Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXVII, págs. 13-32 (artículo completo)
Alberto Cardona, Mario Storti, Carlos Zuppa. (Eds.)
San Luis, Argentina, 10-13 Noviembre 2008

A MASS-PRESERVING GEOMETRY-BASED REINITIALIZATION METHOD FOR THE LEVEL SET FUNCTION

Roberto F. Ausas[a], **Enzo A. Dari**[a], **Gustavo C. Buscaglia**[b,a]

[a]*Centro Atómico Bariloche e Instituto Balseiro, 8400, Bariloche, Argentina, rfausas@gmail.com, darie@cab.cnea.gov.ar*
[b]*Instituto de Ciências Matemáticas e de Computação, Univ. de São Paulo, São Carlos, Brasil, e-mail: gustavo.buscaglia@gmail.com*

Keywords: Level Set Method, Reinitialization, Redistancing, Curvilinear Coordinates.

Abstract.

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 a state-of-the-art 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 state-of-the-art PDE-based reinitialization schemes. We also show that the algorithm is well–suited for the highly–streched curvilinear grids used in CFD simulations.

1 INTRODUCTION

The level set method, introduced by Osher and Sethian (1988), 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 in principle extracted from the level set function making it a very attractive 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.

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 (1988, 1989), and essentially non-oscillatory (ENO) schemes in space, based on the ideas firstly proposed by Harten and Osher (1987); Harten et al. (1987) to solve Hamilton-Jacobi type equations. In this area, the TVD-Runge-Kutta and Hamilton-Jacobi Weighted-ENO scheme developed in (Jiang and Peng, 2000) is considered to be state-of-the-art for solving the level set equation within the framework of eulerian methods (Losasso et al., 2006). In this case, curvilinear coordinates can be used to deal with complex geometries (see for instance Yue et al. (2003) and Carrica et al. (2006)). 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 (2005). 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 Di Pietro et al. (2006). In Marchandise et al. (2006) a discontinuous Galerkin method is proposed and compared with several other methods including the ENO/RK(3) scheme presented in Sussman et al. (1999) and the HJ-WENO(5)/RK(3) scheme used in Enright et al. (2005, 2002). Finally, semi-Lagrangian schemes, which can be implemented in very simple and efficient ways, are also used in level set methods (see Strain (1999a,b); Enright et al. (2005)).

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 in order 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 Enright et al. (2005, 2002) and Zhaorui et al. (2007). The other option uses the VOF method (see e.g. Hirt and Nichols (1981)), 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 (2003) and Sussman and Puckett (2000) for example.

In this article 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. 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, the level of discretization used and of course on the reinitialization algorithm itself. However, most level set methods assume that ϕ has to be reinitialized periodically for robustness of computations. We will thus assume that reinitialization *is* performed, and concentrate on *how* to perform it economically and accurately on general meshes.

A natural choice to reinitialize the level set function is the signed distance function to the interface. Several PDE-based methods have been devised for this purpose, which solve the so-called reinitialization equation as originally proposed by Sussman and Fatemi (1999); Sussman et al. (1994, 1999).

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 reduced (Lew and Buscaglia, 2008) 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 (Losasso et al., 2006). Though suitable implementations of PDE-based reinitialization methods exist for Cartesian grids, they are not well–suited for the highly–streched curvilinear grids used in turbulent flow simulations.

In this article we adapt the finite-element-based reinitialization scheme of Mut et al. (2006) 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.

In Section 2 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 by Yue et al. (2003). Since the proposed method is based on a piecewise-linear representation of the level set function, concerns may arise as to its accuracy. This section also contains a brief summary of a widely used PDE-based method, the HJWENO-RK scheme of Jiang and Peng (2000), that will be used for comparison.

In section 3, numerical experiments are shown in two and three spatial dimensions, including the rigid rotation of Zalesak's disk and the deformation of a ball under a swirling flow vortex, 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 coodinates 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 (Carrica et al., 2006). We present a numerical example 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 4.

2 NUMERICAL FORMULATION

2.1 Level Set Method

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

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

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 Yue et al. (2003), 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

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

where ϕ^n is the value of ϕ at the time level n, δt is the time step and $\mathcal{L}(\phi, t)$ is the spatial operator in equation (1), i.e.

$$\mathcal{L}(\phi, t) = \nabla \cdot (\vec{u}\phi). \tag{3}$$

Now, in order to obtain a fully discrete method, that 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 given by δx and δy. Then, the discrete form of (3) for the control volume (i, j) will be simply given by

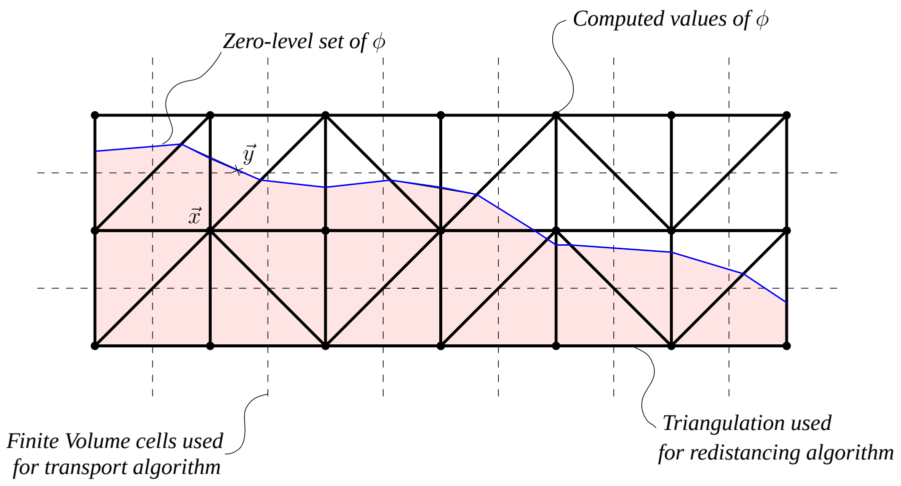
$$\mathcal{L}(\phi) = \frac{(u_x\phi)_{i+1/2,j} - (u_x\phi)_{i-1/2,j}}{\delta x} + \frac{(u_y\phi)_{i,j+1/2} - (u_y\phi)_{i,j-1/2}}{\delta y}. \tag{4}$$

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 centre of the control volume, from which, the cell face values of ϕ ($\phi_{i+1/2,j}$, $\phi_{i,j+1/2}$, ...) are built by using a third-order accurate ENO interpolation. Details for the construction of the corresponding stencils can be found in Shu and Osher (1989) or Yue et al. (2003).

2.2 Geometric mass-preserving redistancing scheme

The geometric mass-preserving reinitialization algorithm proposed here was originally devised to be used within the finite element framework (Mut et al., 2006), 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 1, 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.



Computed values of ϕ
Zero-level set of ϕ
$\vec{y}$
$\vec{x}$
Finite Volume cells used for transport algorithm
Triangulation used for redistancing algorithm

Figure 1: Schematic showing the finite volume discretization cells and the corresponding triangulation $\mathcal{T}_h$ for the redistancing algorithm.

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

$$d(\vec{x}) = \text{sign}\left[\phi_h(\vec{x})\right] \min_{\vec{y}\in\mathcal{S}_h} \parallel \vec{x}-\vec{y} \parallel , \tag{5}$$

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 2. 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 2), 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)***.

2.2.1 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.

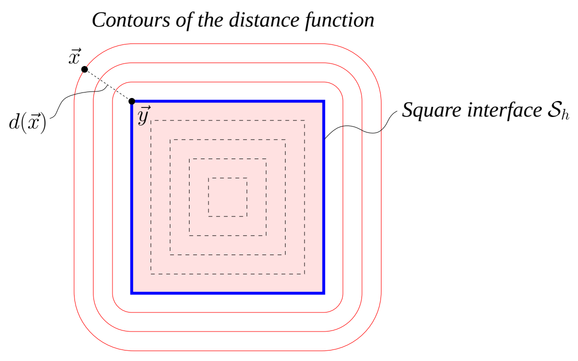

Contours of the distance function
$\vec{x}$
$d(\vec{x})$
$\vec{y}$
Square interface $\mathcal{S}_h$

Figure 2: 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 red lines) does not belong to the space V_h.

Step 1:: We begin by computing

$$\phi_h^*(\vec{X}) = d(\vec{X}) \qquad \forall\, \vec{X} \in \mathcal{P}, \tag{6}$$

so that the nodal values of the intermediate function ϕ_h^* coincide with the exact (signed) distance d. This computation is also divided into two different substeps:

Substep 1-A:: 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 $\mathcal{K}$. We start by computing $\tilde{d}$, such that, for node I we compute its distance only considering those parts of $\mathcal{S}_h$ that are inside simplices of which node I is a vertex.

Substep 1-B:: The simplices in $\mathcal{K}$ are swept until $\tilde{d}$ no longer changes. For each simplex, and for each node I (coordinates denoted by $\vec{X}_I$) of the simplex, $\tilde{d}$ is interpolated linearly on the opposite face F_I, using the current values at the nodes. Then, a tentative new value η_I of $\tilde{d}$ at node I is calculated as

$$\eta_I = \min_{\vec{x} \in F_I} \left[\tilde{d}(\vec{x}) + |\vec{X}_I - \vec{x}| \right] \tag{7}$$

and $\tilde{d}(\vec{X}_I)$ is updated to the value η_I *if the current value is greater than* η_I. When this process is finished, $\phi_h^*(\vec{X}_I)$ is updated to the value $\tilde{d}(\vec{X}_I)$ which at this point is in fact the exact distance d to $\mathcal{S}_h$. The procedure is illustrated in figure 3 and Table 1.

Equation (7) is the key operation in the computation of the distance. It is computed *exactly*. This is not difficult since d is here a linear function and the minimum is calculated over F_I, which is a simplex (a segment in 2D, a triangle or a quadrilateral in 3D). The possibilities of the minimum being attained in the interior of F_I or at its boundary have of course to be considered (in 3D, this latter case decomposes in turn into attaining the minimum either inside an edge or at a vertex).

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.

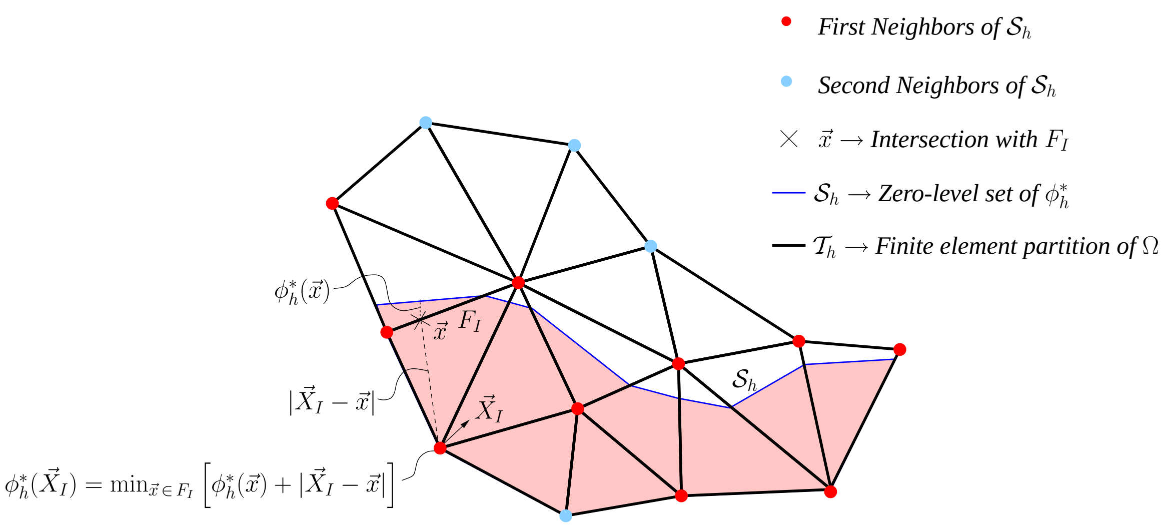

First Neighbors of $\mathcal{S}_h$
Second Neighbors of $\mathcal{S}_h$
$\vec{x} \rightarrow$ Intersection with F_I
$\mathcal{S}_h \rightarrow$ Zero-level set of ϕ_h^*
$\mathcal{T}_h \rightarrow$ Finite element partition of Ω
$\phi_h^*(\vec{x})$
$\vec{x}$
F_I
$|\vec{X}_I - \vec{x}|$
$\vec{X}_I$
$\mathcal{S}_h$
$\phi_h^*(\vec{X}_I) = \min_{\vec{x}\in F_I}\left[\phi_h^*(\vec{x}) + |\vec{X}_I - \vec{x}|\right]$

Figure 3: Schematic of **Substep 1-B** for the reinitialization of the nodes that are first neighbors of $\mathcal{S}_h$.

It is easy to check that the difference in the volumes defined by ϕ_h and ϕ_h^* is given by

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

where $\mathcal{H}$ is the Heaviside function ($\mathcal{H}(s) = 1$ if $s > 0$, $\mathcal{H}(s) = 0$ 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,

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

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 Eq. 11 is a nonlinear equation for η_K, which is solved independently for each K using a simple Regula Falsi procedure that converges in very few iterations.

The piecewise-constant function η_h computed in this way contains the information of 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 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

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

Table 1: **Substeps 1-A** and **1-B** for the Reinitialization of ***first Neighbors of*** $\mathcal{S}_h$. Nomenclature: $N^{\mathcal{P}}_{nod}$: number of nodes in $\mathcal{P}$, $N^{\mathcal{K}}_{el}$: number of elements (simplices) in $\mathcal{K}$, N_{npe}: number of nodes per single element (three for a triangle, four for a tetrahedron).

Substep 1-A::

```
• Set $\tilde{d}(\vec{X}_n) = +\infty$ for $n = 1, 2, ..., N^{\mathcal{P}}_{nod}$
do ($iel = 1, N^{\mathcal{K}}_{el}$)
    • Find $\mathcal{S}_{iel}$, the reconstruction of $\mathcal{S}_h$ inside element $iel$
    do ($I = 1, N_{npe}$)
        • Set $d_t$ as the distance from node $I$ to $\mathcal{S}_{iel}$
        if ($\tilde{d}(\vec{X}_I) > d_t$)
            • Set $\tilde{d}(\vec{X}_I) = d_t$
    end do
end do
```

Substep 1-B::

```
• Set $changes = 1$
do while ($changes == 1$)
    • Set $changes = 0$
    do ($iel = 1, N^{\mathcal{K}}_{el}$)
        do ($I = 1, N_{npe}$)
            • Find $F_I$, the opposite face of node $I$ in $iel$
            • Find $\eta_I$ s.t. $\eta_I = \min_{\vec{x}\in F_I}\left[\tilde{d}(\vec{x}) + |\vec{X}_I - \vec{x}|\right]$
            if ($\tilde{d}(\vec{X}_I) > \eta_I$) then
                • Set $\tilde{d}(\vec{X}_I) = \eta_I$
                • Set $changes = 1$
            end if
        end do
    end do
end do while
• Set $\phi^*_h(\vec{X}_n) = \tilde{d}(\vec{X}_n)$ for $n = 1, 2, ..., N^{\mathcal{P}}_{nod}$
```

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 Regula Falsi 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 Regula Falsi algorithms, which does not play any significant role since convergence to machine precision takes place quickly. Steps 2,3 and 4 are also explained in Table 2.

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 largest; correcting ϕ_h^* by a constant, as done by other authors (Aliabadi and Tezduyar, 2000), corresponds to taking $\xi_h = 1$ and unphysically distributes the correction uniformly over the interface simplices.

Table 2: **Steps 2,3** and **4** for the Reinitialization of ***first neighbors of*** $\mathcal{S}_h$, computation of the mass correction. N_I is the number of simplices in $\mathcal{K}$ that contain node I.

Step 2:: Find η_h, a piecewise constant function

```
do ($K = 1, N_{el}^{\mathcal{K}}$)
    • Set $\delta V_K = \Delta V_K(\phi_h, \phi_h^*)$
    do while ($|\delta V_K| > 10^{-15}$)
        • Find $\mathcal{S}_K$, the reconstruction of $\mathcal{S}_h$ in $K$ using $\phi_h^* + \eta_K$
        • Set $\eta_K = -\delta V_K / size(\mathcal{S}_K)$
        • 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 ($I = 1, N_{nod}^{\mathcal{P}}$)
    • 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 $C = C^{(i)}$
```

2.2.2 Reinitialization of the rest of the mesh

As discussed by Carrica et al. (2007), 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 next section. We however adopt the geometric scheme introduced by (Mut et al., 2006), 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 $\tilde{\phi}_h$ just 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

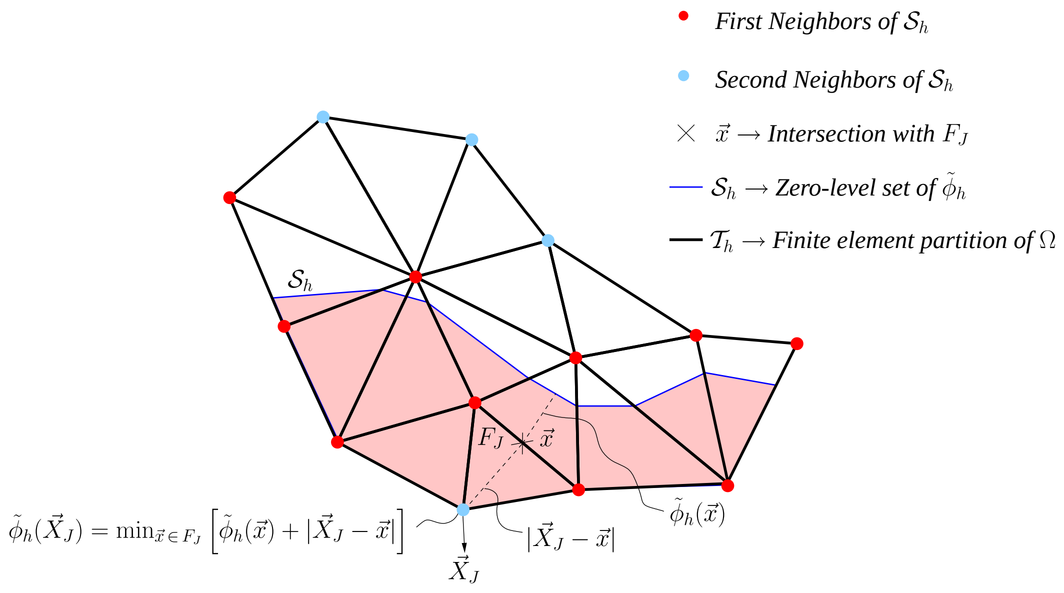
$$\tilde{\phi}_h(\vec{X}_I) = \min_{J \in C_I} \left[\tilde{\phi}_h(\vec{X}_J) + |\vec{X}_I - \vec{X}_J| \right].$$

In the process of initializing $\tilde{\phi}_h$ with this option, the elements can be ordered so as to render the algorithm more effective. Also, 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): This is the same procedure explained in **Substep 1-B**: the simplices in the mesh, excepting 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 $\vec{X}_J$), $\tilde{\phi}_h$ 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

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

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 also illustrated in figure 4.



● First Neighbors of $\mathcal{S}_h$
● Second Neighbors of $\mathcal{S}_h$
$\times$ $\vec{x} \rightarrow$ Intersection with F_J
$\mathcal{S}_h \rightarrow$ Zero-level set of $\tilde{\phi}_h$
$\mathcal{T}_h \rightarrow$ Finite element partition of Ω
$\mathcal{S}_h$
F_J
$\vec{x}$
$\tilde{\phi}_h(\vec{x})$
$|\vec{X}_J - \vec{x}|$
$\vec{X}_J$
$\tilde{\phi}_h(\vec{X}_J) = \min_{\vec{x} \in F_J} \left[\tilde{\phi}_h(\vec{x}) + |\vec{X}_J - \vec{x}| \right]$

Figure 4: Schematic of **Step 2** for the reinitialization of the rest of mesh.

2.3 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, 2000).

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 (5) (with ϕ_h replaced by ϕ^0 and $\mathcal{S}_h$ replaced by $\mathcal{S}$)
The property $\|\nabla d\| = 1$ motivates the method firstly proposed by (Sussman and Fatemi, 1999), in which the following hyperbolic partial differential equation is solved

$$\begin{gathered}\frac{\partial \tilde{\phi}}{\partial \tau} + \text{sign}(\phi^0)\left(\| \nabla \tilde{\phi} \| -1\right) = 0 \qquad \text{in } \Omega, \\ \tilde{\phi}(\vec{x}, 0) = \phi^0(\vec{x}),\end{gathered} \tag{16}$$

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 a RK-HJWENO (weighted essentially non-oscillatory) scheme (see Jiang and Peng (2000)) which can be considered to be state-of-the-art for solving this type of equations. The approach is very similar to that presented in the previous section for the discretization of the level set equation (1). First, the semidiscrete form of equation (16) for node (i, j), that for simplicity is presented again in two spatial dimensions reads

$$\frac{\partial \tilde{\phi}_{i,j}}{\partial \tau} = -\hat{H}(x_i, y_j, \tilde{\phi}_{i,j}, \tilde{\phi}^+_{x,i,j}, \tilde{\phi}^-_{x,i,j}, \tilde{\phi}^+_{y,i,j}, \tilde{\phi}^-_{y,i,j}), \tag{17}$$

where $\hat{H}$ is the discrete form of the spatial operator $\text{sign}(\phi_0)\left(\| \nabla \tilde{\phi} \| -1\right)$. Then, for the construction of $\tilde{\phi}^{\pm}_{x,i,j}$ and $\tilde{\phi}^{\pm}_{y,i,j}$, that 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 exactly Jiang and Peng (2000). Finally, we use a fourth order Runge-Kutta method to explicitly advance the system of ODE's given in (17), which reads

$$\begin{aligned}\tilde{\phi}^{(1)} &= \tilde{\phi}^n - \frac{1}{2}\delta\tau\ \hat{H}(\tilde{\phi}^n), \\ \tilde{\phi}^{(2)} &= \tilde{\phi}^n - \frac{1}{2}\delta\tau\ \hat{H}(\tilde{\phi}^{(1)}), \\ \tilde{\phi}^{(3)} &= \tilde{\phi}^n - \delta\tau\ \hat{H}(\tilde{\phi}^{(2)}), \\ \tilde{\phi}^{n+1} &= -\frac{1}{3}\tilde{\phi}^n + \frac{1}{3}\tilde{\phi}^{(1)} + \frac{2}{3}\tilde{\phi}^{(2)} + \frac{1}{3}\tilde{\phi}^{(3)} - \frac{1}{6}\delta\tau\ \hat{H}(\tilde{\phi}^{(3)}).\end{aligned}$$

For all the numerical experiments we present, the time step $\delta\tau$ will be taken as $\delta x/2$ and the reinitialization will be carried out as long as the quantity $\left|\| \nabla \tilde{\phi} \| -1\right|$ remains greater than a numerical tolerance of 10^{-5}.

3 RESULTS

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

$$e_m = \max_t |V(\phi_c) - V(\phi_e)|, \tag{18}$$
$$e_p = \max_t D(\mathcal{S}_c(t), \mathcal{S}_e(t)) = \max_t \max_{x_e \in \mathcal{S}_e} \min_{x_c \in \mathcal{S}_c} ||\vec{x}_c - \vec{x}_e||, \tag{19}$$

with the subindex c denoting the computed result and the subindex e the exact one, and where $V(\phi)$ is computed according to

$$V(\phi) = \int_\Omega \mathcal{H}(\phi(\vec{x}))\, d\vec{x}. \tag{20}$$

The first measure of error e_m is the classical "mass error". The second measure e_p provides information on the position of the computed interface with respect to the exact one and will be refered to as the "position error". It should be pointed out that linear interpolation will be used to evaluate e_m and e_p. 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. Finally, we must mention that the reinitialization procedure will be applied every 10 time steps for all the simulations to be presented.

3.1 Numerical Experiments in 2D

Two examples will be presented in the two dimensional case: the Zalesak's problem (Zalesak (1979)) and the stretching of a circle under a deformation vortex (LeVeque (1996)).

3.1.1 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 lenght of 0.25 in a unit square computational domain. The disk is convected by the following velocity field

$$u_x = \frac{\pi}{3.14}(0.5 - y), \qquad u_y = \frac{\pi}{3.14}(x - 0.5), \tag{21}$$

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 figure 5 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 mantain the same Courant number. As it can be seen the geometric scheme performs similarly to the PDE-based one when the grid resolution is good enough ($h = 1/256$ and $h = 1/512$), while the former outperforms the latter when the grid resolution is poor. In Figure 6 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 (thin blue line) and on the right the results for the Geometry-Based algorithm (thin red line). Finally, in Table 3 we present the two measures of error for both algorithms and for the different grids considered. It should be noted that the mass error of the PDE-based algorithm is smaller than that of the geometric algorithm when $h = 1/128, 1/256$ and $1/512$, which results from a compensation of the mass gain near the top of the slot with the mass loss near the corners at the bottom of the slot.

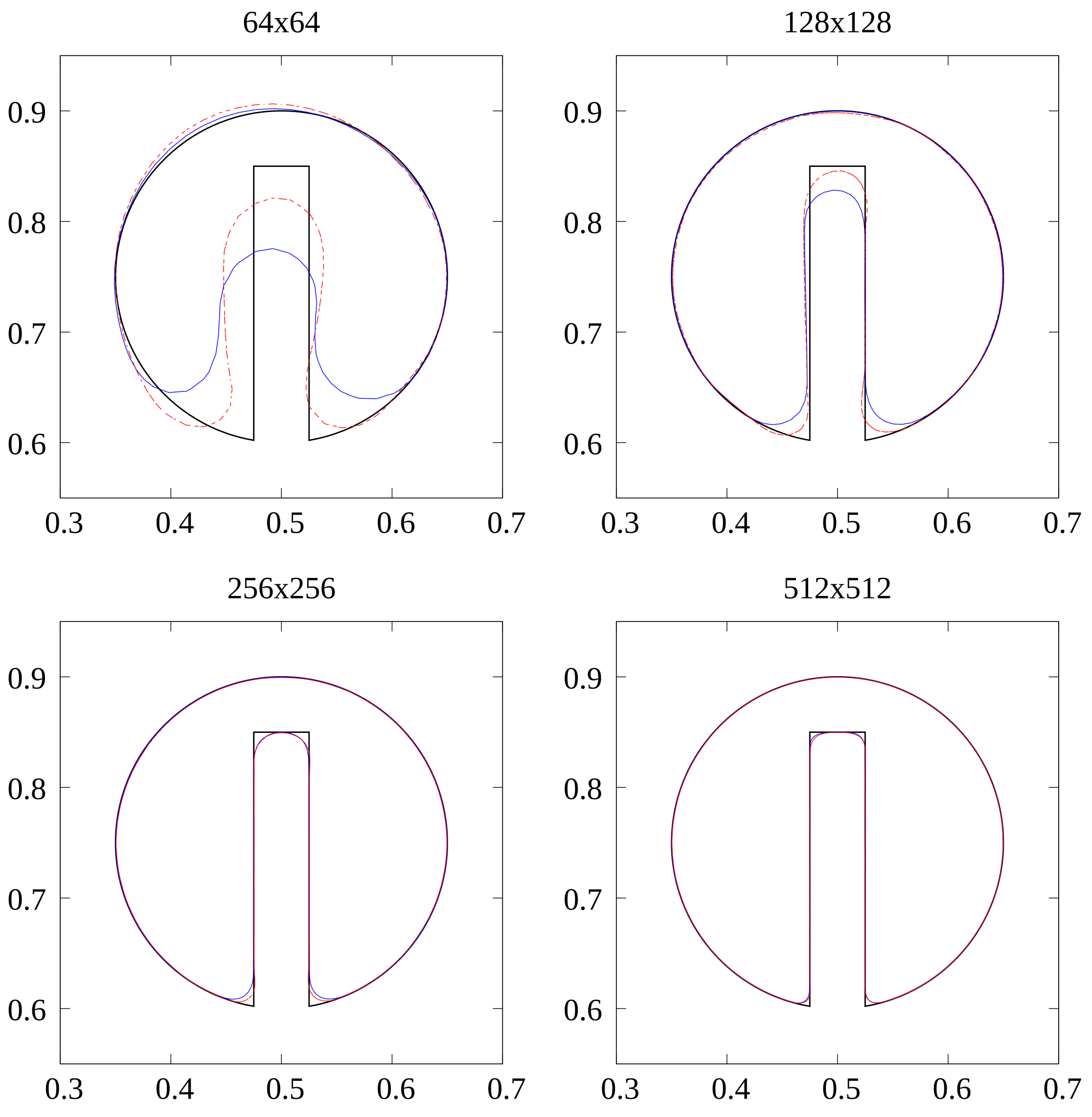


Figure 5: Final stage of the Zalesak's disk after one revolution for different grids. The thick black line corresponds to the exact solution, the thin blue line to the PDE-based and the dashed red line to the geometry based algorithm.

3.1.2 Swirling flow vortex

The initial data consists of a disk centred 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 convected by the following time dependent divergence-free velocity field

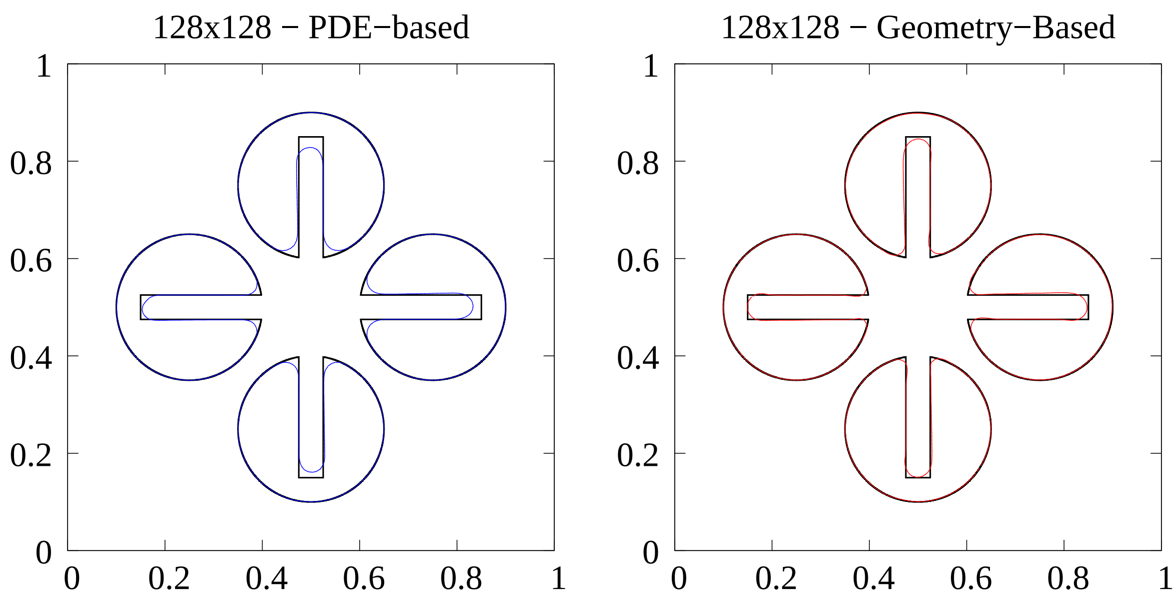


Figure 6: Evolution of the Zalesak's disk using the PDE-based (left) and the Geometry-based (right) redistancing procedures. The thick black line corresponds to the exact solution.

Table 3: Measures of error for the Zalesak's disk after one revolution.

	$e_m[\%]$		e_p	
Mesh	***PDE-based***	***GEO-based***	***PDE-based***	***GEO-based***
64×64	7.790	4.564	0.0709	0.0352
128×128	0.940	2.065	0.0291	0.0137
256×256	0.210	0.470	0.0126	0.0103
512×512	0.014	0.266	0.0065	0.0070

$$\begin{aligned} u_x &= -\sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ u_y &= \sin(2\pi x)\sin^2(\pi y)\cos(\pi t/T). \end{aligned} \tag{22}$$

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 7 we compare the interface at $t = T/2$ (maximum deformation) and $t = T$ (final time) for all the grids previouly 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 quite similar by when the grid resolution is good. Actually, for the case with $h = 1/512$ the difference cannot be seen with the naked eye. For this reason, as done before, in Table 4, we present the different measures of error. In this case, we can see that the Geometric mass-preserving scheme has a better performance than the PDE-based scheme.

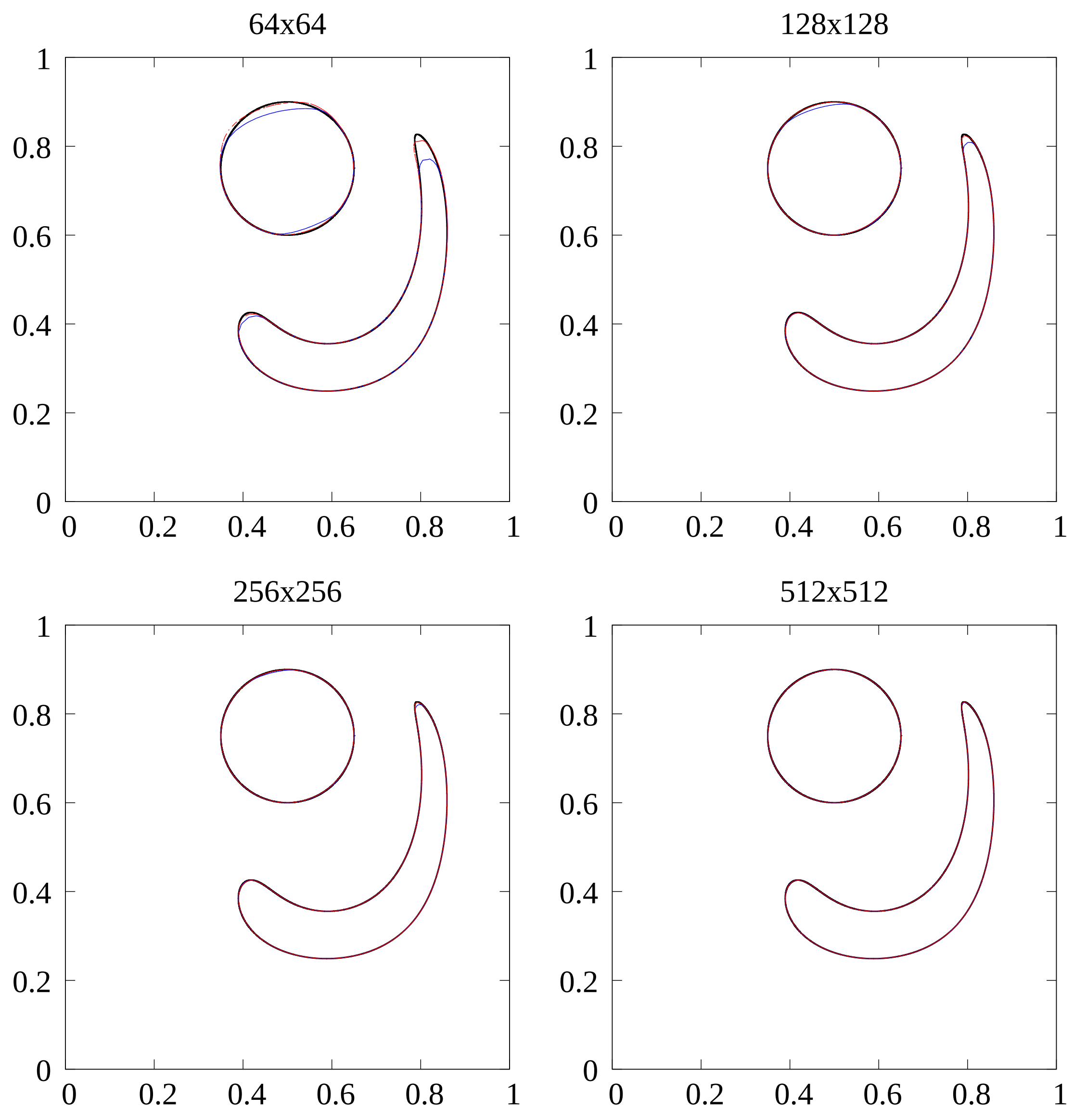


Figure 7: Intermediate ($t = T/2$) and final stage ($t = T$) of the disk under a swirling flow vortex with reversal period $T = 2$ for different grid resolutions. The thick black line corresponds to the exact solution, the thin blue line to the PDE-based and the dashed red line to the geometry based algorithm.

3.2 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 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.

Table 4: Measures of error for the streatching of a disk under a swirling flow vortex with reversal period $T = 2$.

	$e_m[\%]$		e_p	
Mesh	***PDE-based***	***GEO-based***	***PDE-based***	***GEO-based***
64×64	5.172	0.797	0.0641	0.0150
128×128	1.624	0.417	0.0272	0.0035
256×256	0.400	0.256	0.0100	0.0017
512×512	0.081	0.133	0.0021	0.0006

3.2.1 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

$$\begin{aligned} u_x &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \\ u_y &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \\ u_z &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T), \end{aligned} \tag{23}$$

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 5 we present the two measures of error, in this case just 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 8 we plot the level set at different times for both algorithms for the case with $h = 1/128$. In the top (red colour) are the results for the Geometry-based redistancing and in the bottom (blue colour) 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.

Table 5: Measures of error for the deformation of a sphere under a three dimensional vortex. Cartesian coordinates.

	$e_m[\%]$		e_p	
Mesh	***PDE-based***	***GEO-based***	***PDE-based***	***GEO-based***
$64 \times 64 \times 64$	66.40	2.432	0.0673	0.0261
$128 \times 128 \times 128$	11.95	1.593	0.0355	0.0049

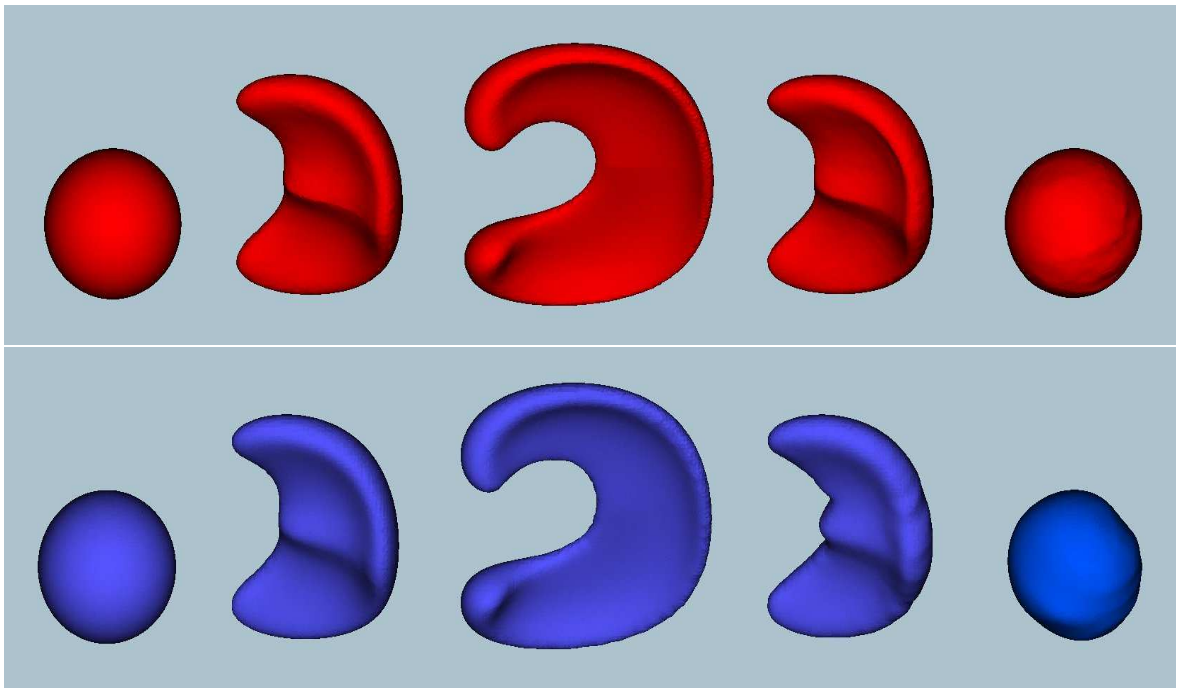


Figure 8: Evolution of a sphere under a three dimensional deformation vortex. Comparison of the Geometry-based redistancing scheme (top-red) with the PDE-based redistancing scheme (bottom-blue). Cartesian grid $128 \times 128 \times 128$.

3.2.2 Sphere approaching a bump - Curvilinear coordinates

For this last example, the initial condition corresponds to a sphere centred 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 LeVeque (1997))

$$\begin{aligned} x(\xi,\eta,\zeta) &= \xi, \\ y(\xi,\eta,\zeta) &= B(\xi) + \eta\,(1 - B(\xi))\,, \\ z(\xi,\eta,\zeta) &= \zeta, \end{aligned} \tag{24}$$

where the function B is given by

$$B(\xi) = \frac{1}{2} e^{-50(\xi-0.5)^2}, \tag{25}$$

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

$$\begin{aligned} u_x &= \frac{1}{1-B(x)}, \\ u_y &= \frac{B'(x)(1-y)}{(1-B(x))^2}, \\ u_z &= 0. \end{aligned} \tag{26}$$

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

Results are shown in figure 9, where a detail of the curvilinear grid can be observed. In this case, the mass change e_m was 4.438% and the value of $e_p = 0.0370$. 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.

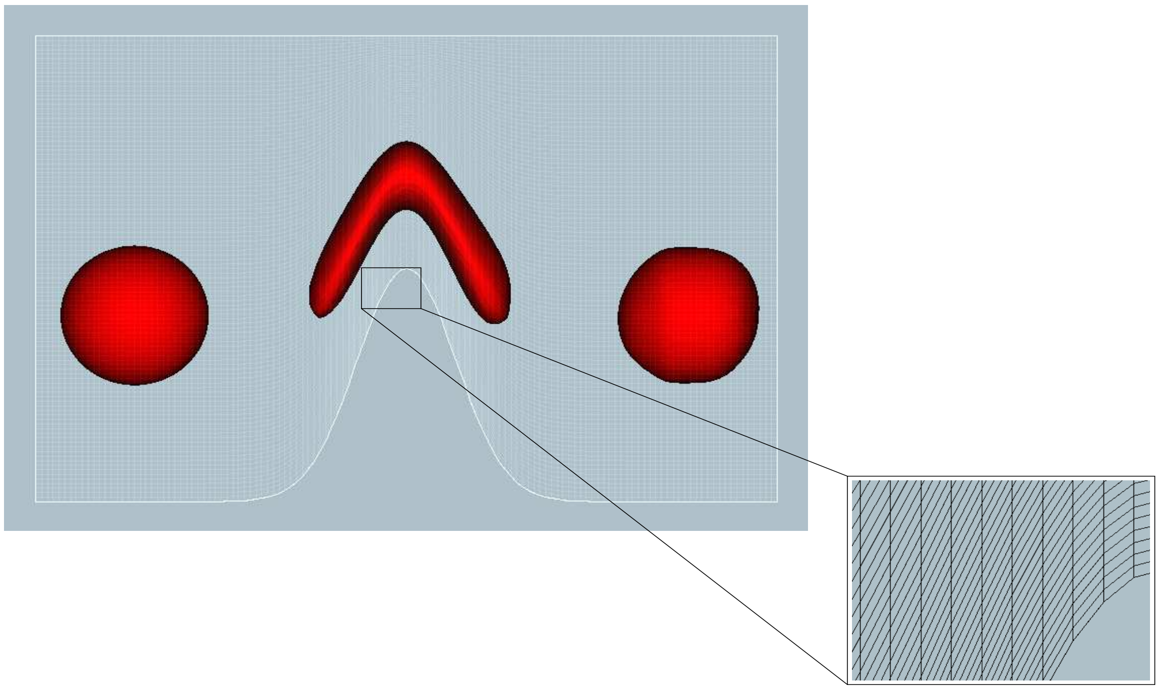


Figure 9: Evolution of the sphere approaching a bump using the geometric mass-preserving redistancing scheme. Curvilinear grid of $128 \times 102 \times 62$ cells.

4 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 robustness, since it lacks of adjustable parameters, which is an important difference as compared to other available methods.

As a main feature, the scheme is designed to preserve the mass (or the volume) limited by the zero level set by means of localized mass corrections, once the triangulation of the computational domain is provided. Of course, this is done at the price of having to detect the level set by means of linear interpolation which is unnecesary in other methods.

In the numerical tests we have presented, using Cartesian coordinates in two and three spatial dimensions, we have observed in general a better performance of the geometric mass-preserving redistancing scheme with respect to the PDE-based method used for comparison. This was illustrated qualitatively by means of plots of the level set and quantitatively be means of computing relevant measures of error for level set methods.

We have also tested the geometric mass-preserving algorithm using curvilinear grids with appreciable distortion and we have observed a good performance of the scheme.

5 ACKNOWLEDGMENTS

The authors thank to CONICET (Argentina). GB also thanks to CNPq (Brazil) and Fapesp.

REFERENCES

Aliabadi S. and Tezduyar T. Stabilized-finite-element/interface-capturing technique for parallel computation of unsteady flows with interfaces. *Comput. Methods Appl. Mech. Engrg.*, 190:243–261, 2000.

Carrica P., Wilson R., Noack R., Xing T., Kandasamy M., Shao J., Sakamoto N., and 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.

Carrica P., Wilson R., and Stern F. An unsteady single–phase level set method for viscous free surface flows. *Int. J. Numer. Meth. Fluids*, 53:229–256, 2007.

Di Pietro D., Lo Forte S., and Parolini N. Mass preserving finite element implementations of the level set method. *Applied Numerical Mathematics*, 56:1179–1195, 2006.

Enright D., Fedkiw R., Ferziger J., and Mitchell I. A hybrid particle level set method for improved interface capturing. *Computers and Structures*, 83:479–490, 2002.

Enright D., Losasso F., and Fedkiw R. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83:479–490, 2005.

Harten A., Engquist B., Osher S., and Chakravarthy S. Uniformly high-order accurate essentially non-oscillatory schemes III. *Journal of Computational Physics*, 71:231–303, 1987.

Harten A. and Osher S. Uniformly high-order accurate essentially non-oscillatory schemes I. *SIAM J. Numer. Anal*, 24:279–309, 1987.

Hirt C. and Nichols H. Volume of fluid (VOF) methods for the dynamics of free boundaries. *Aplied Numerical Mathematics*, 39:201–225, 1981.

Jiang G.S. and Peng D. Weighted eno schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2144, 2000.

LeVeque R. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal of Numerical Analysis*, 33:627–665, 1996.

LeVeque R. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics*, 131:327–353, 1997.

Lew A. and Buscaglia G. A Discontinuous–Galerkin–based immersed finite element method. *International Journal for Numerical Methods in Engineering. In press*, 2008.

Losasso F., Fedkiw R., and Osher S. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and fluids*, 35:995–1010, 2006.

Marchandise E., Remacle J.F., and Chevaugeon N. A quadrature-free discontinuous Galerkin method for the level set equation. *J. Comp. Phys.*, 212:338–357, 2006.

Mut F., Buscaglia G., and Dari E. New mass-conserving algorithm for level set redistancing on unstructured meshes. *Journal of Applied Mechanics*, 73:1011–1016, 2006.

Olsson E. and Kreiss G. A conservative level set method for two phase flow. *Journal of Computational Physics*, 210:225–246, 2005.

Osher S. and Sethian J. Front propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

Shu C. and Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

Shu C. and Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes II (two). *J. Comput. Phys.*, 83:32–78, 1989.

Strain J. Semi-Lagrangian methods for level set equations. *J. Comp. Phys.*, 151:498–533, 1999a.

Strain J. Tree methods for moving interfaces. *J. Comp. Phys.*, 151:616–648, 1999b.

Sussman M. A second order coupled level set and volume of fluid method for computing growth and collapse of vapor bubbles. *J. Comp. Phys.*, 187:110–136, 2003.

Sussman M. and Fatemi E. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM J. Sci. Comput.*, 20:1165–1191, 1999.

Sussman M., Fatemi E., Smereka P., and Osher S. An improved level set method for incompressible two-fluid flows. *Computers and Fluids*, 20:1165–1191, 1999.

Sussman M. and Puckett E. A coupled level set and volume of fluid method for computing 3D and axisymmetric incompressible two phase flows. *J. Comp. Phys.*, 162:301–337, 2000.

Sussman M., Smereka P., and Osher S. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

Yue W., Lin C.L., and Patel V. Numerical simulation of unsteady multidimensional free surface motions by level set method. *International Journal for Numerical Methods in Fluids*, 42:853–884, 2003.

Zalesak S. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 335-362:31, 1979.

Zhaorui L., Farhad A., and 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, (in press) DOI:10.1002/fld.1621*, 2007.