize both procedures.

3.3. PDE-based redistancing scheme

For the sake of completeness we briefly describe the PDE-based redistancing method that will be used for comparison, together with its discretization as proposed by Jiang and Peng [6].
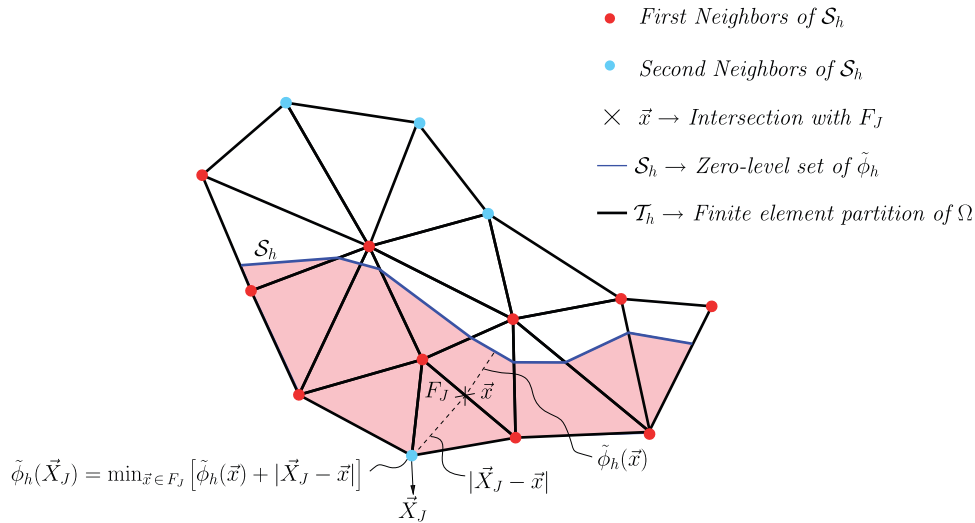
- ● *First Neighbors of $\mathcal{S}_h$*
- ● *Second Neighbors of $\mathcal{S}_h$*
- × *$\vec{x} \rightarrow$ Intersection with F_J*
- — *$\mathcal{S}_h \rightarrow$ Zero-level set of $\tilde{\phi}_h$*
- — *$\mathcal{T}_h \rightarrow$ Finite element partition of Ω*

$$\tilde{\phi}_h(\vec{X}_J) = \min_{\vec{x} \in F_J}\left[\tilde{\phi}_h(\vec{x}) + |\vec{X}_J - \vec{x}|\right]$$

Figure 4. Schematic of **Step 6 (Evaluation)** for the reinitialization of the rest of mesh.

Let ϕ^0 be a level set function with zero-level set denoted by $\mathcal{S}$. Our aim is to compute from this initial data ϕ^0 an approximation $\tilde{\phi}$ for the distance function d to the zero-level set $\mathcal{S}$ of ϕ^0, defined as in (6) (with ϕ_h replaced by ϕ^0 and $\mathcal{S}_h$ replaced by $\mathcal{S}$).

The property $\|\nabla d\| = 1$ motivates the redistancing method in which the following hyperbolic PDE is solved as

$$\frac{\partial \tilde{\phi}}{\partial \tau} + \text{sign}(\phi^0)(\|\nabla \tilde{\phi}\| - 1) = 0 \quad \text{in } \Omega,$$
$$\tilde{\phi}(\vec{x}, 0) = \phi^0(\vec{x}),$$

(16)

where τ is a fictitious time. The steady state solution of Equation (16) is an approximation of the signed distance function to the interface implicitly defined by ϕ_0.

Now, the PDE-based reinitialization method considered here discretizes Equation (16) by an RK-HJWENO (weighted ENO) scheme (see [6]). According to [7] the accuracy gained with this type of high-order scheme is comparable to the one obtained by means of the improvements introduced in [29], in which mass conservation is enforced through the introduction of a constraint in the solution of (16). The approach is very similar to that presented in the previous section for the discretization of the level set equation (2). First, the semidiscrete form of Equation (16) for node (i, j), which for simplicity is presented again in two spatial dimensions reads as

$$\frac{\partial \tilde{\phi}_{i,j}}{\partial \tau} = -\hat{H}(x_i, y_j, \tilde{\phi}_{i,j}, \tilde{\phi}^+_{x,i,j}, \tilde{\phi}^-_{x,i,j}, \tilde{\phi}^+_{y,i,j}, \tilde{\phi}^-_{y,i,j}),$$

(17)

where $\hat{H}$ is the discrete form of the spatial operator $\text{sign}(\phi_0)(\|\nabla \tilde{\phi}\| - 1)$. Then, for the construction of $\tilde{\phi}^\pm_{x,i,j}$ and $\tilde{\phi}^\pm_{y,i,j}$, which are the WENO approximations to $\frac{\partial \tilde{\phi}}{\partial x}(x_i, y_j)$ and $\frac{\partial \tilde{\phi}}{\partial y}(x_i, y_j)$, respectively, we follow Jiang and Peng [6] exactly. Finally, we advance in pseudo-time τ the system of ODEs given in (17) until $|\|\nabla \tilde{\phi}\| - 1| < 10^{-5}$. For that purpose, we use a fourth-order Runge–Kutta method, which reads as

$$\tilde{\phi}^{(1)} = \tilde{\phi}^m - \frac{1}{2}\delta\tau \hat{H}(\tilde{\phi}^m),$$

$$\tilde{\phi}^{(2)} = \tilde{\phi}^m - \frac{1}{2}\delta\tau \hat{H}(\tilde{\phi}^{(1)}),$$

Table II. **Steps 5** and **6** for the reinitialization of rest of the mesh.

Step 5: Edge distance approximation

- Set $changes = 1$
do while $(changes == 1)$
 - Set $changes = 0$
 do $(iel = 1, N_{el})$
  do $(I = 1, N_{npe})$
   if $(glob(I) \notin \mathscr{P})$ **then**
    - Find e_I s.t. $e_I = \min_{J \in C_I}[\tilde{\phi}_h(\vec{X}_J) + |\vec{X}_J - \vec{X}_I|]$
    if $(\tilde{\phi}_h(\vec{X}_I) > e_I)$ **then**
     - Set $\tilde{\phi}_h(\vec{X}_I) = e_I$
     - Set $changes = 1$
    end if
   end if
  end do
 end do
end do while

Step 6: Shadow distance correction

- Set $changes = 1$
do while $(changes == 1)$
 - Set $changes = 0$
 do $(iel = 1, N_{el})$
  do $(J = 1, N_{npe})$
   if $(glob(J) \notin \mathscr{P})$ **then**
    - Find F_J, the opposite face of node J in iel
    - Find η_J s.t. $\eta_J = \min_{\vec{x} \in F_J}[\tilde{\phi}_h(\vec{x}) + |\vec{X}_J - \vec{x}|]$
    if $(\tilde{\phi}_h(\vec{X}_J) > \eta_J)$ **then**
     - Set $\tilde{\phi}_h(\vec{X}_J) = \eta_J$
     - Set $changes = 1$
    end if
   end if
  end do
 end do
end do while

Nomenclature: $\mathscr{P}$, set of nodal points adjacent to $\mathscr{S}_h$; N_{el}, total number of elements (simplices); N_{npe}, number of nodes per single element (three for a triangle, four for a tetrahedron); $glob(I)$, the global index of local incidence I; C_I, set of nodes connected to I, I not included.

$$\tilde{\phi}^{(3)} = \tilde{\phi}^m - \delta\tau \hat{H}(\tilde{\phi}^{(2)}),$$

$$\tilde{\phi}^{m+1} = -\frac{1}{3}\tilde{\phi}^m + \frac{1}{3}\tilde{\phi}^{(1)} + \frac{2}{3}\tilde{\phi}^{(2)} + \frac{1}{3}\tilde{\phi}^{(3)} - \frac{1}{6}\delta\tau\hat{H}(\tilde{\phi}^{(3)}),$$

where the supraindex m refers to the pseudo-time step (i.e. $\tau_m = m\delta\tau$) and $\delta\tau = \delta x/2$.

4. RESULTS

To assess the behavior of the proposed reinitialization procedure, the following two measures of error will be used:

$$e_{\mathrm{m}} = \max_n |V(\phi^n) - V(\phi_e^n)|/V(\phi_e^n), \tag{18}$$

$$e_{\mathrm{p}} = \max_n D(\mathscr{S}(\phi^n), \mathscr{S}(\phi_e^n)), \tag{19}$$

where ϕ^n is the computed solution at time t^n, linearly interpolated whenever needed, ϕ_e^n is the exact solution, $V(\phi)$ is the volume enclosed by ϕ,

$$V(\phi) = \int_\Omega \mathscr{H}(\phi(\vec{x})) \, d\vec{x}, \qquad (20)$$

$\mathscr{S}(\phi)$ is the zero-level set of ϕ

$$\mathscr{S}(\phi) = \{\vec{x} \in \Omega, \phi(\vec{x}) = 0\}, \qquad (21)$$

and $D(\cdot, \cdot)$ is a distance between two surfaces. In the experiments we choose

$$D(\mathscr{S}(\phi^n), \mathscr{S}(\phi_e^n)) = \max_n \left\{ \max_{x \in \mathscr{S}(\phi_e^n)} \min_{y \in \mathscr{S}(\phi^n)} \|\vec{x} - \vec{y}\|, \max_{x \in \mathscr{S}(\phi^n)} \min_{y \in \mathscr{S}(\phi_e^n)} \|\vec{x} - \vec{y}\| \right\}. \qquad (22)$$

The first measure of error e_m is the classical 'mass error'. The second measure e_p provides information about the position of the computed interface with respect to the exact one and will be referred to as the 'position error'. These two measures of error are useless if the level set is not reasonably resolved by the mesh. We have thus chosen cases in which local feature sizes of the level set are not smaller than the grid resolution.

Concerning the actual computation of $D(\mathscr{S}(\phi^n), \mathscr{S}(\phi_e^n))$, it is performed by seeding the initial surface with a huge number of points, each of which is tracked in time with high accuracy, and then the max and min over $\mathscr{S}(\phi_e^n)$ are approximated by taking the max and min over these auxiliary tracking points, respectively.

4.1. Numerical experiments in 2D

Two examples will be presented in the two-dimensional case: the Zalesak's disk problem and the stretching of a circle under a swirling flow vortex. These are standard benchmarks in the area of interface transport [7, 12, 29, 37–39].

4.1.1. Zalesak's disk.
The initial data is a slotted disk centered at $(0.5, 0.75)$ with a radius of 0.15, a slot width of 0.075, and a slot length of 0.25 in a unit square computational domain.

We first study the influence of the repeated application of the redistancing procedures without considering the advection step into account (zero convection velocity). Significant differences are observed between the two schemes. On the left of Figure 5 we show the evolution of the PDE-based scheme towards its steady state, as a function of the number of pseudo-time steps. The trend of the method to make sharp corners round is evident, together with the local loss of mass. In fact, we believe that this behavior of the PDE-based redistancing scheme may be responsible for much of the numerical diffusion attributed to level set-based methods. As the number of pseudo-time steps is increased (or equivalently, as the tolerance to achieving pseudo-steady state is tightened), the smoothing effect of PDE-based reinitialization becomes stronger. The number of pseudo-time steps performed at each time step thus becomes a tunable parameter, for which the optimal choice depends on the specific problem, mesh, etc.

On the right of Figure 5 we show the effect of the repeated application of the geometry-based scheme. Notice that one application of this method is analogous to evolving the PDE-based method to steady state, so that no hidden tunable parameter exists. Repeated application only occurs, as in this example, when the interface velocity is zero. Although the reinitialization distorts the interface, the distortion is much smaller than that caused by the PDE-based method. In addition, the mass is exactly conserved, as by design geometry-based reinitialization is mass-conserving.

Now, in order to study the interaction between advection and reinitialization, we advect the slotted disk by the following velocity field

$$\begin{aligned} u_x &= \frac{\pi}{3.14}(0.5 - y), \\ u_y &= \frac{\pi}{3.14}(x - 0.5), \end{aligned} \qquad (23)$$
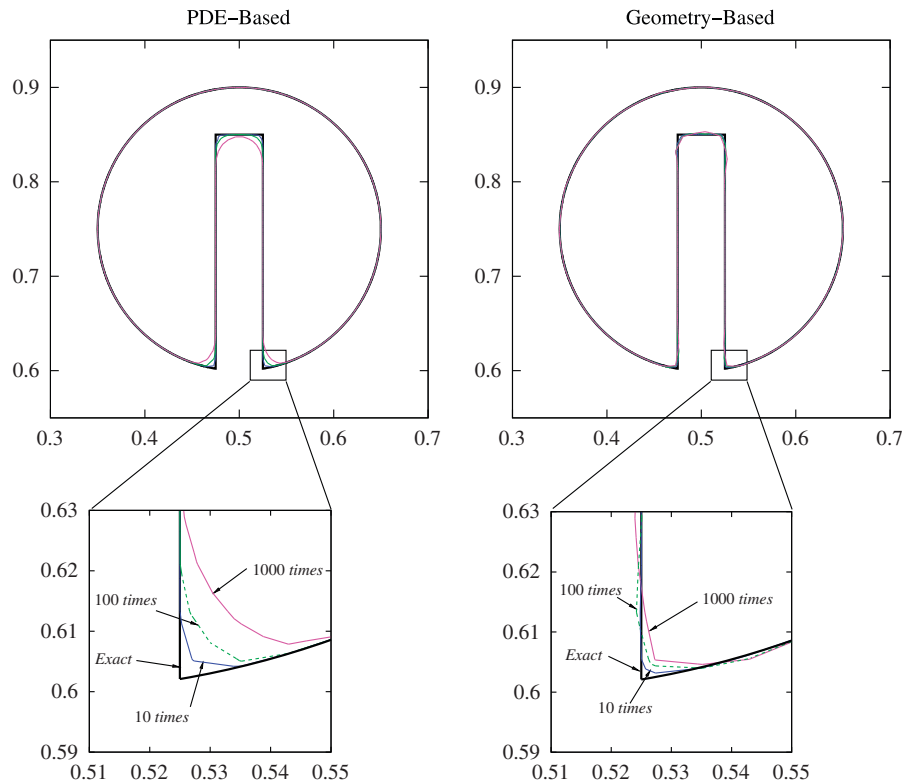
Figure 5. Effect of the repeated application of the PDE-based (left) redistancing scheme on a 128×128 grid. Note in the detail the interfaces after 10, 100, and 1000 pseudo-time steps. On the right, an analogous plot shows the effect of 10, 100, and 1000 applications of the geometry-based scheme.

which represents a rigid body rotation with respect to $(0.5, 0.5)$. The disk completes one revolution after 6.28 units of time. In this experiment, and in all the experiments that follow, the reinitialization procedure is applied every 10 time steps.

In Figure 6 we compare the final stage of the Zalesak's disk with the exact result after one turn, for different grid resolutions. The time step for the first case ($h = 1/64$) is 6.28/600. For the rest of the grids we maintain the same Courant number. As it can be seen the geometry-based scheme performs similar to the PDE-based scheme when the grid resolution is good enough ($h = 1/256$, $h = 1/512$, and $h = 1/1024$), whereas the former outperforms the latter when the grid resolution is poor. In Figure 7 we compare the evolution of the disk for the mesh with 128×128 cells, when both reinitialization algorithms are used. On the left, we show the results for the PDE-based scheme (continuous line) and on the right the results for the geometry-based algorithm (dashed line). Finally, in Table III we present the two measures of error for both algorithms and for the different grids considered. It should be noted, on the one hand, that the mass error of the PDE-based scheme is smaller than that of the geometry-based scheme when $h = 1/128$, 1/256, 1/512, and 1/1024, which, we think, is the result of the compensation of the mass gained near the top of the slot and the mass lost near the corners at the bottom of the slot. The mass loss of the geometry-based scheme obviously arises because of advection, since reinitialization is mass-conserving. Regarding the second measure of error e_p, on the other hand, the geometry-based scheme performs better than the PDE-based for grid resolutions 1/64, 1/128, and 1/256, whereas for better resolutions (1/512 and 1/1024) the PDE-based scheme is superior.

4.1.2. Swirling flow vortex. The initial data consists of a disk centered at $(0.5, 0.75)$ with a radius of 0.15. The computational domain is again a square of size $[0, 1] \times [0, 1]$. The disk of fluid is
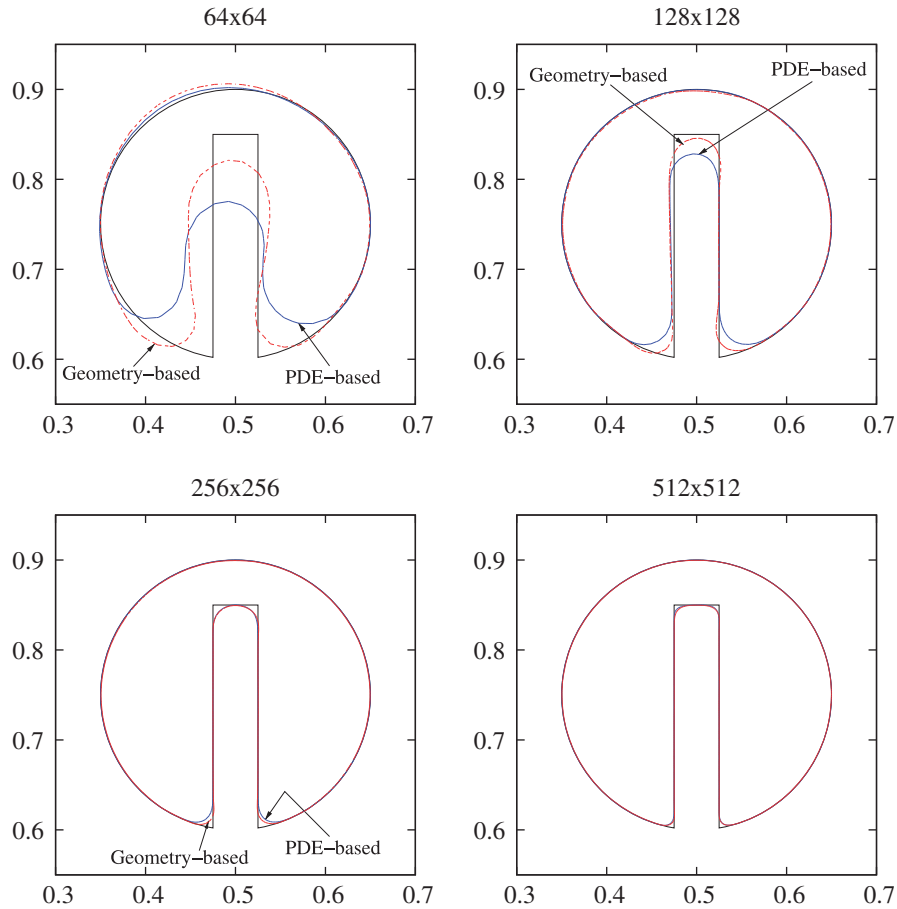
Figure 6. Final stage of the Zalesak's disk after one revolution for different grids. The thick line corresponds to the exact solution, the continuous line to the PDE-based scheme, and the dashed line to the geometry-based scheme.

convected by the following time-dependent divergence-free velocity field

$$
\begin{aligned}
u_x &= -\sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\
u_y &= \sin(2\pi x)\sin^2(\pi y)\cos(\pi t/T).
\end{aligned}
\tag{24}
$$

In this case, the initial disk is stretched out into a filament and after a certain time T it comes back to its initial state. This reversal period T is taken equal to 2, so that the size of the tale of the filament will be reasonably well resolved for all times by the mesh used for computations.

First, in Figure 8 we compare the interface at $t=T/2$ (maximum deformation) and $t=T$ (final time) for all the grids previously considered. Now, the time step for the case with $h=1/64$ was equal to $2/300$ and as done before the Courant number was kept the same for the other grids. Again, both algorithms perform similarly when the grid resolution is good. Actually, for the case with $h=1/512$ the difference cannot be seen without zooming in at the sharp corners. In Table IV we present the different measures of error. In this case, we can see that the geometric mass-preserving scheme in general has a better performance than the PDE-based scheme.

We summarize some results in Figure 9, where the error e_p against h in logarithmic scale is reported. For Zalesak's disk e_p turns out to be smaller for the geometric mass-preserving scheme than for the PDE-based one in the coarser meshes, whereas for the swirling flow vortex, e_p results smaller for the geometric mass-preserving scheme than for the PDE-based one in all meshes.
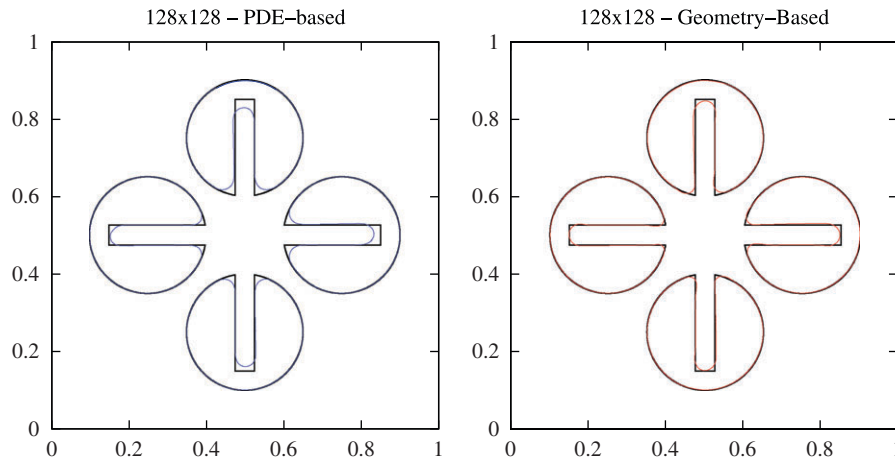
Figure 7. Evolution of the Zalesak's disk using the PDE-based (left) and the geometry-based (right) redistancing procedures. The thick line corresponds to the exact solution.

Table III. Measures of error for the Zalesak's disk after one revolution.

Mesh	e_m (%)		e_p	
	PDE based	GEO based	PDE based	GEO based
64×64	7.902	4.949	0.0709	0.0353
128×128	0.789	2.254	0.0291	0.0134
256×256	0.202	0.477	0.0126	0.0105
512×512	0.018	0.253	0.0065	0.0070
1024×1024	0.0046	0.119	0.0040	0.0050

The PDE-based scheme exhibits an asymptotic order of convergence of h for the Zalesak's disk problem. This order is optimal, since the sharp corners of the disk limit its approximability to first order. It was verified that, for the finest meshes, the error is in fact concentrated at those corners (see Figure 6). The order of the geometric scheme, on the other hand, behaves more like $h^{1/2}$. This may be a cumulative deteriorating effect of geometric mass correction in non-smooth problems as the mesh and the time step are refined, reinitializing every 10 time steps being excessive. Reducing the frequency of reinitialization may prove useful in such situations.

As in the swirling flow vortex case, the surface remains smooth, the optimal approximation order is h^2, which arises from the linear interpolation used in the computation of $\mathscr{S}(\phi)$ in Equation (21). This order is exhibited by the PDE-based scheme, whereas the order of the geometric scheme appears to be slightly smaller than h^2. Notice however that, for the finest mesh, e_p is still about four times smaller for the geometric scheme than for the PDE-based one.

In Figure 10 the error as a function of time is shown for two grid resolutions. Notice that for the swirling flow vortex case the reduction in error brought by the geometric scheme is maximal at time $t=1$, which corresponds to the maximum stretching.

Finally, to assess the robustness of the two methods, the value of the reversal period for the swirling flow vortex is changed to $T=8$. It is revealing to analyze the interface at $t=T/2$ (maximum stretching) and at $t=T$ (final time). Figure 11 shows these interfaces for results corresponding to $h=1/128$, in which the tale of the stretched circle is clearly under-resolved.

4.2. Numerical experiments in 3D

For the three-dimensional case, we present, on the one hand, results using Cartesian coordinates and comparing both redistancing procedures and, on the other hand, results using curvilinear coordinates with the geometric mass-preserving redistancing scheme coupled with a finite difference
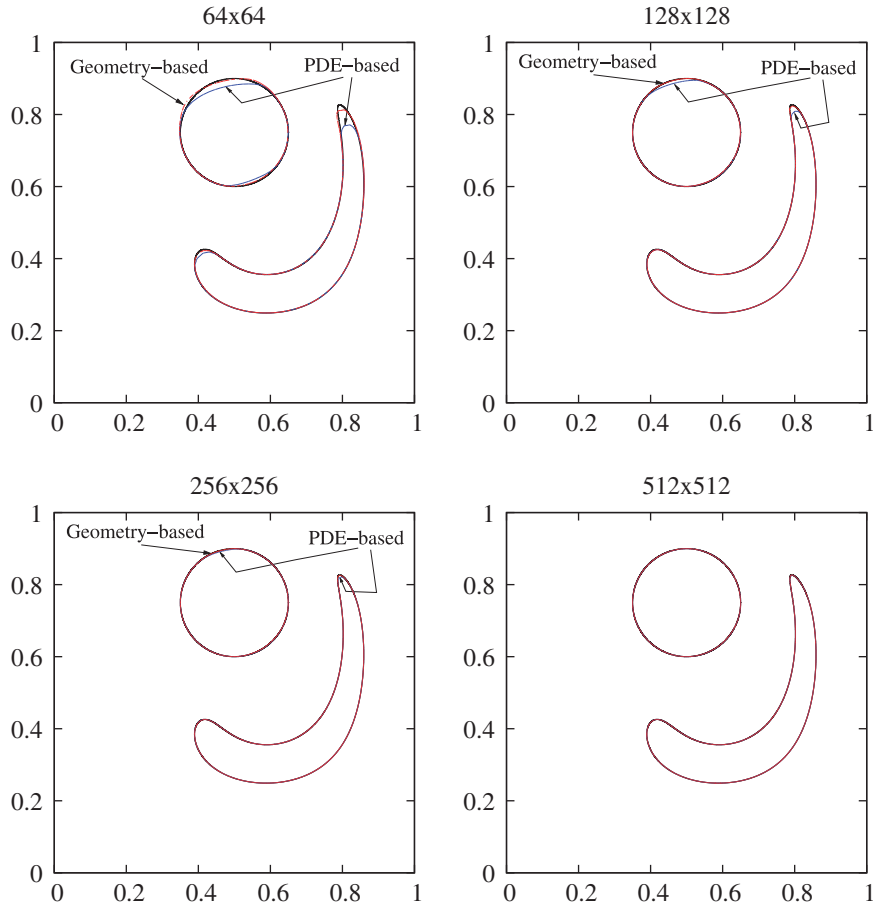
Figure 8. Intermediate $(t=T/2)$ and final stage $(t=T)$ of the disk under a swirling flow vortex with reversal period $T=2$ for different grid resolutions. The thick line corresponds to the exact solution, the continuous line to the PDE-based scheme, and the dashed line to the geometry-based scheme.

Table IV. Measures of error for the stretching of a disk under a swirling flow vortex with reversal period $T=2$.

Mesh	e_m (%)		e_p	
	PDE based	GEO based	PDE based	GEO based
64×64	5.074	0.668	0.0641	0.0146
128×128	1.643	0.118	0.0272	0.0028
256×256	0.408	0.257	0.0100	0.0016
512×512	0.086	0.136	0.0021	0.0005

second-order TVD van Albada scheme for the transport of the level set function, similar to the one used in CFDShip-Iowa as already mentioned in the Introduction.

4.2.1. Deformation vortex—Cartesian coordinates. In this example, the initial data consists simply of a sphere, centered at $(0.35, 0.35, 0.35)$ and with a radius of 0.15. The computational domain is the unit cube. The sphere is then convected by the following solenoidal field

$$
\begin{aligned}
u_x &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \\
u_y &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \\
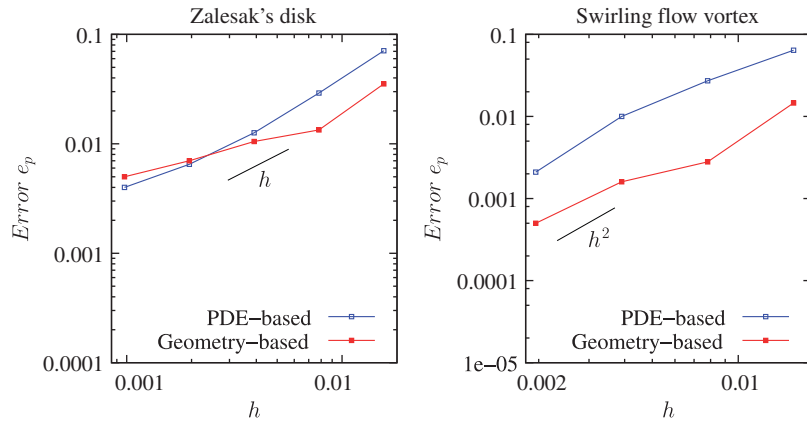u_z &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T),
\end{aligned}
\tag{25}
$$

Figure 9. Measure of error e_p versus h. Left: Rigid body rotation of Zalesak's disk. Right: Deformation of the disk under the swirling flow vortex.
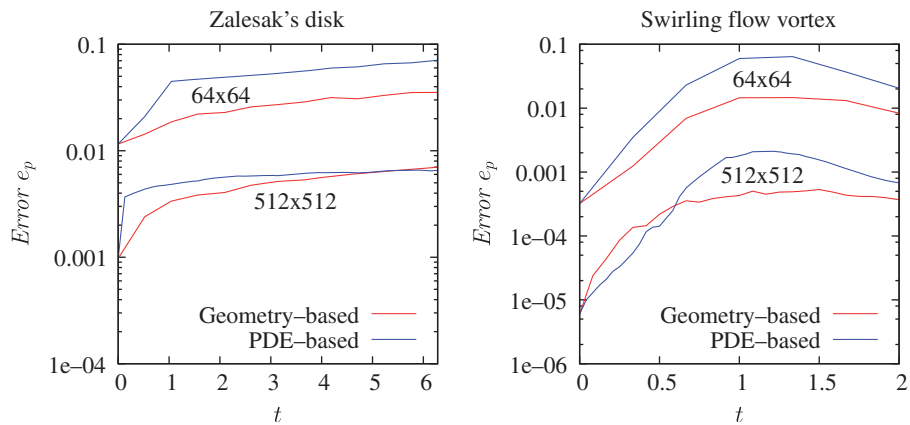

Figure 10. Measure of error e_p versus time t. Left: Rigid body rotation of Zalesak's disk. Right: Deformation of the disk under the swirling flow vortex.
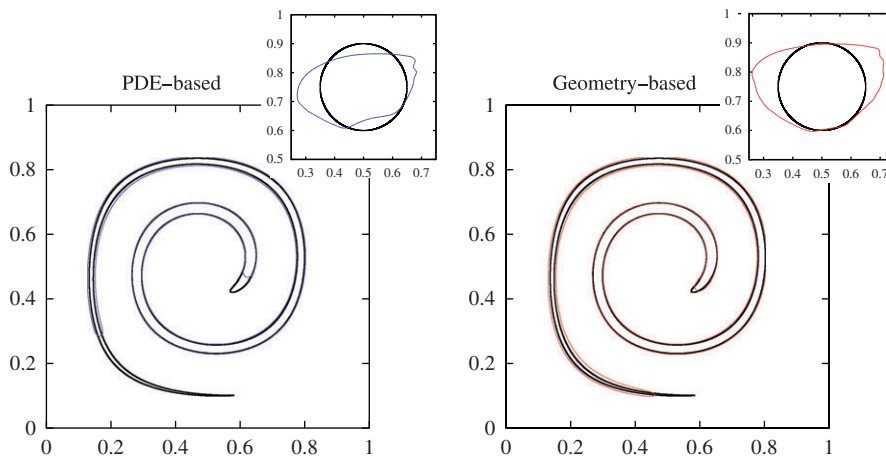

Figure 11. Intermediate ($t = T/2$) and final stage ($t = T$, shown in the insert) of the disk under a swirling flow vortex with reversal period $T = 8$, using the PDE-based (left) and the geometry-based (right) redistancing procedures. The thick black line corresponds to the exact solution.

Table V. Measures of error for the deformation of a sphere under a three-dimensional vortex. Cartesian coordinates.

Mesh	e_m (%)		e_p	
	PDE based	GEO based	PDE based	GEO based
$64 \times 64 \times 64$	40.21	2.241	0.0673	0.0267
$128 \times 128 \times 128$	10.76	1.543	0.0355	0.0056
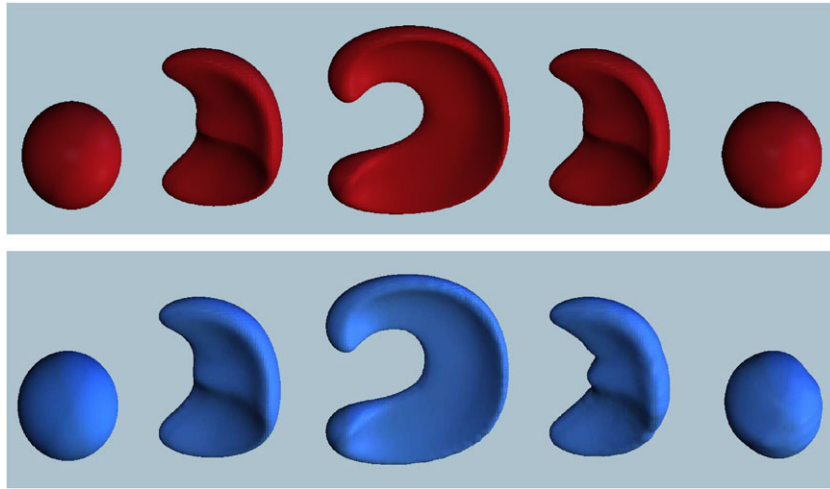
Figure 12. Evolution of a sphere under a three-dimensional deformation vortex. Comparison of the geometry-based redistancing scheme (top) with the PDE-based redistancing scheme (bottom). Cartesian grid $128 \times 128 \times 128$.

again, as in the $2D$ case, the velocity field is modulated by a periodic function, such that the sphere will recover its initial state after a time $T = 2$.

In Table V we present the two measures of error, in this case only for two different grids of $64 \times 64 \times 64$ and $128 \times 128 \times 128$ cells. The time step was taken equal to $2/400$ and $2/800$, respectively. In Figure 12 we plot the zero-level set at different times for both algorithms for the case with $h = 1/128$. At the top are the results for the geometry-based redistancing and at the bottom the results for the PDE-based redistancing. From both the figure and the table we can see that the geometry-based redistancing has a better performance.

4.2.2. Zalesak's sphere—curvilinear coordinates. The initial data consists of a notched sphere centered at $(0.5, 0.75, 0.3)$ of radius 0.15, with a slot width of 0.075 and a slot length of 0.25. The sphere is rotating in the xy-plane with the velocity field given in (23) ($u_z = 0$). The computational domain is the region $[0, 1] \times [0, 1] \times [0, 0.6]$ deformed by the mapping given by

$$x(\xi, \eta, \zeta) = \xi - \frac{5}{16\pi} \sin(2\pi(\xi - \eta - \zeta)),$$

$$y(\xi, \eta, \zeta) = \eta - \frac{5}{16\pi} \sin(2\pi(\eta - \xi - \zeta)), \tag{26}$$

$$z(\xi, \eta, \zeta) = \zeta - \frac{5}{16\pi} \sin(2\pi(\zeta - \eta - \xi)).$$

The resulting mesh (see Figure 13) is then used to compute the appropriate metric coefficients by using a second-order finite difference rule, in order to solve the transformed level set equation (see [9]). The grid in this case consists of $128 \times 128 \times 64$ cells and the time step is equal to $6.28/800$.
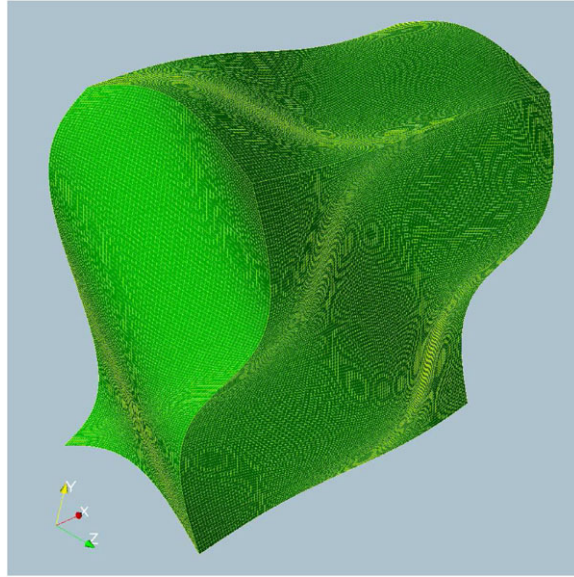
Figure 13. Three-dimensional curvilinear grid used and reference system for the transport of the Zalesak's sphere.
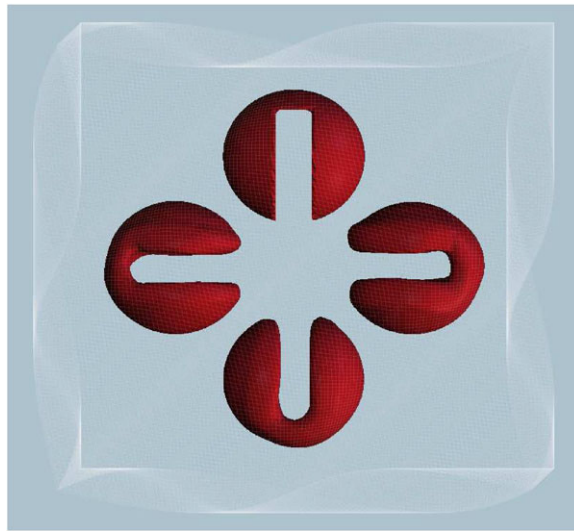

Figure 14. Evolution of the Zalesak's sphere using the geometric mass-preserving redistancing scheme. Curvilinear grid of $128 \times 128 \times 64$ cells.

In this case, for the sake of simplicity, instead of e_p we compute the following measure of error e_l

$$e_l = \max_n \int_{\Omega} |\mathcal{H}(\phi^n(\vec{x})) - \mathcal{H}(\phi_e^n(\vec{x}))| \, d\vec{x}. \tag{27}$$

This is possible since we are dealing here with a rigid body rotation, which makes the computation of ϕ_e^n very easy.

The results are shown in Figure 14 where we can appreciate that the Zalesak's sphere is reasonably preserved. Regarding the measures of error we have computed a mass change e_m of 1.859% and a value for e_l equal to 0.00162.
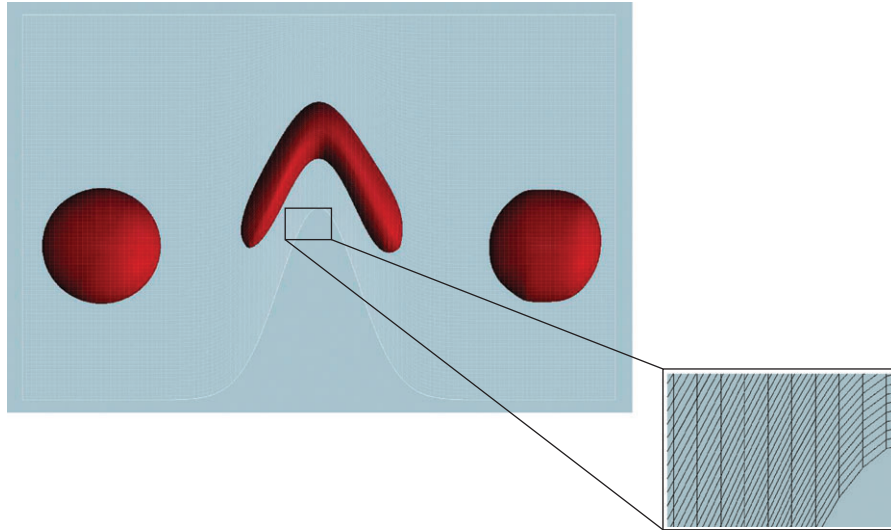
Figure 15. Evolution of the sphere approaching a bump using the geometric mass-preserving redistancing scheme. Curvilinear grid of $128 \times 102 \times 62$ cells.

4.2.3. Sphere approaching a bump—curvilinear coordinates. For this last example, the initial condition corresponds to a sphere centered at $(-0.05, 0.4, 0.25)$ of radius 0.15. The computational domain is the region $[-0.25, 1.25] \times [0, 1] \times [0, 0.5]$ transformed under the following mapping (see [40])

$$
\begin{aligned}
x(\xi, \eta, \zeta) &= \xi, \\
y(\xi, \eta, \zeta) &= B(\xi) + \eta(1 - B(\xi)), \\
z(\xi, \eta, \zeta) &= \zeta,
\end{aligned}
\tag{28}
$$

where the function B is given by

$$
B(\xi) = \frac{1}{2} e^{-50(\xi - 0.5)^2},
\tag{29}
$$

which represents a bump centered at $x = 0.5$. The sphere is then transported by the following divergence-free velocity field based on the shape of the bump

$$
\begin{aligned}
u_x &= \frac{1}{1 - B(x)}, \\
u_y &= \frac{B'(x)(1 - y)}{(1 - B(x))^2}, \\
u_z &= 0.
\end{aligned}
\tag{30}
$$

In this numerical test the grid consists of $128 \times 102 \times 62$ cells and the time step is equal to $1/800$.

The results are shown in Figure 15, where a detail of the curvilinear grid can be observed. In this case, the mass change e_m was 4.686% and the value of $e_p = 0.0306$. We should mention that the error reported here (which is the maximum over t) happens when the level set passes near the cusp of the bump, where the maximum distortion of cells is present, as seen in the detail of the grid.

5. CONCLUSIONS

In this paper we have discussed some issues related to the reinitialization of the level set function and we have focused on the description and evaluation of a *geometric mass-preserving* redistancing scheme that was originally introduced in the framework of finite elements.

The geometric mass-preserving algorithm proposed can be used on an arbitrary triangulation of the computational domain, making it a very attractive tool to be used on any type of discretized domains such as the structured curvilinear grids widely used in CFD computations. A salient feature of the scheme is its simplicity and lack of adjustable parameters, which is an important difference as compared to other available methods.

The scheme is designed to preserve the mass (or the volume) delimited by the zero-level set, which is defined by linear interpolation on a subdivision of the computational grid into simplices.

In the numerical tests using Cartesian grids in two and three spatial dimensions, we have observed in some cases a better performance of the geometric mass-preserving redistancing scheme and in some cases a better performance of the PDE-based method used for comparison, which is formally of a higher order of convergence. This was illustrated qualitatively by means of plots of the level set and quantitatively by means of computing relevant measures of error for level set methods. Numerical examples were then reported, showing that the performance of the method does not deteriorate when applied on arbitrary curvilinear grids.

REFERENCES

1. Osher S, Sethian J. Front propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics* 1988; **79**:12–49.
2. Shu C, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics* 1988; **77**:439–471.
3. Shu C, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes II (two). *Journal of Computational Physics* 1989; **83**:32–78.
4. Harten A, Osher S. Uniformly high-order accurate essentially non-oscillatory schemes I. *SIAM Journal on Numerical Analysis* 1987; **24**:279–309.
5. Harten A, Engquist B, Osher S, Chakravarthy S. Uniformly high-order accurate essentially non-oscillatory schemes III. *Journal of Computational Physics* 1987; **71**:231–303.
6. Jiang GS, Peng D. Weighted eno schemes for Hamilton–Jacobi equations. *SIAM Journal on Scientific Computing* 2000; **21**:2126–2144.
7. Losasso F, Fedkiw R, Osher S. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids* 2006; **35**:995–1010.
8. Yue W, Lin CL, Patel V. Numerical simulation of unsteady multidimensional free surface motions by level set method. *International Journal for Numerical Methods in Fluids* 2003; **42**:853–884.
9. Carrica P, Wilson R, Noack R, Xing T, Kandasamy M, Shao J, Sakamoto N, Stern F. A dynamic overset, single-phase level set approach for viscous ship flows and large amplitude motions and maneuvering. *26th Symposium on Naval Hydrodynamics*, Rome, Italy, 2006.
10. Olsson E, Kreiss G. A conservative level set method for two phase flow. *Journal of Computational Physics* 2005; **210**:225–246.
11. Di Pietro D, Lo Forte S, Parolini N. Mass preserving finite element implementations of the level set method. *Applied Numerical Mathematics* 2006; **56**:1179–1195.
12. Marchandise E, Remacle JF, Chevaugeon N. A quadrature-free discontinuous Galerkin method for the level set equation. *Journal of Computational Physics* 2006; **212**:338–357.
13. Sussman M, Fatemi E, Smereka P, Osher S. An improved level set method for incompressible two-fluid flows. *Computers and Fluids* 1999; **20**:1165–1191.
14. Enright D, Losasso F, Fedkiw R. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures* 2005; **83**:479–490.

15. Enright D, Fedkiw R, Ferziger J, Mitchell I. A hybrid particle level set method for improved interface capturing. *Computers and Structures* 2002; **83**:479–490.
16. Strain J. Semi-Lagrangian methods for level set equations. *Journal of Computational Physics* 1999; **151**:498–533.
17. Strain J. Tree methods for moving interfaces. *Journal of Computational Physics* 1999; **151**:616–648.
18. Zhaorui L, Farhad A, Shih T. A hybrid Lagrangian–Eulerian particle-level set method for numerical simulations of two-fluid turbulent flows. *International Journal for Numerical Methods in Fluids* 2007; DOI: 10.1002/fld.1621.
19. Hirt C, Nichols H. Volume of fluid (VOF) methods for the dynamics of free boundaries. *Applied Numerical Mathematics* 1981; **39**:201–225.
20. Sussman M. A second order coupled level set and volume of fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics* 2003; **187**:110–136.
21. Sussman M, Puckett E. A coupled level set and volume of fluid method for computing 3D and axisymmetric incompressible two phase flows. *Journal of Computational Physics* 2000; **162**:301–337.
22. Chopp D. Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics* 1993; **106**:77–91.
23. Tsitsiklis J. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control* 1995; **40**:1528–1538.
24. Sethian J. A fast marching method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America* 1996; **93**:1591–1595.
25. Sethian J. Fast marching methods. *SIAM Review* 1999; **41**:199–235.
26. Chopp D. Some improvements of the fast marching method. *SIAM Journal on Scientific Computing* 2001; **23**:230–244.
27. Adalsteinsson D, Sethian J. A fast level set method for propagating interfaces. *Journal of Computational Physics* 1995; **118**:269–277.
28. Adalsteinsson D, Sethian J. The fast construction of extension velocities in level set methods. *Journal of Computational Physics* 1999; **148**:2–22.
29. Sussman M, Fatemi E. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on Scientific Computing* 1999; **20**:1165–1191.
30. Sussman M, Smereka P, Osher S. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics* 1994; **114**:146–159.
31. Lew A, Buscaglia G. A discontinuous Galerkin-based immersed boundary method. *International Journal for Numerical Methods in Engineering* 2008; **76**:427–454.
32. Russo G, Smereka P. A remark on computing distance functions. *Journal of Computational Physics* 2000; **163**:51–67.
33. Min C, Gibou F. A second order accurate level set method on non-graded adaptive cartesian grids. *Journal of Computational Physics* 2007; **225**:300–321.
34. Mut F, Buscaglia G, Dari E. New mass-conserving algorithm for level set redistancing on unstructured meshes. *Journal of Applied Mechanics* 2006; **73**:1011–1016.
35. Aliabadi S, Tezduyar T. Stabilized-finite-element/interface-capturing technique for parallel computation of unsteady flows with interfaces. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:243–261.
36. Carrica P, Wilson R, Stern F. An unsteady single-phase level set method for viscous free surface flows. *International Journal for Numerical Methods in Fluids* 2007; **53**:229–256.
37. Zalesak S. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics* 1979; **31**:335–362.
38. LeVeque R. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis* 1996; **33**:627–665.
39. Gois J, Nakano A, Nonato L, Buscaglia G. Front tracking with moving-least-squares surfaces. *Journal of Computational Physics* 2008; **227**:9643–9669.
40. LeVeque R. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics* 1997; **131**:327–353.