Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A self-adaptive oriented particles Level-Set method for tracking interfaces

S. Ianniello *, A. Di Mascio

INSEAN, Italian Ship Model Basin, Rome, Italy

ARTICLE INFO

Article history:
Received 12 March 2009
Received in revised form 28 July 2009
Accepted 16 October 2009
Available online 28 October 2009

Keywords:
Tracking interface
Oriented particles
Level-Set
Numerical method

ABSTRACT

A new method for tracking evolving interfaces by lagrangian particles in conjunction with a Level-Set approach is introduced. This numerical technique is based on the use of time evolution equations for fundamental vector and tensor quantities defined on the front and represents a new and convenient way to couple the advantages of the Eulerian description given by a Level-Set function ϕ to the use of Lagrangian massless particles. The term *oriented* points out that the information advected by the particles not only concern the spatial location, but also the local (outward) normal vector $\mathbf{n}$ to the interface Γ and the second fundamental tensor (the shape operator) $\nabla\mathbf{n}$. The particles are exactly located upon Γ and provide all the requested information for tracking the interface on their own. In addition, a self-adaptive mechanism suitably modifies, at each time step, the markers distribution in the numerical domain: each particle behaves both as a potential *seeder* of new markers on Γ (so as to guarantee an accurate reconstruction of the interface) and a *de-seeder* (to avoid any useless gathering of markers and to limit the computational effort). The algorithm is conceived to avoid any transport equation for ϕ and to confine the Level-Set function to the role of a mere post-processing tool; thus, all the numerical diffusion problems usually affecting the Level-Set methodology are removed. The method has been tested both on 2D and 3D configurations; it carries out a fast reconstruction of the interface and its accuracy is only limited by the spatial resolution of the mesh.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

The tracking of a moving interface Γ represents a key-point in the numerical simulation of free-surface and multiphase flows appearing in many mathematical and engineering problems. The interest in the topic is witnessed by the uncountable papers and books published on this issue and the wide variety of involved research fields, covering aerospace, naval and steel industry, many environmental, geophysical and medical applications, wide branches of computer graphics and digital image processing. Generally speaking, the problem can be treated by Lagrangian and Eulerian methods. In the first case, the interface evolution is followed by some marker advected with the front velocity; in the second one, a new variable (a Level-Set function, a volume fraction, the density) is introduced in the model to identify different flow domains with respect to a fixed grid. In between lie mixed Eulerian–Lagrangian approaches, where the grid moves to fit the actual front position; these approaches have the advantage of providing an accurate description of the interface motion, thus making easier to impose of the surface tension forces and, in general, the boundary conditions on Γ. Nevertheless, they are used only when the displacements of the interface are small, because the related grid distortion can cause an intolerable loss of accuracy. Of course, wave breaking or large topological changes cannot be handled by methods of this kind.

* Corresponding author. Tel.: +39 0650299328; fax: +39 065070619.
E-mail address: s.ianniello@insean.it (S. Ianniello).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.10.034

In the context of Lagrangian approaches we certainly have to include the particles mesh-free methods, introduced in the sixties. There, the interface motion is tracked by massless markers moved by the velocity field independently of any mesh. The markers can be placed in the whole computational domain, as in the pioneering marker and cell (MAC) method [1], or concentrated just upon Γ to exactly track the interface time evolution with a smaller computational effort (see, for example, [2]). The particles are often coupled to an Eulerian method in order to achieve a more accurate description of both motion and the relevant interface's geometrical characteristics. As mentioned above, in standard Eulerian approaches an additional unknown is introduced to represent the interface and an additional equation to model its time evolution. In the volume of fluid (VOF) method, introduced by DeBar [3] and subsequently improved by Noh and Woodward [4], Hirt and Nichols [5], Lafaurie et al. [6] and Gueyffier et al. [7], this unknown corresponds to the mass (volume) fraction of one of the fluids. The new variable characterizes each cell of the computational domain and can be equal to 0, 1 or some intermediate value at the grid cells containing the first fluid, the second fluid or the interface, respectively. A transport equation is solved for the volume fraction and the reconstruction of Γ is carried out by computing the mass fluxes among adjacent cells with a procedure of geometrical nature. For this reason, the method exhibits very good mass-preserving properties and a satisfactory tracking of the interface, especially when the piecewise linear interpolation construction (PLIC) [8] is adopted. Nevertheless, the interface appears as a discontinuous surface at the cells boundaries, so that the computation of the curvature (required to evaluate the surface tension forces) is not simple and still represents a research issue [9]. Moreover, the use of a straight line to represent Γ inevitably flattens the regions characterized by high curvature, so that the interface can be broken when its thickness is comparable to the grid spacing. As a result, filaments are artificially divided and coalesce into some blobs of numerical nature. Classical reviews of the volume tracking methods can be found, for instance, in [10,11].

In Level-Set methods [12] the new unknown is represented by a signed distance function ϕ: the sign of the function determines the mutual position of the fluid domains and the interface is represented by the zero level of the function itself. A transport equation governs the time evolution of the Level-Set function field, but in this case no mechanism enforces mass conservation. Thus, such a physical, fundamental requirement is somehow demanded to the accuracy of the numerical scheme adopted to solve the transport equation. For this reason, high order ENO (essentially non-oscillatory) and WENO (weighted essentially non-oscillatory) schemes are the most used algorithms in the Level-Set community. The method provides a continuous representation of Γ which accounts automatically for possible topological changes; furthermore, the knowledge of the function ϕ enables the estimation of the most important geometrical features of the interface, as the local normal vector and the curvature. To this aim, however, the Level-Set function must be often reinitialized through a further equation, since its property of being a distance function is lost during the advection [13]. It's worth noting that also the reinitialization may provide an additional numerical diffusion, so that different reinitialization techniques have been developed to improve mass conservation [14–16].

In the last decades, a lot of numerical approaches were developed in the attempt of suitably combining the advantages of both Lagrangian and Eulerian techniques to achieve an optimum interface reconstruction algorithm. Just to mention some examples, in [17] a multi-fluid flow is treated by solving the Navier–Stokes equations on a stationary finite difference grid, while the sharp interface separating the fluids is tracked by an additional moving interface mesh. A combination of Lagrangian markers to the VOF method is realized in [18,19], while in the CLSVOF (coupled Level-Set and volume of fluid) method a mixed Eulerian scheme couples the good mass conservation properties of VOF with the accurate surface curvature reconstruction of the function ϕ [20–22]. In the Point-Set method [23], unconnected markers are used to determine an indicator function through a B-Spline interpolation, which somehow couples certain features of both the VOF method and the Level-Set approach. In the immersed boundary (IB) methods, originally developed for cardiovascular flows [24], the incompressible/compressible flow equations are solved in the whole domain in Eulerian variables, while a Lagrangian viewpoint is adopted for the simulation of an embedded flexible structure, here representing the interface. A review of the immersed boundary methods is reported in [25].

A further approach is the smoothed particle hydrodynamics (SPH) method [26,27], where the markers are put in the flow with mass and velocity and exhibit a mutual influence at a prescribed distance from each others. Also this method can handle large interface deformations and topological changes automatically and guarantees the mass conservation, since the particles themselves represent mass (although the volume is not preserved). One drawback over grid-based techniques is the need for a large number of markers and the requirement of some graphic approach to provide a continuous, renderable free surface geometry. An exhaustive review of the SPH method can be found in [28].

In the framework of the possible combinations between different numerical techniques, a significant role is played by the particle Level-Set (PLS) method developed in [30] and subsequently refined in [31]. Here, Lagrangian massless markers are randomly located near the interface, passively advected by the flow and locally used to correct the Level-Set function in poorly resolved regions, where the loss of mass occurs. The PLS method proved to be really effective in the improvement of the Γ reconstruction process provided by a standard Level-Set procedure; it is able to carry out very realistic simulations of fluids, so as becoming the reference approach for a large part of the graphics community. Also in the PLS method, however, the Level-Set function plays the primary role in tracking the interface. Despite the proved efficiency of the particles, the Γ time evolution is substantially governed by the transport equation for ϕ and the massless markers are used to locally correct the errors arising from the numerical diffusion, mainly related to the computation of the spatial term $\nabla\phi$.

The interface tracking procedure proposed in this paper takes the PLS method as a starting-point and inverts the aforementioned roles played by the particles and the Level-Set function by linking the interface geometrical parameters (the normal and tangent vectors and the second fundamental tensor) to the particles and by making these quantities to evolve in time through some well-defined evolution equations. The Lagrangian markers placed on Γ become the leading mechanism in tracking the interface and are used to rebuild the front very accurately by a high order Runge–Kutta integration. In other words, the particles dictate the time evolution of Γ, whereas the function ϕ becomes a post-processing tool, used to turn the interface into a continuous function, to be coupled to other field equations (e.g. the Navier–Stokes equations). Unlike the PLS method, it is the transport equation of ϕ which is now used to correct possible numerical inaccuracies and make the algorithm more robust; this equation, however, does not govern the process so that no numerical diffusion (related to the computation of the term $\nabla\phi$) can affect the solution. An effective seeding algorithm has also been developed to manage, at each step, the particles distribution and check the accuracy of the Γ reconstruction process. Based on a Taylor expansion series of the function ϕ, the procedure uses the updated geometrical quantities associated to the particles to refine their distribution in presence of a large stretching and deformation of Γ. The method is fast and quite easy to implement. It has been tested on typical two-dimensional problems and on a very severe three-dimensional test-case (by exploiting the analytical knowledge of the velocity field) and has always provided an impressively accurate interface reconstruction.

The structure of the paper is the following. Section 2 is devoted to a summary of the Level-Set methodology and the corresponding numerical problems. The results obtained through a PLS-based code are presented, in order to show some features of the method and to recall some typical test cases (like the vortex in a box) studied in the following. In Section 3, some issues reported in [30] on the capability of surface-markers to correctly trace a 2D shrinking square are discussed. The oriented particles and their coupling with the Level-Set function are introduced in Section 4, by deriving the evolution equation for the normal vector to Γ, by describing how the function ϕ can be estimated through the updated particles location and by pointing out the most important features of the procedure through the numerical results achieved on the vortex in a box and the so called Zalesak's disk problem. The evolution equations for the second fundamental tensor $\nabla\mathbf{n}$ and any tangent vector $\mathbf{t}$ to Γ are discussed in Section 5, by pointing out the relation of $\mathbf{t}$ to the interface stretching. Section 6 points out the intrinsic drawbacks of a tracking interface method based on the sole use of Lagrangian markers and deals with the well-known, critical 3D Enright test-case to show how the aforementioned weak-points can affect the numerical results. Within this context, the use of the ϕ transport equation and the curvature evolution equation are also treated. Finally, Section 7 introduces the proposed self-adaptive oriented particle Level-Set method, by equipping the algorithm with an automatic procedure, able to manage undesired spatial clustering and/or depletion of markers. Section 8 shows some results concerning the simulations of a shrinking and expanding cube (where interface merging occurs) and Section 9 draws conclusions.

2. Summary of Level-Set methods

As is well known, the Level-Set method is based on the use of a level function ϕ characterized by the following properties

$$\begin{aligned} &\phi(\mathbf{x},t) > 0 \quad \mathbf{x}\in\Omega \\ &\phi(\mathbf{x},t) \leqslant 0 \quad \mathbf{x}\notin\Omega \end{aligned}$$

where Ω represents an open region in R^n and the ϕ zero level represents the interface $\Gamma\subset R^{n-1}$ whose motion we are interested in. This motion is determined by a velocity field $\mathbf{u}$, which gives rise to the following transport equation for ϕ

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0 \tag{1}$$

The level function ϕ is usually initialized as a signed distance function and enables the computation of some fundamental geometrical quantities for Γ, as the unit normal vector

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{2}$$

and the curvature

$$\kappa = \nabla\cdot\mathbf{n} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{3}$$

Nevertheless, due to the time evolution, the role of a signed distance function is immediately lost by ϕ, and in order to minimize numerical errors it is convenient to reinitialize the function at each time step by solving the equation

$$\frac{\partial\phi}{\partial\tau} + \mathcal{S}(\phi_0)(|\nabla\phi| - 1) = 0 \tag{4}$$

where τ represents a fictitious time and $\mathcal{S}(\phi_0)$ is the sign function, often smeared in numerical applications [32]. It's worth mentioning that from a numerical standpoint the reinitialization process has to be coded carefully, since it should not alter the ϕ zero level representing the interface. The solution of Eq. (1) strongly depends on the adopted numerical scheme. As is well-known, the use of simple first-order finite difference schemes (both in time and in space) suffers a notable numerical diffusion which appears as a loss of mass. This behaviour can even get worse by the reinitialization procedure, which always represents an approximated process and should be avoided as much as possible [33] (for example, through the so-called extension velocities technique [34]). Diffusion problems can be alleviated by using more suitable high-order finite difference schemes. As suggested in [29], a 5th order WENO scheme for the computation of the spatial term $\nabla\phi$ can be successfully combined with a 3rd order TVD Runge Kutta procedure for time integration. Nonetheless, even these schemes can be not sufficient to trace Γ in a accurate way, especially when the interface exhibits certain critical configurations characterized by thin filaments or some corner points. For this reason, typical test cases are used to check the ability of any interface tracking algorithm, and two problems have become very popular in the literature for 2D problems: the Zalesak's disk and the vortex in a box test-cases. The first test (originally introduced in [35]) refers to a rotating circular interface with a rectangular notch, immersed in a velocity field corresponding to a solid body rotation: although in the exact solution the disk evolves maintaining its shape unaltered, in numerical simulations the region of the sharp corners of the notch rapidly deteriorates because of diffusion errors. In the second problem, a circular interface undergoes a strong deformation under the action of a vorticity field; in this case, Γ exhibits a progressive thinning which severely tests the algorithm capability in resolving thin filaments and dealing with loss of mass.

In the attempt of achieving a further improvement on diffusion problems, Enright et al. proposed the particle Level-Set (PLS) method by randomly inseminating a limited region around the interface with massless particles of different size [30]. The markers are initially associated with each side of the interface, i.e. are set positive or negative depending on their location with respect to Γ. Then, they are moved by the same velocity field acting on the Level-Set function and therefore should remain on the proper side of ϕ since the interface represents a contact discontinuity. When this requirement is not fulfilled (because of numerical diffusion) the sign of the updated ϕ is wrong and must be corrected. In this situation, the radius of the escaped markers plays the role of a local Level-Set function and is used to change the value of ϕ at the nodes of the computational domain. The PLS method has been successfully tested both on 2D and 3D configurations and, at present, can be considered the most effective way to code the Level-Set technique for tracking interfaces. In a subsequent paper, it was shown that the correction provided by the particles even makes useless the adoption of high-order finite difference schemes [31].

As the PLS method is recognized as the state-of-the-art among the numerical approaches for tracking interfaces based on the Level-Set methodology, this procedure was taken as a reference-point to check the capability of the method proposed in this paper; to compute our reference numerical data, a PLS-based code has been implemented by using the same parameters suggested by the authors in [30]. Then, the vortex in a box test case has been taken into account and treated by a rather fine (100×100) Cartesian mesh, with the aforementioned high-order finite difference schemes. The starting circular interface has a radius $r = 0.15$ and is centered at (0.5,0.75) in a unit square domain, while the velocity components are given by

$$u(x,y) = -2\sin(\pi y)\cos(\pi y)\sin^2(\pi x)$$
$$v(x,y) = +2\sin(\pi x)\cos(\pi x)\sin^2(\pi y)$$

Fig. 1 shows the comparison of the interface determined at $t = 3$ by a standard LS approach (on the left) and the PLS code (on the right). It is easy to recognize the effectiveness of the PLS method in limiting the loss of mass due to the numerical diffusion and the improvements achieved in tracking the interface both at the nose and, above all, the tail of the vortex.

In spite of the good results achieved by the PLS method, some problems can be still recognized. First of all, the local character of the corrections provided by the particles can give rise to a rather irregular behaviour of the zero level representing the interface. This can be irrelevant for computer graphics purposes, but may represent an undesirable limit for hydrodynamic simulations, where an accurate estimation of the curvature is required to determine surface tension effects. Furthermore, many particles must be used to achieve an effective correction of the zero level of ϕ, most being never used in the correction process itself. Thus, the computational effort is generally relevant and can become prohibitive for three-dimensional problems.

To better understand these important issues, Fig. 2 shows the comparison between the PLS solution (top figures) and the "exact" configuration of the vortex (bottom figures) at $t = 5$, when Γ is characterized by the presence of very thin filaments. The results have been obtained on a finer (200×200) mesh and by using approximately 6×10^4 particles placed on both

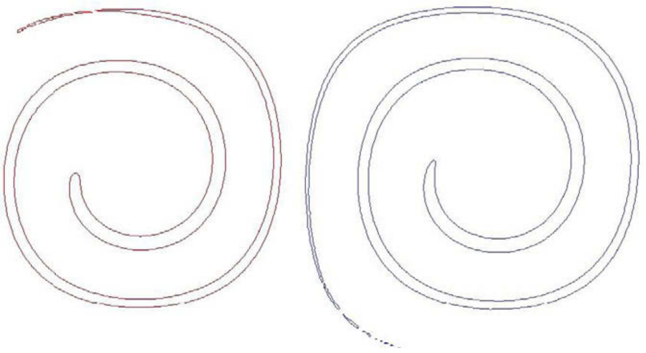


Fig. 1. Comparison of the Level-Set (left) and the particle Level-Set (right) solutions for the vortex test-case at $t = 3$.

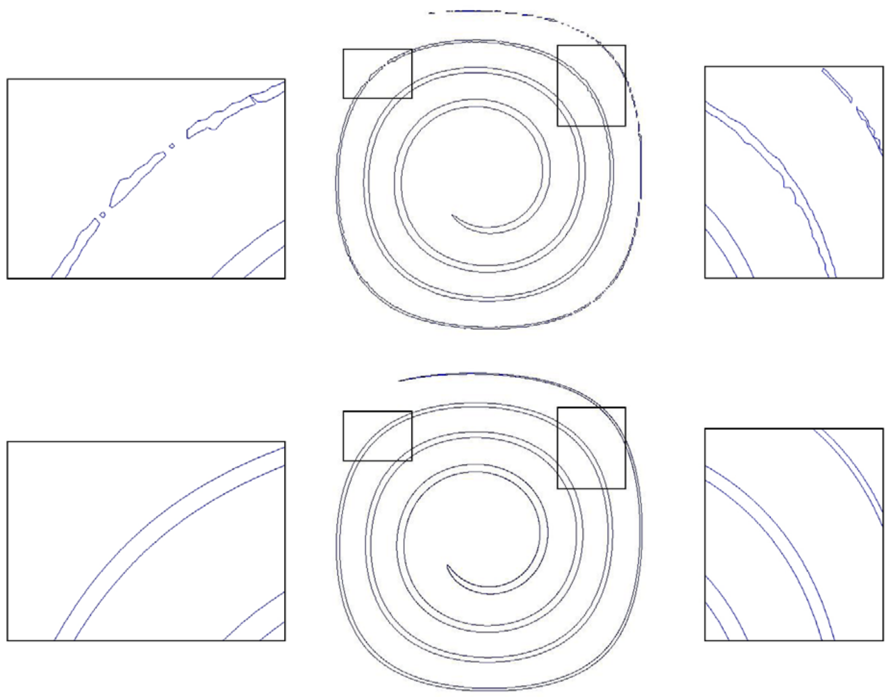


Fig. 2. The PLS solution at $t = 5$ (top figures) exhibits a rather ragged profile on the outer rings of the vortex and a not negligible fragmentation compared to the exact interface (bottom figures).

sides of the interface. The agreement is quite satisfactory and the use of the particles (combined with the higher spatial mesh resolution) provides a very limited loss of mass. Nevertheless, the outer rings of the vortex locally exhibit a notable fragmentation, especially at the vortex tail region, and a very irregular profile. This behaviour is highlighted in the comparison of the zoom views reported in the same figure. The irregularities can be probably alleviated by increasing the number of markers or using some refinement techniques. Substantially, however, they cannot be removed since they depend on how the particles are used in the correction process. The results here reported perfectly mirrors the analogous results in [30].

3. Lagrangian markers and merging fronts

Before introducing the oriented particles, it's worthwhile to briefly discuss a further 2D test-case proposed in [30] and concerning a shrinking square where each side is advected with a constant speed normal to the front itself. Such a particular configuration was used by the authors to demonstrate that a pure Lagrangian front-tracking model is not able to trace the interface correctly when merging of fronts occurs.

This problem is rather anomalous since a discontinuity occurs (at the square corners) on the normal vector to Γ and, consequently, on the corresponding component of the velocity. Nevertheless, in this paper we will show that a merging front problem can be handled successfully by Lagrangian particles provided that the particles are managed in the proper way.

Fig. 3 shows the starting configuration of both the interface and the velocity field, where the velocity direction is defined, as in [30], by the unit normal vector $\mathbf{n}$ computed through the Level-Set function and Eq. (2). In this problem, however, the cell vertices close to the corners are affected by the presence of the discontinuity and an error occurs in the evaluation of the velocity. This error is obviously transferred to the particles, whose velocity components are determined through a simple bilinear interpolation of the values at the grid vertices. The situation is clearly shown in the left Fig. 4, where the markers and the corresponding velocity vectors at the top-left corner of the square are reported at the starting time $t = t_0$ and at $t = 5\Delta t$ (where we set Δt according to the CFL condition, with $|\mathbf{u}| = 0.01$). From the figure it is clear that the particles approaching the corner do not move correctly, their motion being retarded with respect to both front sides. For this reason, a slender filament appears along the diagonal. This behaviour was clearly described in [30] and gives rise to the subsequent configurations of Γ depicted in the right Fig. 4 (at $t = 20\Delta t$ and $t = 40\Delta t$, respectively), characterized by an undesired wake shed by the square corners.

Nevertheless, this behaviour is not to be ascribed to the Lagrangian markers but to the way in which the velocity is assigned to them via the Level-Set function derivatives. As a matter of fact, in the formulation of the problem an entropy condition must be enforced to make the solution unique and remove the undesired solutions. In this case, if no condition at all is enforced within the simulation, the point close to the corner would go beyond the diagonal and would give rise to a multi-

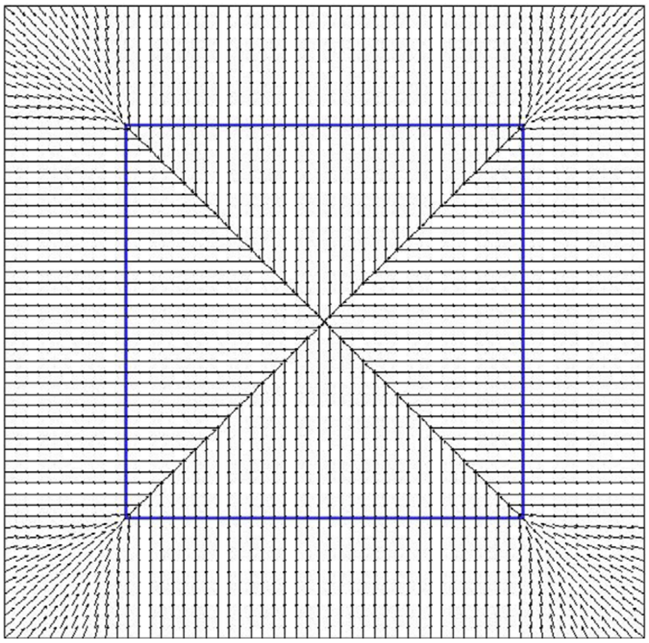


Fig. 3. Shrinking square: initial interface configuration Γ_0 and the velocity field.

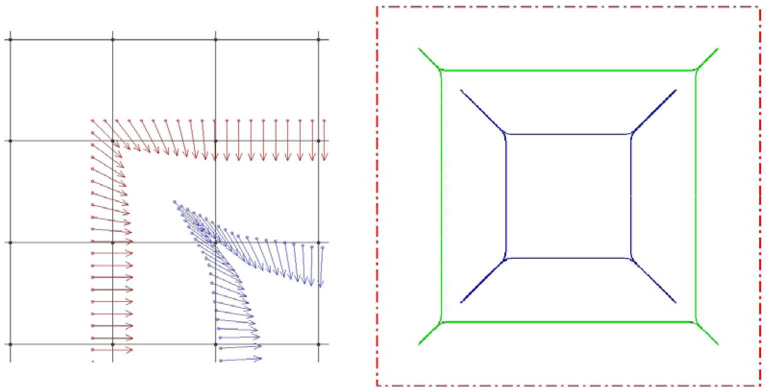


Fig. 4. Left figure depicts the particle velocities at the top-left corner of the square, as incorrectly determined through the use of the $\mathbf{n} = \nabla\phi/|\nabla\phi|$ vector. The right figure shows the initial configuration (dashed–dotted line) and the interface traced at two subsequent steps.

valued solution. Although the velocity field depicted in Fig. 3 (computed as in [30]) seems to be conceived to yield a single value solution (i.e. with the fronts merging and the locally correct direction), the magnitude of the resulting velocities is not sufficient to preserve the two fronts straight. Then, the corners are left behind with respect to the rest of the interface. We want to underline again that the problem does not concern the use of particles, but rather the way in which the velocity field is computed. We will come back on this interesting issue in Section 9.

4. The oriented particles

As already mentioned, in [31] it was shown how the use of Lagrangian markers made the use of high-order finite difference schemes to solve the transport equation for ϕ redundant. In fact, the effectiveness of the massless particles to correct locally the Level-Set function enables the use of a simple (and fast) first-order upwind scheme for computing the spatial term $\nabla\phi$. In this manner, it also becomes easier to use some adaptive mesh technique and achieve an accurate interface reconstruction in a reduced CPU time. Such a high potential of the particles has been recognized for a long time. However, our feeling is that it has not yet been fully exploited.

This last assertion can be appreciated by looking at the bottom Fig. 2, where the exact interface configuration is traced by using a huge number of particles placed on Γ and moved by the 3rd order Runge–Kutta integration. There's no doubt that the markers exactly located on the interface provide a more accurate information about the shape of the interface; nevertheless, positive and negative particles around the interface have to be used in the PLS method in order to identify the two domains. The basic idea of the oriented particles is to link this fundamental information to the particles themselves. To this aim, at the starting time t_0 we put N_p particles upon the interface and evaluate, at each particle location $\mathbf{x} = \mathbf{x}_P$, the local (outward) normal vector $\mathbf{n} = \mathbf{n}_P$ to Γ. Obviously, the time evolution of $\mathbf{x}_P$ is determined by

$$\frac{d\mathbf{x}}{dt} = \mathbf{u} \tag{5}$$

where $\mathbf{u} = \mathbf{u}_P$ is the velocity vector at $\mathbf{x}_P$. Similarly, an evolution equation of the corresponding normal (unit) vector can be easily determined by taking the gradient of the evolution equation for ϕ:

$$\frac{D\mathbf{n}}{Dt} = \frac{\partial \mathbf{n}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{n} = \frac{\partial}{\partial t}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \mathbf{u}\cdot\nabla\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{6}$$

or, in terms of vector components (by using the Einstein summation convention on repeated indices)

$$\begin{aligned}\frac{Dn_i}{Dt} &= \frac{\partial}{\partial t}\frac{\phi_{,i}}{|\nabla\phi|} + u_j\frac{\partial}{\partial x_j}\frac{\phi_{,i}}{|\nabla\phi|} = \frac{\phi_{,it}}{|\nabla\phi|} - \frac{\phi_{,i}}{|\nabla\phi|^3}\phi_{,k}\phi_{,kt} + u_j\left\{\frac{\phi_{,ij}}{|\nabla\phi|} - \frac{\phi_{,i}}{|\nabla\phi|^3}\phi_{,k}\phi_{,kj}\right\}\\ &= \frac{1}{|\nabla\phi|}\{\phi_{,it} + u_j\phi_{,ij}\} - \frac{\phi_{,i}\phi_{,k}}{|\nabla\phi|^3}\{\phi_{,kt} + u_j\phi_{,kj}\} = \frac{1}{|\nabla\phi|}\left\{\frac{\partial}{\partial x_i}(\phi_{,t} + u_j\phi_{,j}) - u_{j,i}\phi_{,j}\right\} - \frac{\phi_{,i}\phi_{,k}}{|\nabla\phi|^3}\left\{\frac{\partial}{\partial x_k}(\phi_{,t} + u_j\phi_{,j}) - u_{j,k}\phi_{,j}\right\}\\ &= -u_{j,i}n_j + u_{j,k}n_jn_kn_i\end{aligned}$$

since

$$\phi_{,t} + u_i\phi_{,i} = \frac{D\phi}{Dt} = 0$$

Thus, Eq. (6) can be rewritten as

$$\frac{D\mathbf{n}}{Dt} = -u_{j,i}n_j\mathbf{e}_i + u_{j,k}n_jn_k\mathbf{n} \tag{7}$$

$\mathbf{e}_i$ being the three base unit vectors. Note that, although we used the relation between $\mathbf{n}$ and ϕ to derive Eq. (7), the Level-Set function does not appear any longer and the time evolution of the normal vector only depends on the velocity field (through the term $\nabla\mathbf{u}$) and $\mathbf{n}$ at the current time step. In this way, starting from an initial configuration $\mathbf{n} = \mathbf{n}_0$ (at $t = t_0$) we can use the same 3rd order Runge–Kutta integration used for solving Eq. (5) to compute $\mathbf{n}$ at subsequent steps and follow the time evolution of the normal vector to Γ. The coupled solution of Eqs. (5) and (7) removes the need of accounting for positive and negative particles of the PLS method, since at any time step the $\mathbf{n}_P$ vector related to the particle $\mathbf{x}_P$ uniquely identifies the two separated domains. For this reason the markers are called *oriented particles*. In order to show the capability of Eq. (7) in tracking the time evolution on the outward normal vector to Γ, Fig. 5 shows the numerical solution for the vortex at $t = 2$. For the sake of clearness, only 200 particles have been taken into account in the left figure to show the global normal distribution to the interface, while the right figure highlights the accuracy of the numerical result around the tail region of the vortex, characterized by a cusped shape and here traced by 1000 markers. In order to prevent possible accumulation of numerical errors, at each step $\mathbf{n}$ is explicitly normalized to maintain $|\mathbf{n}| = 1$. It is important to note that even though the particles provide a discrete representation of the interface, they cannot be used to trace it in a direct way (by any data-fitting procedure), since no connection law exists between two subsequent markers. Thus, we still need the Level-set function ϕ to trace Γ and for

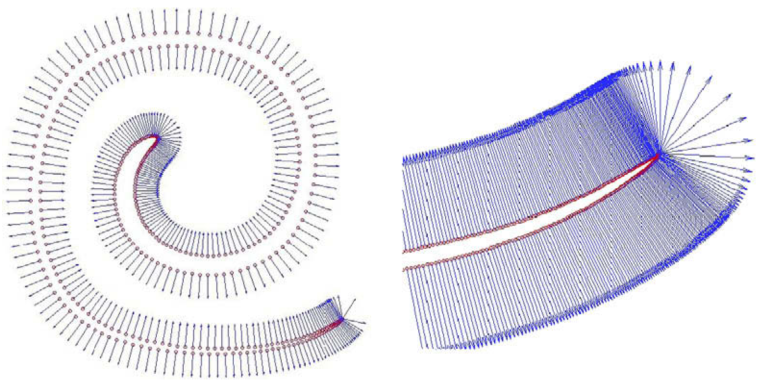


Fig. 5. The left figure shows the distribution of $\mathbf{n}$ related to 200 oriented particles, as determined for the vortex test-case $t = 2$. On the right, the zoom view highlights the accuracy of the normal vector distribution computed on the cusped tail region of the interface.

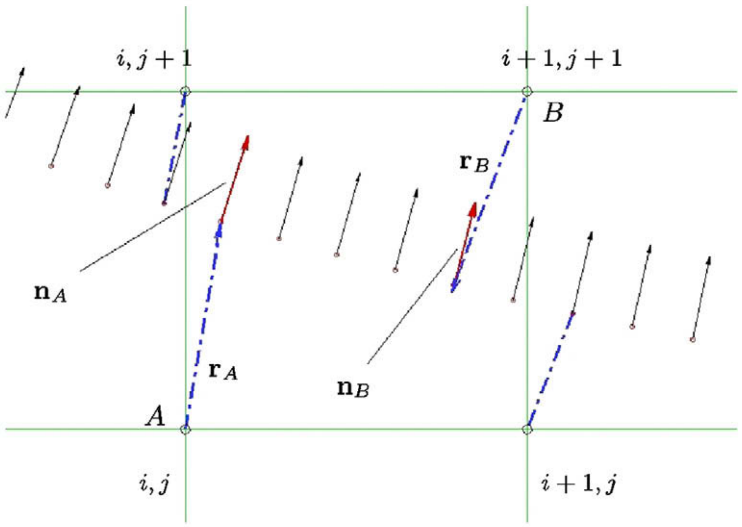


Fig. 6. The Level-Set function at the vertices of an interface cell is carried out by the scalar product of the minimum distance vectors (dashed–dotted lines) and the outward normal vectors associated to the corresponding closest particles.

this reason the procedure has been named the oriented particle Level-Set method. However, the ϕ zero level can be determined through the oriented particles in a very simple and fast way.

In fact, by a single loop on the particles we can identify the interface cells (i.e. the cells containing at least one particle) and compute the distance between their vertices and the markers. This distance is updated within the loop in order to identify the minimum value and, consequently, the closest marker with respect to each vertex of the mesh. Then, the Level-Set function at the vertices of any interface cell is provided by the scalar product between the minimum distance vector and the corresponding particle (unit) normal vector.

The procedure is outlined in Fig. 6, where the minimum distance vectors at the cell vertices A and B are depicted by a dashed–dotted line and the corresponding function ϕ has a sign that depends on the sign of the scalar products $\mathbf{r}_A \cdot \mathbf{n}_A$ and $\mathbf{r}_B \cdot \mathbf{n}_B$, respectively. The knowledge of the signed distance function at all cells vertices straddling the interface enables the initialization of a solver devoted to the evaluation of the function ϕ in the whole numerical domain through the Eikonal equation

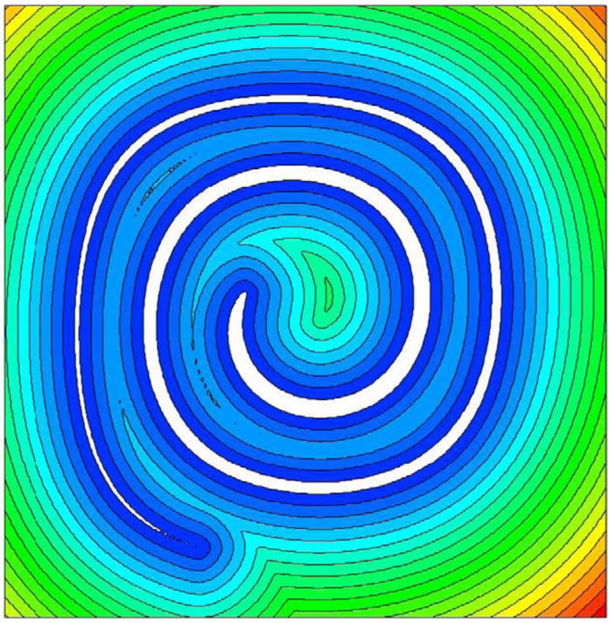


Fig. 7. Contour plots of the level-set function field ($\phi > 0$) determined for the vortex at $t = 3$ through the oriented particles and the fast sweeping method.

$$|\nabla\phi| = 1 \tag{8}$$

Fig. 7 shows the contour plots of the function ϕ field; the computation was carried out by an Eikonal solver based on the fast sweeping method [36]. Here, we have used the fine (200×200) mesh with 2000 oriented particles and the negative levels have been removed in order to make the picture clearer. It's worth noting that the whole simulation, including a step-by-step solution of Eq. (8) (2400 steps with Δt set to 1.25E−3) has required less than 17 s of CPU time on a PC equipped with a (single) Xeon processor at 2.5 Ghz; this value cannot be assumed as an absolute measure of the CPU requirements, but it has to be intended as a qualitative measure of the order of magnitude for CPU time burden.

At this stage, it is useful to point out the following issues:

1. The oriented particles are totally unrelated, since the time evolution of each particle and the corresponding normal vector does not depend on the others. Such a feature is very important, since it allows to handle data in an unstructured fashion. This is extremely desirable for both parallel coding and the seeding procedure introduced in the following.
2. The time evolution of the interface is determined without any transport equation for ϕ. Thus, no diffusion problem affects the numerical solution, whose accuracy depends only on the time integration. Of course, this is true for the present example, where the analytical values of the velocity field and its derivatives are available. In a hydrodynamic simulation, where the velocity components are determined at the mesh nodes, spatial discretization errors occur because the velocity field must be transferred from the nodes to the particles by means of some data-fitting procedure. In any case, the inaccuracies related to the numerical discretization of the spatial term $\nabla\phi$ are completely removed. It's worth noting that although the function ϕ simply plays the role of a post-processing tool for tracking Γ, its use is useful to maintain unaltered the structure of all the existing solvers based on the Level-Set methodology.
3. The use of an Eikonal solver and the step-by-step numerical solution of Eq. (8) replaces any separate reinitialization process on ϕ: at each time step the computed Level-Set function is exactly the signed distance function we need at the nodes of the Cartesian mesh. In three-dimensional simulations, where the computing effort rapidly increases, the evaluation of ϕ can be suitably limited to a prescribed distance from the particles.

In order to appreciate the performances of the new method, Fig. 8 shows the vortex interface at $t = 5$, corresponding to Fig. 2 and obtained through the use of the oriented particles.

The zoomed views highlight the accuracy of the numerical result with respect to the PLS approach and the notable similarity with the exact solution. In practice, the only appreciable discrepancies are due to the spatial resolution of the mesh and occur when Γ is so thin as to be included in a single cell; this situation gives rise to the fragmentation of the vortex tail appearing in the same figure. In this case the sign of the Level-Set function is the same at the vertices of the interface cell and the Eikonal solver fails to determine the ϕ function field locally. It's interesting to note that such singular cells (containing some markers but exhibiting the same sign of the function ϕ at its vertices) are easily identified and could be somehow treated by some adaptive mesh technique. The fundamental difference, however, between the use of the (oriented) particles and the solution of Eq. (1) for tracking interface stands in the persistence of the particles: the information is never lost and cannot be numerically diffused. Even when Γ stretches into very thin filaments and the Eikonal solver fails to recognize the interface cells, the particles are still there and go on with the evolution. Moreover, in the present model the markers exactly represent all points of the interface and their location dictate, at each step, the values of the function ϕ at the mesh nodes. Therefore, the main difference with the PLS method is that the particles are not used to correct locally the zero level of ϕ determined by the transport equation, but rather to evaluate it explicitly. In this way, the aforementioned high potentiality of the particles is fully exploited.

Fig. 9 shows eight consecutive contour plots of ϕ for the vortex in a box test-case, where the simulation (performed by 5×10^3 markers in the fine 200×200 mesh) lasts 20 and is characterized by an inversion of the velocity field at $t = 10$. As expected, the interface suffers a strong thinning and appears as a fragmented filament. Nevertheless, the field computed by

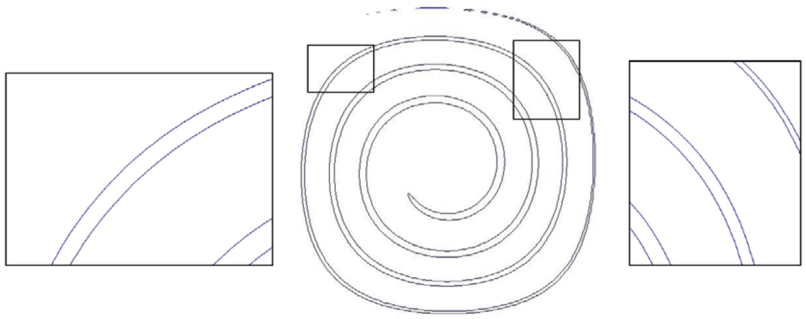


Fig. 8. The numerical solution for the vortex at $t = 5$ exhibits a regular profile even on the outer rings of the vortex, very similar to the exact interface configuration reported in the bottom Fig. 2.

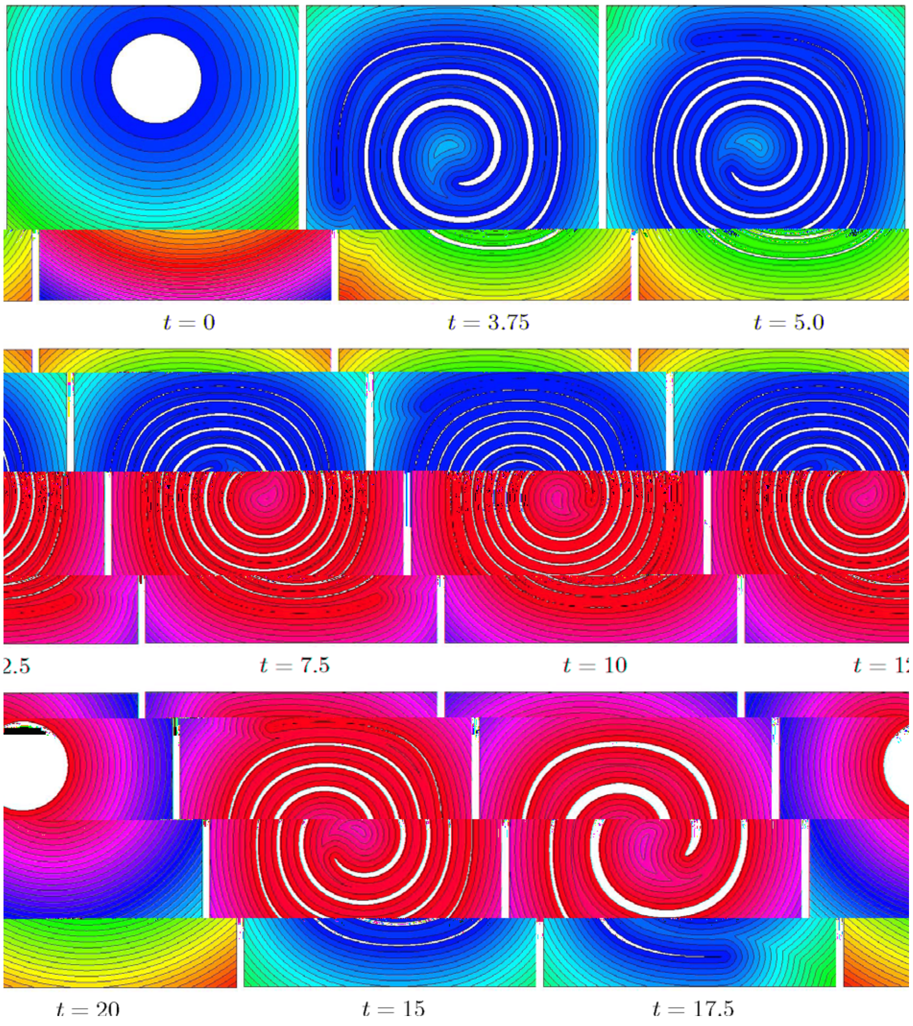


Fig. 9. Contour plots of the level-set function field for the vortex in a box test-case, with a motion inversion at $t = 10$. The notable fragmentation of zero level of ϕ, due to the occurrence of very thin filaments, does not jeopardize the perfect recomposition of the starting circular Γ at $t = 20$.

the Eikonal solver through the use of the oriented particles always represents the best achievable result according to the spatial resolution of the mesh. Moreover, the starting circular interface is perfectly rebuilt at the end of the simulation.

Fig. 10 shows the contour plots of the function ϕ carried out by 2×10^3 oriented particles for the Zalesak's disk test-case. As before, the starting circular interface ($r = 0.15$) is centered at (0.5,0.75), while the velocity field is characterized by the following components

$$u(x,y) = +\pi(y - 0.5)$$
$$v(x,y) = -\pi(x - 0.5)$$

The simulation covers five complete disk revolutions (each being completed in a time interval equal to 2) and the figure shows ϕ as determined by the Eikonal solver at eight different time steps. As expected, the zero level of ϕ is perfectly maintained during the rotation and no distortion affects the aforementioned critical region of the corner points, up to the end of the simulation.

5. Evolution equations for the second fundamental tensor and the tangent vectors

In Section 4 we derived the evolution equation of the (outward) normal vector to the interface and showed how to determine the Level-Set function field from the oriented particles. Numerically, we could evaluate the curvature κ at the mesh nodes from Eq. (3) and, then, on the particles through an interpolation procedures. A further evolution equation, however, can be derived for $\nabla\mathbf{n}$, the second fundamental tensor of the surface Γ (whose trace exactly corresponds to κ), which is then

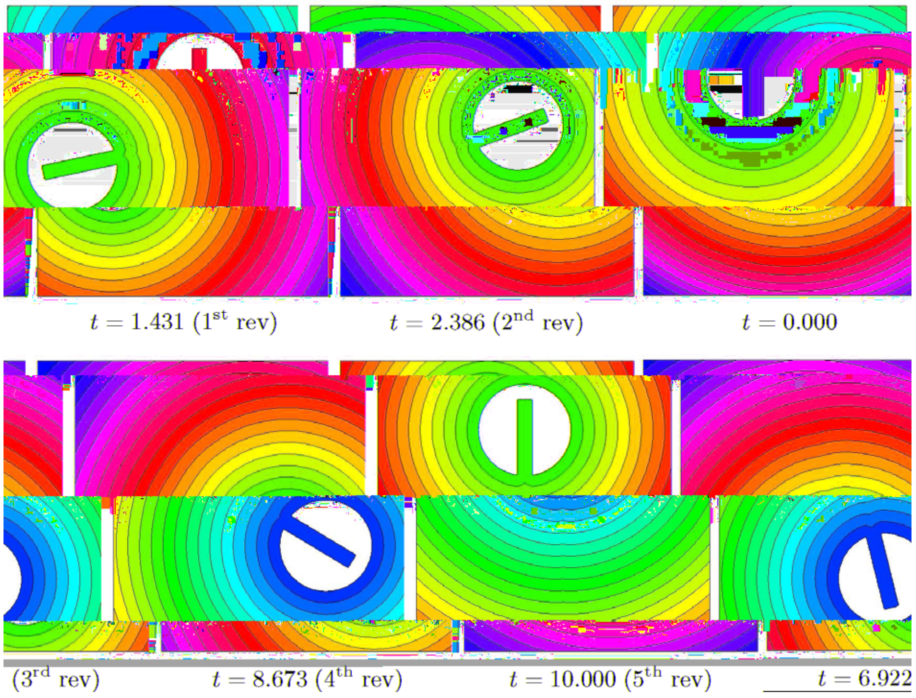


Fig. 10. Contour plots of the level-set function field for the Zalesak's disk test-case. The last frame corresponds to $t = 10$, at the end of the fifth disk revolution.

considered as an additional quantity linked to the markers. In this way, the set of data passively advected by the oriented particles completely define the local interface geometry and still remain independent of the evaluation of the function ϕ on the Eulerian mesh. For clarity, we recall Eq. (7), written in terms of components

$$\frac{Dn_i}{Dt} = u_{j,k}n_jn_kn_i - u_{j,i}n_j$$

and determine the spatial derivative of this equation. The left-hand side reads

$$\left[\frac{Dn_i}{Dt}\right]_{,s} = \left[\frac{\partial n_i}{\partial t} + u_j\frac{\partial n_i}{\partial x_j}\right]_{,s} = \frac{\partial n_{i,s}}{\partial t} + u_j\frac{\partial n_{i,s}}{\partial x_j} + u_{j,s}n_{i,j} = \frac{Dn_{i,s}}{Dt} + u_{j,s}n_{i,j}$$

whereas the right-hand side becomes

$$[u_{j,k}n_jn_kn_i - u_{j,i}n_j]_{,s} = u_{j,ks}n_jn_kn_i + u_{j,k}n_{j,s}n_kn_i + u_{j,k}n_jn_{k,s}n_i + u_{j,k}n_jn_kn_{i,s} - u_{j,is}n_j - u_{j,i}n_{j,s}$$

Then, the evolution equation for $\nabla\mathbf{n}$ can be written in the following form

$$\frac{Dn_{i,s}}{Dt} = u_{j,ks}n_jn_kn_i + u_{j,k}n_{j,s}n_kn_i + u_{j,k}n_jn_{k,s}n_i + u_{j,k}n_jn_kn_{i,s} - u_{j,is}n_j - u_{j,i}n_{j,s} - u_{j,s}n_{i,j} \tag{9}$$

The right-hand side of this equation exhibits, for each particle, the first and second spatial derivatives of the velocity (which can be determined through the aforementioned data-fitting procedures when only a discrete **u** field is available), the normal vector components (known from Eq. (7)) and the tensor components $n_{i,j}$ whose time evolution we are looking for. Then, starting from an initial value of such a second-order tensor $\nabla\mathbf{n}|_{t=t_0}$, Eq. (9) is integrated forward in time through the usual Runge–Kutta solver and provides, at each step, the interface curvature corresponding to the particles by the relation

$$\kappa = \mathrm{tr}(\nabla\mathbf{n}) = n_{i,i} \tag{10}$$

In order to test the capability of Eq. (9) in determining the time history of the interface curvature, we still refer to the vortex in a box test case and run the simulation up to $t = 1.5$ when the velocity field is inverted. Top Fig. 11 shows the three configurations of Γ at $t = 0$, 1.5 and 3 and the instantaneous position of four different particles whose time evolution has been analysed.

As expected, the last interface configuration perfectly matches the starting one. Nevertheless, the most interesting result is reported in the bottom figures showing the time histories of the curvatures κ corresponding to the selected points and

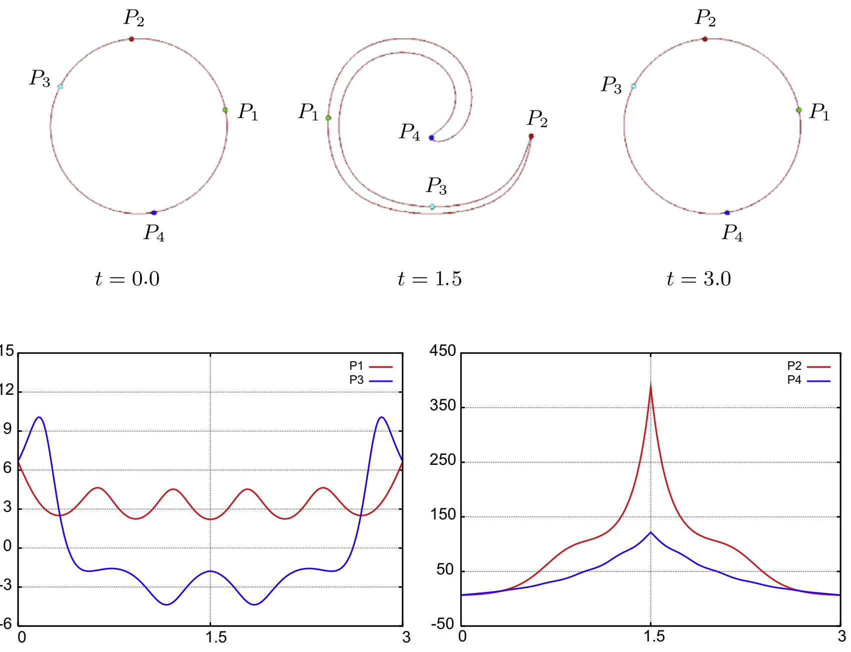


Fig. 11. Top figures show the interface configurations of the vortex at three subsequent time steps of an inverted motion ($t_{inv} = 1.5$ and four oriented particles used for computations. At the bottom, the corresponding four time histories of the curvature κ are reported.

determined by Eq. (9). At $t = 0, \kappa = 1/r$ (being $r = 0.15$) at all the oriented particles of the circular interface. During the simulation the curvature at points P_1 and P_3 fluctuates rather close to the starting value, while at P_2 and P_4 a large gradient occurs, since the corresponding particles move along the tail and the nose of the vortex where κ attains the highest values. Note the negative values of κ at P_3, a marker associated to an outward normal vector pointing towards the local center of curvature (as in the configuration at $t = 1.5$ reported in the top-center figure). Furthermore, as expected, all the time histories for κ show a perfect symmetry with respect to $t = 1.5$, the instant when the motion is inverted. For completeness, Table 1 provides the numerical values of the curvature at the aforementioned points, where it can be seen that the discrepancies between the starting and final configurations (i.e. the numerical errors due to the integration procedure) are very limited.

There is another important quantity related to the particles that is convenient to compute during the time evolution of Γ. Corresponding to each particle P, any tangent vector $\mathbf{t}$ to the interface is a function of both the space variables $\mathbf{x}$ and time t and is defined as

$$\mathbf{t} = \mathbf{t}(\mathbf{x}, t) = \frac{\partial \mathbf{x}}{\partial \xi} \tag{11}$$

where ξ represents a curvilinear coordinate on Γ. By fixing the particle P, the material time derivative of this quantity can be written

$$\left.\frac{d\mathbf{t}}{dt}\right|_{\mathbf{x}=\mathbf{x}_P} = \frac{\partial \mathbf{t}}{\partial t} = \frac{d}{dt}\left(\frac{\partial \mathbf{x}}{\partial \xi}\right) = \frac{\partial}{\partial \xi}\left(\frac{d\mathbf{x}}{dt}\right) = \frac{\partial \mathbf{u}}{\partial \xi} \tag{12}$$

The gradient $\nabla_\xi \mathbf{u}$ appearing on the right-hand side can be expressed in terms of Cartesian components in the following form

$$\frac{\partial u_i}{\partial \xi} = \frac{\partial u_i}{\partial x_j}\frac{\partial x_j}{\partial \xi} = \frac{\partial u_i}{\partial x_j} t_j$$

so that Eq. (12) (in terms of components) reads

Table 1
Curvature values at the four P_i oriented particles highlighted in Fig. 11 for the inverted motion of the vortex in a box test-case.

	P_1	P_2	P_3	P_4
$t = 0.0$	0.666e+01	0.666e+01	0.666e+01	0.666e+01
$t = 1.5$	0.221e+01	0.388e+03	−0.176e+01	0.121e+03
$t = 3.0$	0.660e+01	0.667e+01	0.671e+01	0.671e+01

$$\frac{\partial t_i}{\partial t} = \frac{\partial u_i}{\partial x_j}\, t_j \tag{13}$$

It's essential to note the link between the time derivative of the tangent vector **t** and the spatial term $\nabla \mathbf{u}$, a quantity directly related to the rate-of-strain tensor. In other words, the evolution of **t** is related to the deformation of the velocity field and, consequently, to the deformation of Γ during the time evolution. In order to demonstrate this last assertion, Eq. (13) has been solved for the vortex in a box test case, by computing the vectors **t** on the starting circular interface and assigning to their magnitude the unit value. For clarity, only 100 oriented particles have been used. The resulting configurations at $t = 0$, 1.5 and 3 are shown in Fig. 12, where the tangent vectors are reported. It is evident that $|\mathbf{t}|$ increases at points where the curvature of Γ decreases, according to the tendency of the particles to spread out. It will be shown, in the following sections, how the behaviour of these vectors can be exploited to check the possible clustering or depletion of markers on the interface and to change their spatial distribution during the numerical simulation.

6. Failure of the algorithm

At this stage the oriented particles represent a comprehensive set of data which enables to trace Γ in a very fast and accurate way. Through the evolution equations derived in the previous sections they provide, at each time step, the mutual location of the subdomains, the local outward normal vector and curvature on Γ and even the possible stretching suffered by the interface during its motion. Unfortunately, there is an implicit weak-point in the sole use of Lagrangian markers: the spatial distribution of the particles. If the markers must govern the interface tracking process, two obvious questions arise immediately: how many particles do we have to use and where do we have to place them?

As a matter of fact, no condition assures that at each step the number and the position of markers adopted for the simulation are sufficient to provide a correct reconstruction of Γ and, in our case, the right estimation of the function $\mathcal{S}(\phi)$. Fig. 13 shows two typical situations where the oriented particles can fail in tracking Γ (here depicted by a solid, red line). In the left figure, the interface is shown at two subsequent time steps (t and $t + \Delta t$) and at three contiguous cells of the computational domain. At time t the outlined procedure for the assignment of the $\mathcal{S}(\phi)$ function at the cells vertices works correctly; on the contrary, at $t + \Delta t$ the center cell C is no longer recognized as an interface cell, since the velocity field causes all the particles to go out from the cell itself. Although at this step the identification of $\mathcal{S}(\phi)$ at the vertices of C vertices would be still realizable through the particles in the neighbouring L and R cells, at subsequent steps (when none of these cells will contain any marker) the sign of the Level-Set function will be not defined in the correct way. In fact, not all mesh nodes actually straddling the interface are recognized in the initialization of the Eikonal solver, and a local error will affect the evaluation of the zero level of ϕ. Note that the finer is the grid with respect to the marker distribution, the easier is the occurrence of such void cells during the simulation. An even worse situation is depicted in the right Fig. 13 where again the very coarse distribution of oriented particles causes an error in the estimation of $\mathcal{S}(\phi)$: in fact, particle 2 would be identified as the closest marker to the vertex $i + 1, j + 1$ and a wrong sign for the Level-Set function at that node would follow, due to the normal vector orientation at particle 2. In both cases, the errors arise from an unsuitable distribution of the markers, not sufficient to represent the interface correctly.

It's interesting to note that these errors initially give rise to the occurrence of some bubbles within the interface, because the wrong identification of $\mathcal{S}(\phi)$ at some computational nodes corresponds to an inversion of the two domains. These errors could be identified and removed with some numerical device capable of analysing the surrounding distribution of the function, but it would soon become ineffective to manage the general case in three dimensions. In practice, a progressive scattering of the markers can cause the number of bubbles to grow very rapidly and the interface reconstruction process can easily become unreliable.

To mend this problem, we could simply increase the starting number of particles in order to assure, in all cells and at each time step, a sufficient concentration of markers. Obviously, such a solution can provide an impressive increase of the com-

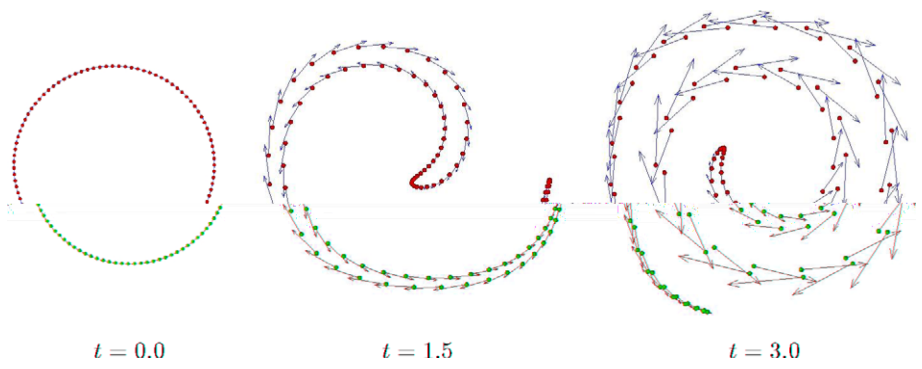


Fig. 12. The spatial distribution of the tangent vector to Γ, determined at three time steps on the vortex for 100 oriented particles. An increasing value for $|\mathbf{t}|$ corresponds to a larger stretching of the interface.

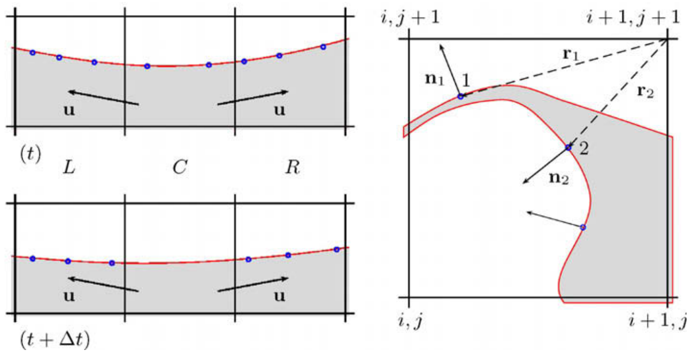


Fig. 13. Critical configurations where a coarse distribution of the oriented particles make them not able to identify the sign of the Level-Set function at the vertices of a cell (left figure) or even provide a wrong evaluation of it (right figure).

puting cost and, while practicable in 2D simulations, it could be unbearable in 3D. A more reasonable approach could be the coupling of the oriented particles to the usual transport equation for ϕ. In this context, the Level-Set function should be updated at each time step t according to the field ϕ determined by the Eikonal solver and used, at $t+\Delta t$, to assign the $\mathcal{S}(\phi)$ value at all cells not containing the particles. In this way, the solution of Eq. (1) helps the markers in the estimation of $\mathcal{S}(\phi)$ and makes the algorithm more robust. This approach can successfully deal with a pronounced rarefaction of the markers and possible interface configurations similar to that shown in the left Fig. 13, so that its use is highly recommended. Nevertheless critical situations such as the one described in the right Fig. 13 still appear unsolved, since the oriented particles dictate the (wrong) values at the vertices of the interface cells. In these cases, a suitable redistribution of the oriented particles seems to be the only way to deal with the problem.

In order to show a practical occurrence of the aforementioned numerical difficulties and to analyse their own effects on the interface tracking process, let's move our attention on a well-known 3D test case. Originally introduced in [37] and presented again in [30], the problem has been named the Enright test in [32] and consists of a sphere immersed in a velocity field given by

$$\begin{aligned} u(x,y,z) &= +2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\\ v(x,y,z) &= -\sin^2(2\pi x)\sin^2(\pi y)\sin(2\pi z)\\ w(x,y,z) &= -\sin^2(2\pi x)\sin(2\pi y)\sin^2(\pi z)\end{aligned} \tag{14}$$

the time modulation being $t=2$. As in [30], we use a sphere with $r=0.15$, centered at (0.35, 0.35, 0.35) in a cubic computational domain discretized by a 100^3 mesh. At starting time t_0 the components of the outward normal vector can be easily determined at each point $\mathbf{x}=\{x_i\}$ on the sphere through the relation

$$n_i = \frac{x_i - x_i^C}{r} \tag{15}$$

where $\mathbf{x}^C=\{x_i^C\}$ and $r=\sqrt{(x-x^C)^2+(y-y^C)^2+(z-z^C)^2}$ represent the sphere center and radius, respectively. In the exact solution the sphere, given the flow reversal, recovers its initial shape. Apart for this, this is a severe test for any 3D interface tracking algorithm, because Γ undergoes a notable deformation and stretching, to such an extend as to become locally a thin sheet with a thickness comparable with the grid spatial resolution. The knowledge of $\mathbf{n}$ at the oriented particles on the sphere enables the evaluation of two tangent vectors through the procedure suggested in [38]. Thus, $\mathbf{t}_1$ is set to the vector product between $\mathbf{n}$ and the vector with all zero components but a unit component equal to $\min(|n_i|)$. For example, if $n_x=\min\{|n_x|,|n_y|,|n_z|\}$ we have

$$\mathbf{t}_1 = \frac{\mathbf{n}\times[1,0,0]}{|\mathbf{n}\times[1,0,0]|} = \left\{0, \frac{-n_z}{\sqrt{n_y^2+n_z^2}}, \frac{+n_y}{\sqrt{n_y^2+n_z^2}}\right\} \tag{16}$$

The additional tangent vector $\mathbf{t}_2$ is directly provided by

$$\mathbf{t}_2 = \mathbf{n}\times\mathbf{t}_1 \tag{17}$$

From Eq. (15), the curvature tensor $\nabla\mathbf{n}$ at t_0 in the Cartesian reference reads

$$\left.\frac{\partial n_i}{\partial x_j}\right|_{t_0} = \frac{x_{i,j}}{r} - \frac{x_i - x_i^C}{r^2}\frac{\partial r}{\partial x_j} = \frac{\delta_{ij}}{r} - \frac{(x_i-x_i^C)(x_j-x_j^C)}{r^3} = \frac{1}{r}(\delta_{ij}-n_in_j) \tag{18}$$

δ_{ij} being the Kronecker delta. The aforementioned quantities represent all the data requested to start the procedure and the numerical results regarding the interface time evolution are summarized in the subsequent frames reported in Fig. 14.

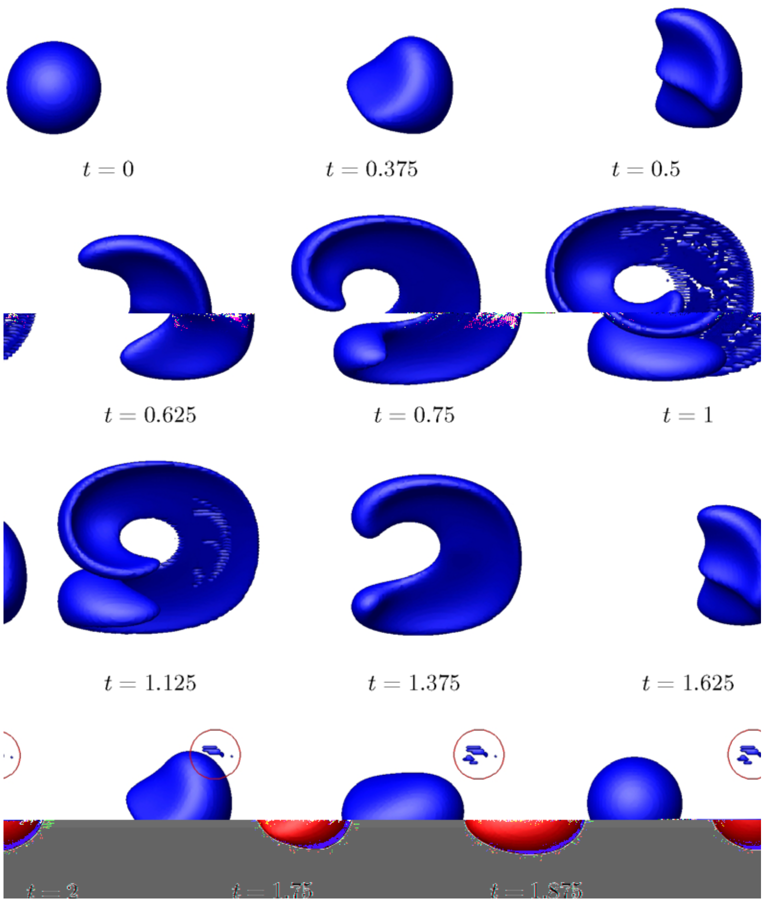


Fig. 14. The interface time evolution reconstructed by 125×10^3 oriented particles for the Enright 3D test on a Cartesian mesh 100^3. Note in the last frames the presence of the (red-circled) fictitious mass, which arises from a local wrong estimation of the $\mathcal{S}(\phi)$ function. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The pictures show an *XZ*-view of Γ in order to enable a direct comparison with the analogous frames published in [30,32]; 125×10^3 oriented particles were used for the simulation up to $t = 2$, with the motion inversion occurring at $t = 1$. It can be seen that the sphere is perfectly recomposed at the end of the simulation. Nevertheless, the last frames reveal the presence of an additional, fictitious mass. This numerical error is due to a too coarse distribution of the oriented particles corresponding to the thinning of the interface and a wrong estimation of $\mathcal{S}(\phi)$ at some node in the Eulerian grid. Some more details of the reconstructed surface Γ can be appreciated in Fig. 15, where together with the usual *XZ* and *XY* views two additional view points (Σ and Λ) have been reported. In particular, since the velocity field (14) rotates and stretches the sphere so as to align it approximately to a diagonal plane with respect to the three-dimensional cubic mesh, some relevant details of the surface are better appreciated (at $t = 1$) from these mutually perpendicular views, that together with the *XY*-view (center Fig. 15) show the upper side of Γ more clearly and reveal a remarkably rough central region. As already said, this roughness is due to the inaccurate estimation of the Level-Set function at the nodes of cells containing the (locally very thin) interface.

In order to better understand the origin of the inaccuracies affecting the computation of the interface, top Fig. 16 shows the Σ-view of the zero level of ϕ (on the left) and the corresponding distribution of the oriented particles (on the right).

In spite of the very fine particle distribution adopted (125×10^3), the markers appear highly concentrated upon the two curved, lower lobes of the interface and, at the same time, notably sparse upon the upper surface. Here, corresponding to the particles, the zero level of ϕ exhibits a (correct) void representation and tends to disappear, since the thickness of the interface is locally smaller than the mesh spatial resolution and the Eikonal solver is not able to identify any sign inversion of the Level-Set function. On the contrary, a local lack of markers on the upper surface gives rise to the jagged outline depicted in the right Fig. 16, since the procedure fails in recognizing the actual interface cells. In other words, the very irregular shape occurring on the upper region of the zero level of ϕ is of pure numerical nature. The Λ-view reported in the bottom Fig. 16 highlights the reduced thickness of Γ (on the left) and the rather unnatural protuberances occurring on the upper front (right figure) caused by the incorrect estimation of the $\mathcal{S}(\phi)$ function. Thus, contrary to the usual numerical diffusion caused by finite difference approximation in standard Level-Set approaches, our Lagrangian-based reconstruction of the interface gives

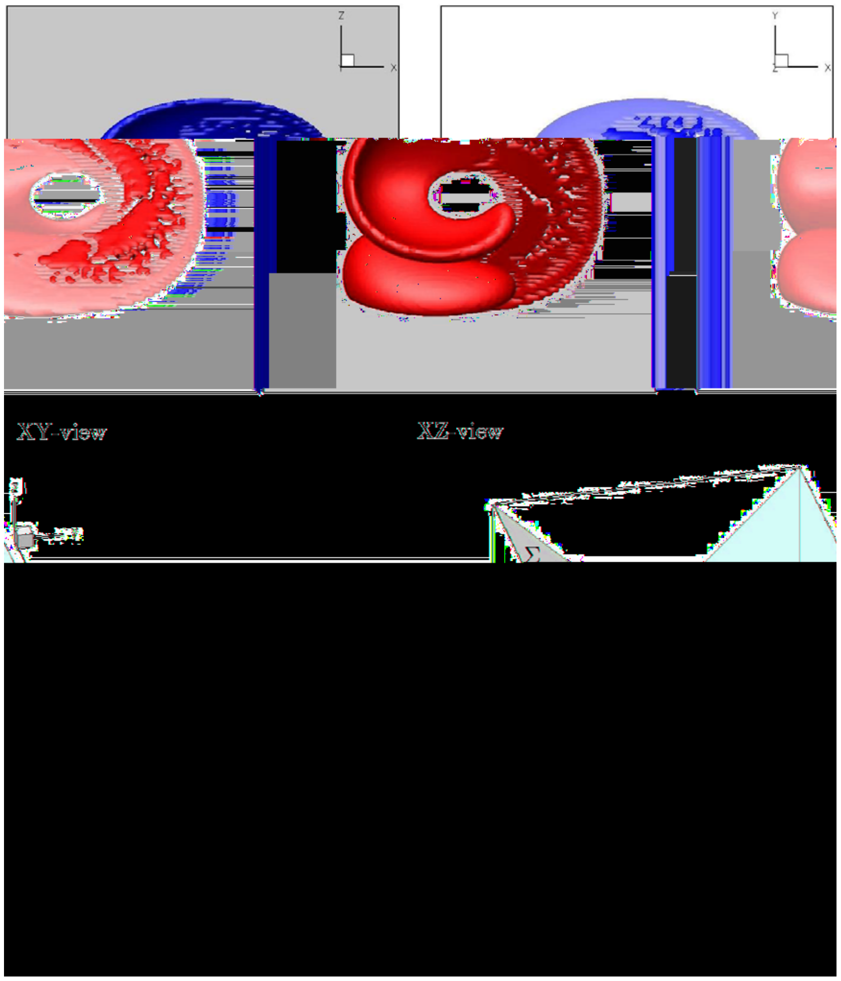


Fig. 15. The velocity field corresponding to the Enright test deforms the sphere into a stretched manifold which approximately alignes along a diagonal plane Σ within the cubic 3D mesh.

rise to a sort of additional mass, depending on the local particles distribution and motion and the mesh spatial resolution. This behaviour is clearly shown in Fig. 17, where three subsequent steps of the Enright test during the return run are depicted. The errors due to the wrong estimation of the function $\mathcal{S}(\phi)$ persist up to the end of the simulation, since the particles do not have any mechanism to remove these numerical inaccuracies on their own.

According to the above discussion concerning the possible failure of the procedure, part of the aforementioned errors can be removed in a rather simple (and convenient) way by coupling the use of the markers to the solution of the transport equation for ϕ.

Fig. 18 shows the particles and the corresponding ϕ zero level at the same time steps of Fig. 17, as computed by the aid of Eq. (1). At $t = 1.5$ the wrong values of $\mathcal{S}(\phi)$ occurring on the upper surface of Γ give rise to a rather extended fictitious mass; the numerical errors, however, are locally repaired by the estimation of $\mathcal{S}(\phi)$ through the transport equation at cells not containing the markers and the fictitious mass somehow disperses in the field. At the final step the error is reduced to some scattered micro-bubbles. Before concluding the section and suggesting an algorithm to deal with the aforementioned problems, we note that the numerical solution of the evolution equations for $\mathbf{n}$, $\mathbf{t}$ and $\nabla\mathbf{n}$ do not exhibit any particular problem for a three-dimensional configuration. For example, Fig. 19 shows the markers distributions for the Enright test at nine consecutive time steps, where each particle color represents the corresponding mean curvature determined through Eq. (9). As expected, the higher values of κ always correspond to the curved boundaries of the manifold and the starting uniform distribution is perfectly recomputed at the end of the simulation.

7. The self-adaptive oriented particle Level-Set method

The numerical results reported in the previous section prove that the use of the oriented particles for tracking an evolving interface requires some expedient to check the particles distribution on Γ and guarantee, at each time step, the correct

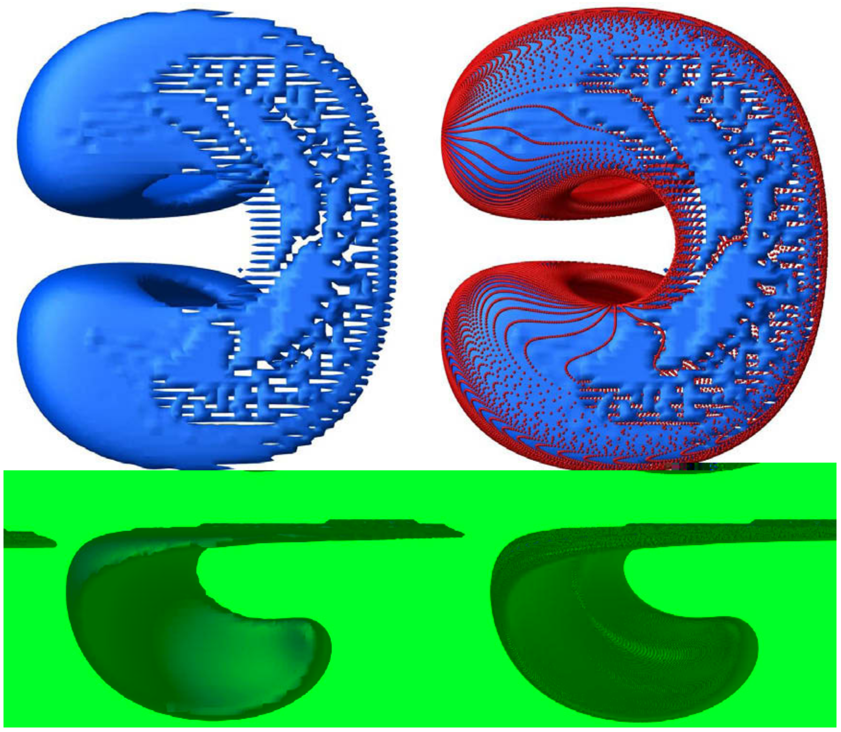


Fig. 16. The rough zero level of ϕ determined at $t = 1$ is due to the coarse distribution of markers occurring on the upper surface of Γ (top figures). The Λ-views reported in the bottom figures show the reduced interface thickness and the presence of some protuberances corresponding to the thinner front.

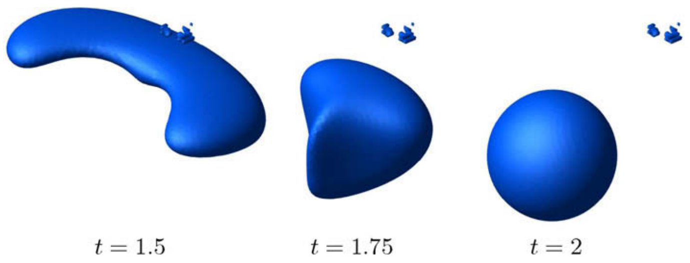


Fig. 17. The interface of the Enright test during the return run, determined through the only use of oriented particles, is affected by the persistent presence of a fictitious mass.

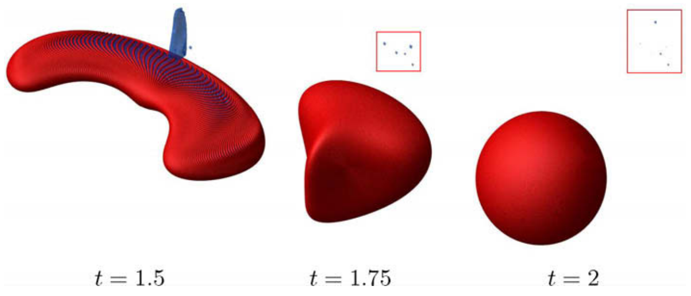


Fig. 18. The same interface configurations of Fig. 17 as determined through the aid of the transport equation for ϕ. Here both the zero level of ϕ and the oriented particles are depicted.

estimation of the function $\mathcal{S}(\phi)$ at the nodes of the Eulerian mesh. A more sound way to deal with this problem is to make the oriented particles the *seeders* of new markers and provide the algorithm with a self-adaptive mechanism, able to recognize where the particles distribution has to be refined. This fundamental task should be achieved by maintaining the particles

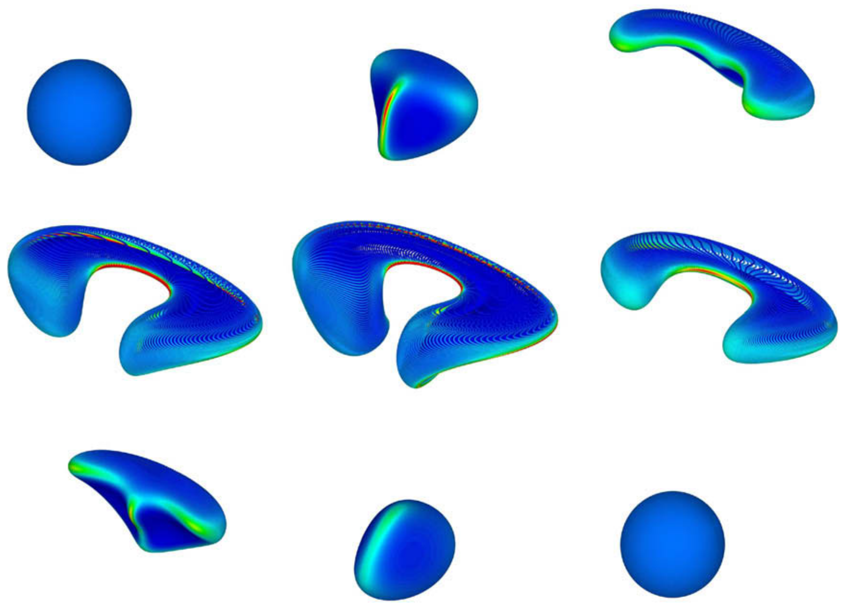


Fig. 19. Time evolution of the oriented particles used for the Enright test: the color at each marker is assigned according to the mean curvature determined by the evolution equation for $\nabla\mathbf{n}$.

unrelated and exploiting the local information provided by the unit (outward) normal vector, the tangent vectors and the shape operator. In addition, the procedure should also be able to suitably de-seed the interface in the regions where a useless gathering of markers occurs. In this way, at each time step the updated particles distribution can provide the most accurate interface reconstruction through the least computational effort.

7.1. The seeding algorithm

For the time being, leave apart where to seed the interface and let us focus our attention on how to do it. From the knowledge of the aforementioned quantities at point $\mathbf{x}_P$, we need to locate a new oriented particle on Γ, by equipping it with the appropriate values for $\mathbf{n}$, $\mathbf{t}$ and $\nabla\mathbf{n}$. This process (seeding) can be realized by expanding in Taylor series the function ϕ around $\mathbf{x}_P$ up to the second-order terms

$$\phi = \phi_P + \nabla\phi|_{\mathbf{x}_P} \cdot (\mathbf{x} - \mathbf{x}_P) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_P)^T \nabla(\nabla\phi)|_{\mathbf{x}_P}(\mathbf{x} - \mathbf{x}_P) \tag{19}$$

where $(\mathbf{x} - \mathbf{x}_P)$ represents the vector position of $\mathbf{x}$ with respect to $\mathbf{x}_P$ and the spatial operator (at $\mathbf{x}_P$) $\nabla(\nabla\phi)$ in terms of components reads

$$\phi_{,ij} = \frac{\partial\phi_{,i}}{\partial x_j} = \frac{\partial}{\partial x_j}(n_i|\nabla\phi|) = n_{i,j}|\nabla\phi| + n_i\frac{\phi_{,k}}{|\nabla\phi|}\phi_{,kj} = n_{i,j}|\nabla\phi| + n_i n_k \phi_{,kj} \tag{20}$$

At this stage, we fix a (unit) vector $\mathbf{t}_P$ at $\mathbf{x}_P$ tangent to Γ and express $\mathbf{x} - \mathbf{x}_P$ in the following form

$$\mathbf{x} - \mathbf{x}_P = \beta\,\mathbf{t}_P + \alpha\,\mathbf{n}_P \tag{21}$$

The choice of $\mathbf{t} = \mathbf{t}_P$ uniquely identifies one plane α (among ∞^1) which contains the unit normal vector $\mathbf{n}_P$ and a curve $\mathcal{C}$ upon Γ, as clearly shown in the left Fig. 20. While β sets the distance between $\mathbf{x}$ and $\mathbf{x}_P$ along the tangent direction $\mathbf{t}_P$ (thus establishing how far the new particle is from the seeder), the parameter α has to be determined in order to place the new oriented particle upon the circle $\mathcal{O}$ osculating $\mathcal{C}$ at $\mathbf{x}_P$. To this aim, Eq. (19) can be rewritten in terms of components, by exploiting Eqs. (2) and (21), in the following form

$$\phi = \phi_P + n_i|\nabla\phi|(\beta t_i + \alpha n_i) + \frac{1}{2}\phi_{,ij}(\beta t_i + \alpha n_i)(\beta t_j + \alpha n_j) = \alpha|\nabla\phi| + \frac{1}{2}\phi_{,ij}(\alpha^2 n_i n_j + 2\alpha\beta n_i t_j + \beta^2 t_i t_j) + O(\Delta^2) \tag{22}$$

Both the particles at $\mathbf{x}_P$ and at the new position $\mathbf{x}$ lie on Γ, so that $\phi_P = \phi = 0$. Moreover, from geometrical considerations we have

$$\alpha = \frac{\beta^2\tilde{\kappa}}{2} + O(\beta^3) \tag{23}$$

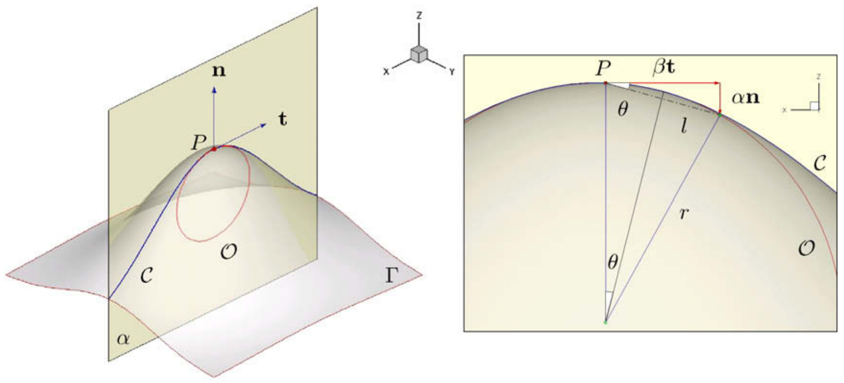


Fig. 20. On the left, the choice of a (unit) vector **t** tangent to Γ at $\mathbf{x}_P$ identifies the plane α (containing **t** and **n**) and a curve $\mathcal{C}$ on the interface, characterized by an osculating circle $\mathcal{O}$ at $\mathbf{x}_P$. The right figure shows the geometrical quantities used to evaluate the order of magnitude of the parameter α.

where $\tilde{\kappa}$ is the normal curvature, i.e. the curvature for the surface curve ($\mathcal{C}$) whose tangent is **t**. Eq. (23) can be proved by looking at the right Fig. 20, where a sketch of the circle $\mathcal{O}$ osculating $\mathcal{C}$ at $\mathbf{x}_P$ is reported. First of all, the arc length s is given by

$$s = 2r\theta = 2\theta\tilde{\kappa}^{-1}$$

whereas the chord from $\mathbf{x}_P$ to $\mathbf{x}$ is

$$l = 2r\sin\theta = 2r\sin\left(\frac{s}{2r}\right)$$

Then

$$\beta = l\cos\theta = 2r\sin\left(\frac{s}{2r}\right)\cos\left(\frac{s}{2r}\right) = r\sin\left(\frac{s}{r}\right) = s - \frac{1}{3!}\tilde{\kappa}^2 s^3 + O(s^5)$$

where, of course, the last equality arises from a series expansion of the sine function. Therefore, if we move along the curve of a quantity $s = O(\Delta)$, then $\beta = \Delta + O(\Delta^3)$. Consequently,

$$\alpha = \beta\tan\theta = \beta\tan\left(\frac{s}{2r}\right) = \beta\frac{s}{2r} + O(s^3) = s^2\frac{\tilde{\kappa}}{2} + O(s^3) = \beta^2\frac{\tilde{\kappa}}{2} + O(\beta^3)$$

Due to the evaluated order of magnitude of α, Eq. (22) can be rewritten up to second order in the form

$$\alpha|\nabla\phi| + \frac{1}{2}\phi_{,ij}t_it_j\beta^2 + O(\beta^3) = 0 \tag{24}$$

being $\beta = O(\Delta)$. From Eq. (20), we have

$$\phi_{,ij}t_it_j = |\nabla\phi|n_{i,j}t_it_j$$

so that (under the assumption $|\nabla\phi| \neq 0$)

$$\alpha + \frac{1}{2}n_{i,j}t_it_j\beta^2 + O(\beta^3) = 0 \tag{25}$$

where α depends only on known quantities. It's worth noting that, except for the sign, the term $n_{i,j}t_it_j$ exactly corresponds to the aforementioned normal curvature $\tilde{\kappa}$ of the circle osculating $\mathcal{C}$ at $\mathbf{x}_P$. In fact, from the Frenet formulae

$$\frac{d\mathbf{t}}{ds} = \hat{\kappa}\mathbf{n}$$

we have (being $\mathbf{t}\cdot\mathbf{n} = 0$)

$$\hat{\kappa} = \mathbf{n}\cdot\frac{d\mathbf{t}}{ds} = -\mathbf{t}\cdot\frac{d\mathbf{n}}{ds} = -\mathbf{t}\cdot(\nabla\mathbf{n}\cdot\mathbf{t}) \tag{26}$$

Finally, the coordinates of the new oriented particle on Γ are provided by Eq. (21) rewritten (in terms of components) in the form

$$x_i^{\text{new}} = x_i + \beta t_i - \frac{1}{2}n_{k,l}t_kt_ln_i = x_i + \beta t_i + \frac{1}{2}\tilde{\kappa}n_i \tag{27}$$

Once the position of the new particle is computed, its normal vector can be assumed equal to

$$n_i^{\mathrm{new}} = n_i + \frac{\partial n_i}{\partial x_j}\left(x_j^{\mathrm{new}} - x_j^P\right) \tag{28}$$

while for the curvature tensor we simply put

$$\nabla \mathbf{n}|_{\mathbf{x}^{\mathrm{new}}} \approx \nabla \mathbf{n}|_{\mathbf{x}_P} \tag{29}$$

As to the tangent unit vectors $\mathbf{t}$ to be used in the choice of the new particle, different strategies can be adopted. A good choice could be the computation of the eigenvectors of the shape operator in order to define the local tangent directions corresponding to the principal curvatures of Γ at $\mathbf{x}_P$. Alternatively, we can initialize the tangent vectors by simply choosing two orthogonal directions (for example, by Eqs. (16) and (17)) and setting their initial modulus to a suitable value (see next section). This approach is very simple and has been adopted in the present implementation. Although we can locate any particle by simply rotating a single tangent vector, in the application we have limited the seeding process to four new particles for each seeder, by using the aforementioned tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ and the corresponding opposite ones ($-\mathbf{t}_1$ and $-\mathbf{t}_2$).

The seeding procedure has been tested on a sphere, where the requested quantities have already been determined by treating the Enright problem. Fig. 21 shows a starting distribution of 800 markers located on a sphere with $r = 1$ and two different results achieved by the seeding algorithm, the first being obtained by setting the parameter β to 0.025 (center figure) and the second by setting it to 0.05 (right figure). As expected, a larger value for β scatters the new particles on a wider region and yields a more uniform distribution of markers; this is exactly what we want, in order to avoid the occurrence of interface regions affected by a dangerous lack of particles as much as possible. On the other hand, by moving far from $\mathbf{x}_P$, the numerical approximations related to the seeding process increase so that a suitable compromise is required, according to many other parameters (the spatial mesh resolution, the starting particles distribution, the time step, etc.).

7.2. The self-adaptive mechanism

At this stage, we can focus our attention on where to locate the new particles or, in other words, when to switch on the seeding process. The insertion of additional oriented particles is related to the stretching of the interface that causes a local depletion of markers. Intuitively, this stretching could be evaluated through the (maximum) distance between the markers, but such an approach is not convenient and would require a significant increase of the computational effort. Actually, we could limit the computation of this distance to each cell of the numerical domain or to a prescribed influence region around each particle, but, in any case, this approach would introduce an undesired link among the markers. For this reason, it is preferable to switch on the seeding process on the basis of a direct estimation of the local interface stretching. According to our discussion on the meaning and the behaviour of the tangent vector $\mathbf{t}$, at each point $\mathbf{x}_P$ this stretching can be estimated by

$$\sigma_P = |\mathbf{t}_P| \tag{30}$$

When σ_P exceeds a prescribed limit value σ_{max} the particle is recognized as a seeder and new particles around $\mathbf{x}_P$ are added upon Γ through the relations derived in the previous section. Obviously, the limit value σ_{max} characterizes the procedure and must be selected in a suitable way. Generally speaking, a small value can provide an excellent distribution of particles and a very accurate reconstruction of Γ, but also a notable increase of both the CPU time and the arrays storage requirements. On the other hand, a too large limit can make the seeding process ineffective. It's worth noting that the unstructured nature of the algorithm allows an easy handling of data, as the new quantities can be simply appended to the corresponding arrays, no re-ordering being needed.

In order to better understand the activation of the seeding procedure, let us focus our attention on the numerical simulation of an expanding sphere. To this aim, we account for a unit computational domain discretized by a 50^3 mesh and a centered sphere of radius $r = 0.1$, where we placed 554 oriented particles (23 markers on each of 24 semi-meridians, plus two markers on poles). Note that, at each point on the sphere, the starting (mean) curvature is equal to the sum of the two principal curvatures, so that $\kappa_0 = 2/r = 20$ at any $\mathbf{x}_P$. The front moves normal to itself ($u_i = |\mathbf{u}|\, n_i$, where we set $|\mathbf{u}| = 0.1$) so

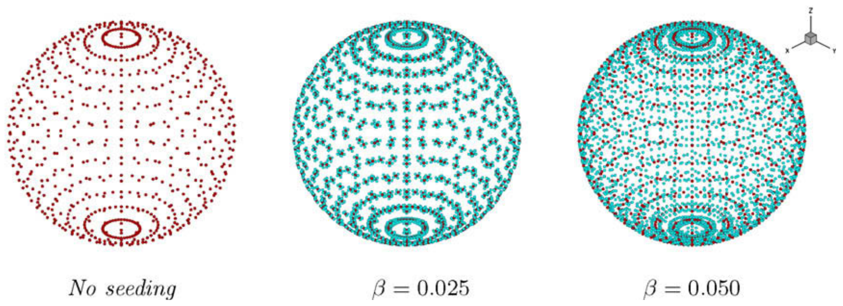


Fig. 21. Particles distributions provided by the seeding algorithm on a unit sphere for two different values for β . Each particle produces four markers.

that Eqs. (15) and (18) provide the starting values for $\mathbf{n}$ and $\nabla\mathbf{n}$, while the derivatives of the velocity to be used in Eqs. (7) and (9) are determined by

$$u_{j,k} = |\mathbf{u}|\left[\frac{1}{r}(\delta_{jk} - n_j n_k)\right]$$
$$u_{j,ks} = |\mathbf{u}|\left[-\frac{n_s}{r^2}\,\delta_{jk} - \frac{n_j}{r}\,n_{k,s} - \frac{n_k}{r}\,n_{j,s} + \frac{n_j n_k n_s}{r^2}\right] = -\frac{|\mathbf{u}|}{r}\,[n_j\,n_{k,s} + n_k\,n_{j,s} + n_s n_{j,k}] \tag{31}$$

The prescribed expanding motion causes the radius of the spherical front to increase, but the rather coarse distribution of particles used for the simulation soon becomes insufficient to determine the Level-Set function at the mesh nodes; thus, the interface is not tracked in the correct way. Fig. 22 shows the zero level of the ϕ function and the corresponding location of the oriented particles at $t = 0$ (on the left) and at the end of our simulation ($t = 3$). In particular, the center figure depicts the interface determined by the only use of the oriented particles and highlights the inadequacy of the starting distribution to follow the evolution of Γ. On the contrary, the right picture shows the same result as determined by the aid of the transport equation for ϕ: the resulting interface is certainly improved, but significant errors still affect the numerical solution.

In order to simulate the expanding sphere correctly, it's clear that we need to refine the markers distribution. To this aim, we account for a system of curvilinear coordinates ξ, η upon Γ_0 and set the (four) tangent vectors at the oriented particle $\mathbf{x}_P(\xi, \eta)$ in the following form

$$\mathbf{t}_1 = \frac{\mathbf{x}_{\xi+1,\eta} - \mathbf{x}_{\xi-1,\eta}}{2}, \quad \mathbf{t}_2 = \frac{\mathbf{x}_{\xi,\eta+1} - \mathbf{x}_{\xi,\eta-1}}{2}, \quad \mathbf{t}_3 = -\mathbf{t}_1, \quad \mathbf{t}_4 = -\mathbf{t}_2 \tag{32}$$

In this way, the module of the $\mathbf{t}$ vectors define the (averaged) distance of the particle from the neighbouring markers. Then, we switch on the seeding algorithm, by setting the following seeding criterium

$$|\mathbf{t}|_{max} = \sigma_{max} = \frac{\Delta}{2} \tag{33}$$

where Δ is the spatial cell size in our (uniform) Cartesian mesh. In practice, this condition forces the distance between two neighbouring particles to be less or equal to half a cell size: when the module of a $\mathbf{t}$ vector (evolving by Eq. (13)) exceeds this value, the corresponding oriented particle becomes a seeder. For simplicity, the tangent vectors of each new marker $\mathbf{x}_{new}$ are computed through Eqs. (16) and (17), while their module, as well as the value $|\mathbf{t}|$ of the seeder itself, are (re)set to the new distance $|\mathbf{x}_{new} - \mathbf{x}_P|$. Moreover, the added particles are placed around each seeder by setting the value β of Eq. (27) equal to $\Delta/3$. This mechanism has been proved to be very effective to refine the markers distribution upon Γ.

Left Fig. 23 shows an XZ-view of the starting interface Γ_0 and three subsequent configurations (at $t = 1, 2$ and 3), rendered by the zero level of the function ϕ and a suitable level of transparency so as to show the accuracy and correctness of the spherical evolving front. Moreover, the figure highlights a generic particle of the starting distribution, whose curvature, computed by Eq. (13), is reported in the right picture with the corresponding radius $r_P = 2/\kappa_P$ (scaled by a factor 10). As expected, the values of the increased radius matches the subsequent Γ configurations perfectly. The table in Fig. 23 also lists the overall number of markers used to determine the zero level of ϕ, and it's quite clear that the procedure guarantees a very fine particles distribution on Γ at each time step.

It is to be noted that the seeding is carried out by each marker in an independent way, so that an excessive concentration of particles can easily occur. Even though this behaviour does not affect the accuracy of the tracked interface, it represents a useless burden in terms of computational effort and should be somehow avoided. For this reason, the algorithm should also be able to recognize a useless gathering of particles and suitably de-seed the interface. The most intuitive way to realize this process consists of estimating the relative distance between any original particle and all the newly added markers, and remove these last ones when they are both too close and almost parallel, i.e.

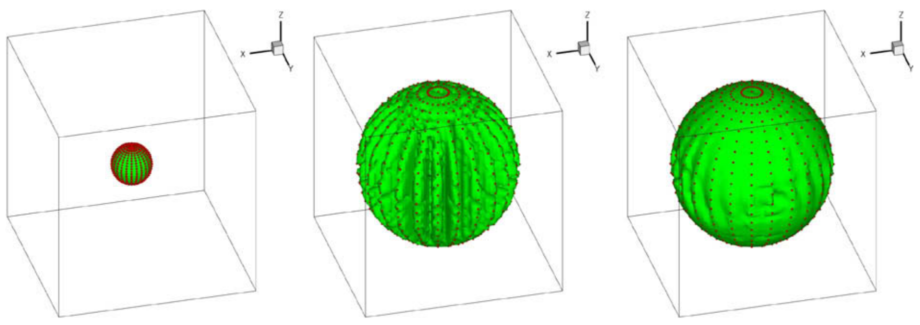
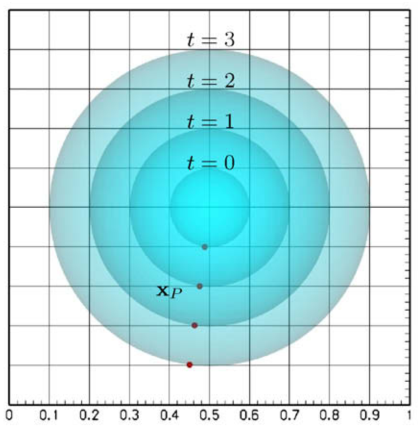
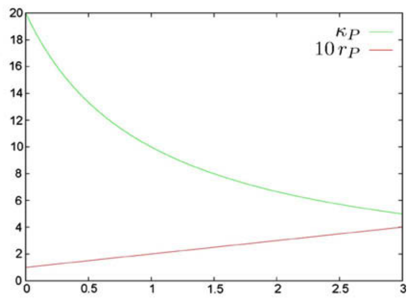


Fig. 22. The expanding sphere at $t = 0$ modeled by 554 oriented particles, and at $t = 3$ as determined by the only use of the markers (center figure) and the combined use of markers and the transport equation for ϕ (right figure).





T	N_p	κ_P	r_P
$0s$	554	20.000	0.1000
$1s$	12794	9.9956	0.2008
$2s$	63970	6.6636	0.3001
$3s$	319370	4.9947	0.4004

Fig. 23. The interface configurations of the expanding sphere at $t = 0$, 1, 2 and 3 as determined by the aid of the seeding procedure is reported on the left. The right picture shows the time histories of the curvature κ_P and the corresponding radius of the evolving front for the generic oriented particle at $\mathbf{x}_P$.

$$\begin{aligned} &|\mathbf{x}_p - \mathbf{x}_{add}| \leqslant d_{min} \\ &\mathbf{n}_p \cdot \mathbf{n}_{add} \geqslant \alpha_{min} \quad \text{with} \quad \alpha_{min} = O(1) \end{aligned} \tag{34}$$

If both these relations are satisfied, the added marker $\mathbf{x}_{add}$ does not provide any further information to trace the zero level of ϕ compared to the particle $\mathbf{x}_P$ and can be deleted. In other words, each original oriented particle also behaves as a potential *de-seeder* so as to optimize, at each time step, the (local) marker distribution upon Γ. Furthermore, this procedure preserves the starting distribution of markers.

As for σ_{max} in the seeding process, the parameters d_{min} and α_{min} make the de-seeding procedure more or less effective and must be selected according to the available computational and storing resources. It's worth noting, from a computational point of view, that the comparison between each starting particle with all the possible added ones can become very expensive. Then, in order to reduce the requested CPU time, it is convenient to code the procedure in a cell-by-cell way and to compare the quantities related to each $\mathbf{x}_P$ with the only markers added in the same cell. To this aim, the conditions (34) are implemented within the routine devoted to the evaluation of the Level-Set function, where a single loop on N_p (the number of markers) is used to determine the minimum distance between the cell vertices and the oriented particles. There, it is possible to identify the number of particles contained within each cell $\left(N_p^{i,j,k}\right)$ and enforce the de-seeding criteria by a double loop on the particles in each cell instead of the whole set, thus achieving a remarkable CPU saving.

Finally, left Fig. 24 shows the excellent simulation of the expanding sphere as computed by the self-adaptive particles mechanism, including both the seeding and de-seeding procedure. Still, the β parameter in Eq. (27) is set to $\Delta/3$, while

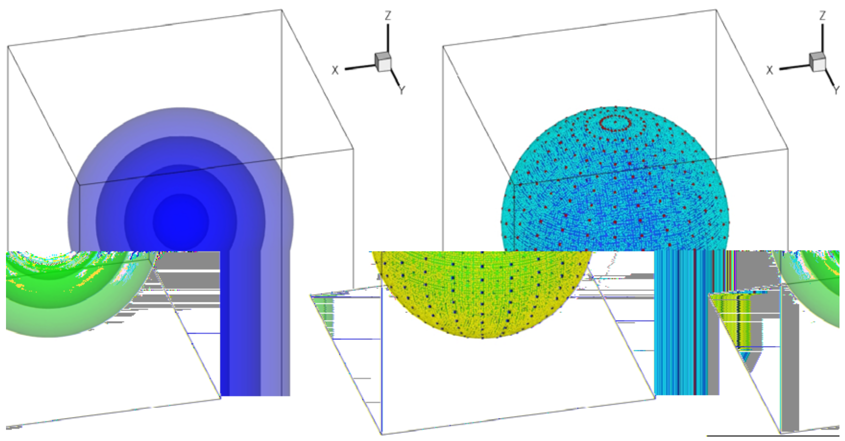


Fig. 24. On the left, the expanding motion is depicted by the computed interface at $t = 0$, 1, 2 and 3. The right figure reveals the actual particle distribution at $t = 3$, as determined by the seeding–de-seeding process.

Table 2
Number of oriented particles used by the self-adaptive algorithm to simulate the expanding sphere and corresponding CPU times.

Code/time	0 s	0.5 s	1s s	1.5 s	2s s	2.5 s	3 s	CPU
Seed	554	2674	12,794	63,874	63,970	319,370	319,370	37.5 m
Seed + de-seed	554	1992	6233	16,474	16,515	36,587	36,648	8.3 m

the de-seeding criteria fix the d_{min} and α_{min} values to $\Delta/6$ and 20 degrees, respectively. According to the CFL condition, the time step is set to 0.0025 and covers a time interval equal to 3 by 1200 steps. Compared to the aforementioned 319,370 particles on Γ at $t = 3$, the last configuration of Γ is traced by only 36,602 particles and is reported in the right Fig. 24. Table 2 summarizes the number of markers at different steps of the two simulations (with and without the de-seeding) and the resulting CPU time on the PC equipped with the Xeon processor at 2.5 Ghz. As expected, the gain in the computational effort is significant.

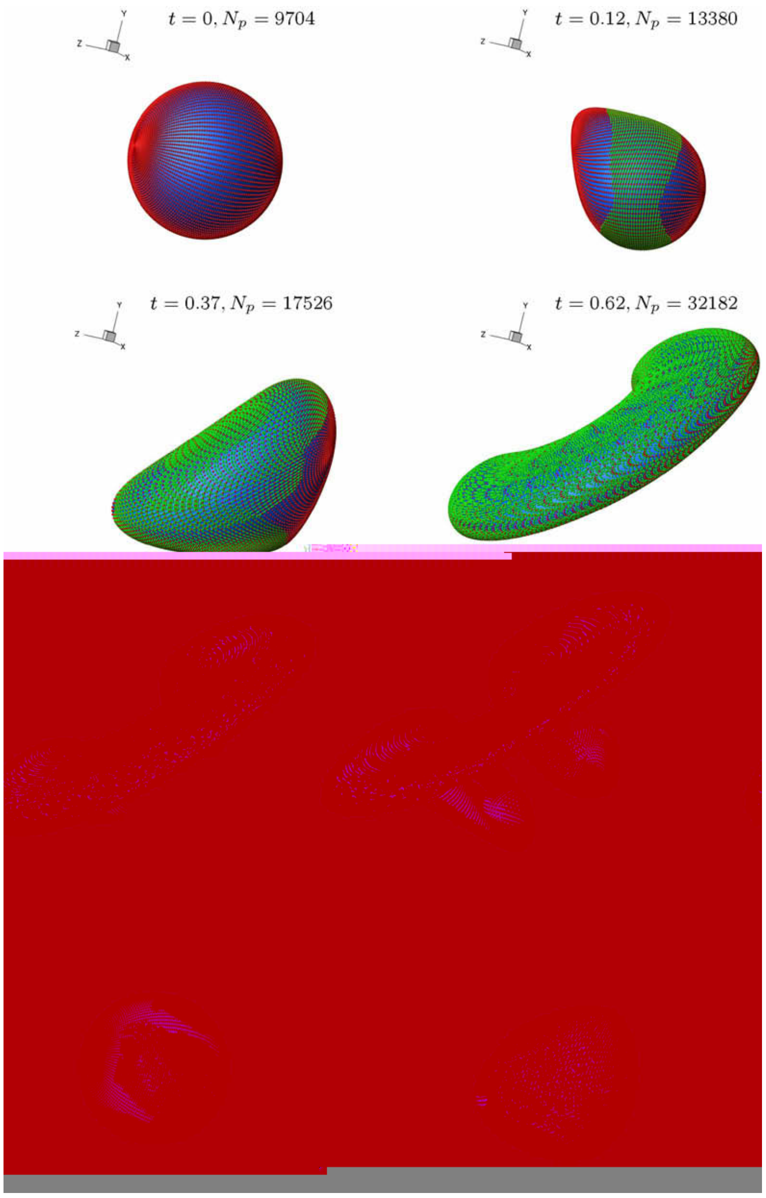


Fig. 25. Interface time evolution for the Enright test and the corresponding oriented particles distributions as determined by the self-adaptive procedure.

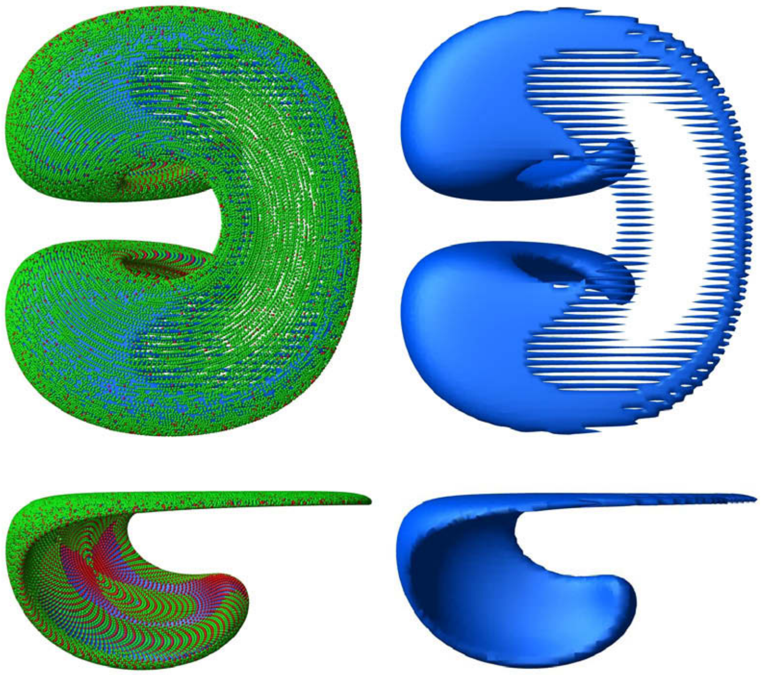


Fig. 26. Σ- and Λ-views of the Enright interface at $t = 1$, as determined by the self-adaptive procedure. Note, on the left, how the added green markers mainly arrange themselves upon the critical upper region of the interface. The resulting zero level of ϕ is reported on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7.3. The Enright test

Let's come back to the Enright test to check the capability of our self-adaptive oriented particle Level-Set method in dealing with a critical time evolution of Γ and solve the numerical problems outlined in Section 6. To this aim, we still account for the 100^3 Cartesian mesh and switch on the self-adaptive procedure used for the expanding sphere, by using the following values

$$\beta = \Delta/3, \quad \sigma_{max} = \Delta, \quad d_{min} = \Delta/5, \quad \alpha_{min} = 20^\circ$$

An initial distribution of only 9704 oriented particles is used (98 markers on 99 semi-meridians plus the two poles), so that the seeding procedure starts very soon to fit the particle pattern on the stretched interface. Fig. 25 shows a three-dimensional view of Γ at subsequent time steps. The red markers represent the starting distribution, while the green ones correspond to the particles added by the self-adaptive procedure; each frame points out the time and the overall number of particles used to determine the zero level of the ϕ function. The seeding process seems to work very well and provides an accurate reconstruction of the interface and a fine distribution of particles at each time step. It's worth noting the last frame, where the sphere is recomposed perfectly and a lot of added particles are still present; this result somehow proves the coarseness of the starting distribution of particles and, consequently, the effectiveness of the algorithm to refine the marker pattern. The Σ and Λ-views of both the particles distribution and the zero level of ϕ at $t = 1$ are reported in Fig. 26. These pictures should be compared with the analogous result in Fig. 16. As desired, the new markers mainly arrange themselves along the upper surface of the interface, which suffers the most pronounced stretching and a notable depletion of particles. No roughness appears in the central region of the manifold and the level of detail attained by the algorithm is very high, as the interface exhibits an excellent symmetry and, above all, the expected void representation due to the aforementioned limits of the spatial grid resolution.

It can be seen that any numerical problem related to the wrong estimation of the function $\mathcal{S}(\phi)$ is removed, so that no fictitious mass affect the tracked interface up to the end of the simulation. Note that such an excellent result is obtained by less than half the number of oriented particles used for the simulation corresponding to Fig. 16, with a large saving of CPU time. Finally, Fig. 27 shows the Σ-views of the interface for the Enright problem, here only traced by the zero level of the function ϕ.

8. Simulation of front merging by particles

At this stage, we come back to the problem summarized in Section 3, since we want to show that the occurrence of merging fronts can be modeled successfully by lagrangian particles placed on Γ, provided that their motion is assigned in the cor-

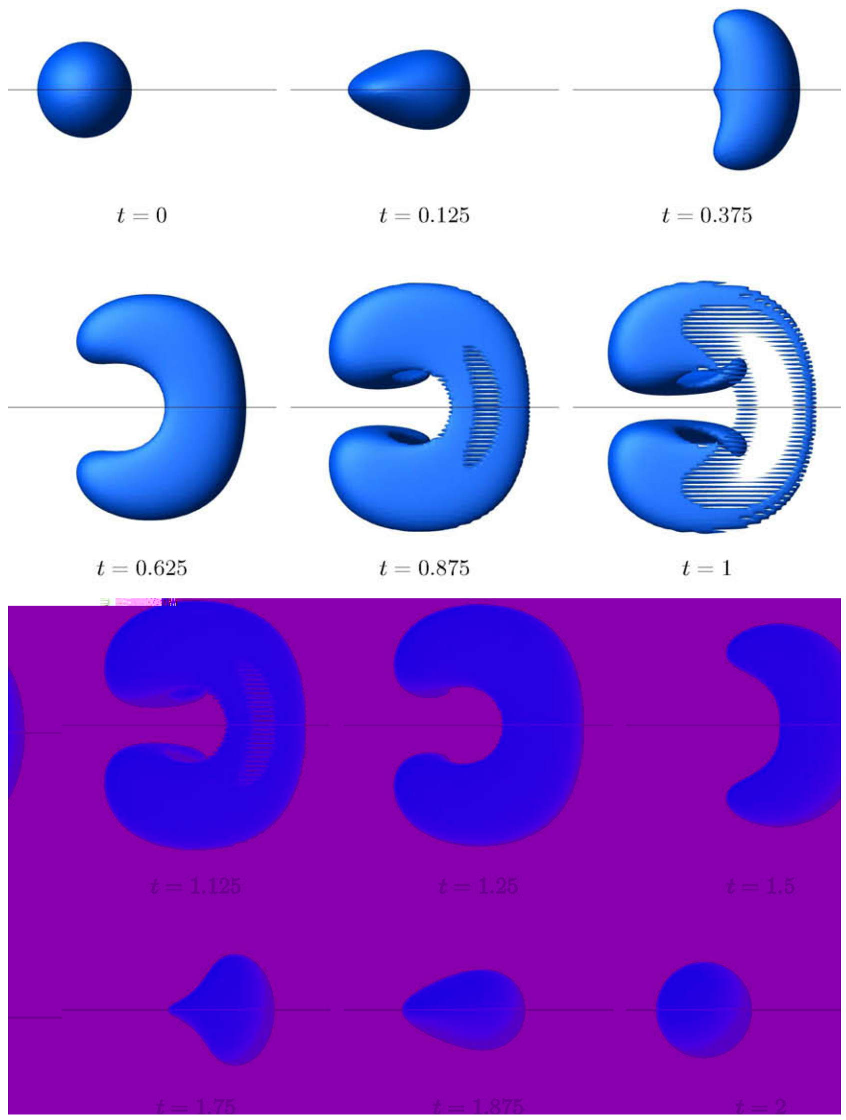


Fig. 27. Σ-views of the interface configurations for the Enright test, determined by using our self-adaptive oriented particles Level-Set method (100^3 uniform Cartesian mesh, starting distribution of 9704 markers, $\sigma_{max} = \Delta, \beta = \Delta/3, d_{min} = \Delta/5$ and $\alpha_{min} = 20^\circ$).

rect way. In fact, by a suitable modification of the de-seeding procedure the algorithm can achieve an effective estimation of an entropy solution, where at any time instant the position of each point on the front can be related to the initial condition uniquely. To this aim, we have to recognize the points on the interface exhibiting a merging behaviour (through the possible intersections of their own trajectories) and simply remove the corresponding particles.

Let $\mathbf{x}_A^n$ be the position at t^n of the particle A. The trajectory described by A from t^{n-1} to t^n can be approximated by

$$\mathbf{x}_A(\eta_A) = \mathbf{x}_A^{n-1} + \eta_A(\mathbf{x}_A^n - \mathbf{x}_A^{n-1}) \quad \text{with} \quad 0 \leqslant \eta_A \leqslant 1 \tag{35}$$

and analogously for any other particle B

$$\mathbf{x}_B(\eta_B) = \mathbf{x}_B^{n-1} + \eta_B(\mathbf{x}_B^n - \mathbf{x}_B^{n-1}) \quad \text{with} \quad 0 \leqslant \eta_B \leqslant 1 \tag{36}$$

By removing the constraints on η_A and η_B (thus accounting for $-\infty < \eta_A < +\infty$ and $-\infty < \eta_B < +\infty$), these two equations describes two straight lines in R^3, whose distance can be computed by minimizing the distance

$$d(\eta_A, \eta_B) = |\mathbf{x}_B(\eta_B) - \mathbf{x}_A(\eta_A)| \tag{37}$$

with respect to η_A and η_B. In the hypothesis that

$$\Sigma = |\mathbf{x}_B^n - \mathbf{x}_B^{n-1}|^2 |\mathbf{x}_A^n - \mathbf{x}_A^{n-1}|^2 - \left[\left(\mathbf{x}_A^n - \mathbf{x}_A^{n-1}\right) \cdot \left(\mathbf{x}_B^n - \mathbf{x}_B^{n-1}\right)\right]^2 \neq 0 \tag{38}$$

(i.e. the two lines are not parallel) the general solution of this problem is

$$\begin{aligned} \eta_A &= \frac{-(\mathbf{d}^{n-1} \cdot \delta\mathbf{x}_A)\,(\delta x_B)^2 + (\mathbf{d}^{n-1} \cdot \delta\mathbf{x}_B)\,\delta x_{AB}}{\Sigma} \\ \eta_B &= \frac{(\mathbf{d}^{n-1} \cdot \delta\mathbf{x}_B)\,(\delta x_A)^2 - (\mathbf{d}^{n-1} \cdot \delta\mathbf{x}_A)\,\delta x_{AB}}{\Sigma} \end{aligned} \tag{39}$$

where

$$\begin{aligned} &\mathbf{d}^{n-1} = \mathbf{x}_A^{n-1} - \mathbf{x}_B^{n-1} \\ &\delta\mathbf{x}_A = \mathbf{x}_A^n - \mathbf{x}_A^{n-1}; \quad \delta x_A = \left|\mathbf{x}_A^n - \mathbf{x}_A^{n-1}\right| \\ &\delta\mathbf{x}_B = \mathbf{x}_B^n - \mathbf{x}_B^{n-1}; \quad \delta x_B = \left|\mathbf{x}_B^n - \mathbf{x}_B^{n-1}\right| \\ &\delta x_{AB} = \left(\mathbf{x}_A^n - \mathbf{x}_A^{n-1}\right) \cdot \left(\mathbf{x}_B^n - \mathbf{x}_B^{n-1}\right) \end{aligned}$$

If η_A and η_B (computed with the above formulas) are such that

$$0 \leqslant \eta_A \leqslant 1 \quad \text{and} \quad 0 \leqslant \eta_B \leqslant 1 \tag{40}$$

the trajectories of the particles intersect (within $O(\Delta)$). In this case, a single valued (entropy) solution can be obtained by simply removing both markers A and B; consequently, conditions (40) can be adopted within our self-adaptive oriented particles approach as an additional de-seeding criteria to simulate front merging.

As discussed in Section 7, also the conditions (40) should be implemented in a cell-by-cell way to reduce the computational effort. Therefore, the intersections are determined by a double loop on the particles contained in each cell. In this case, however, the intersections must be computed for all the particles in each cell and also for the particles that occupied the

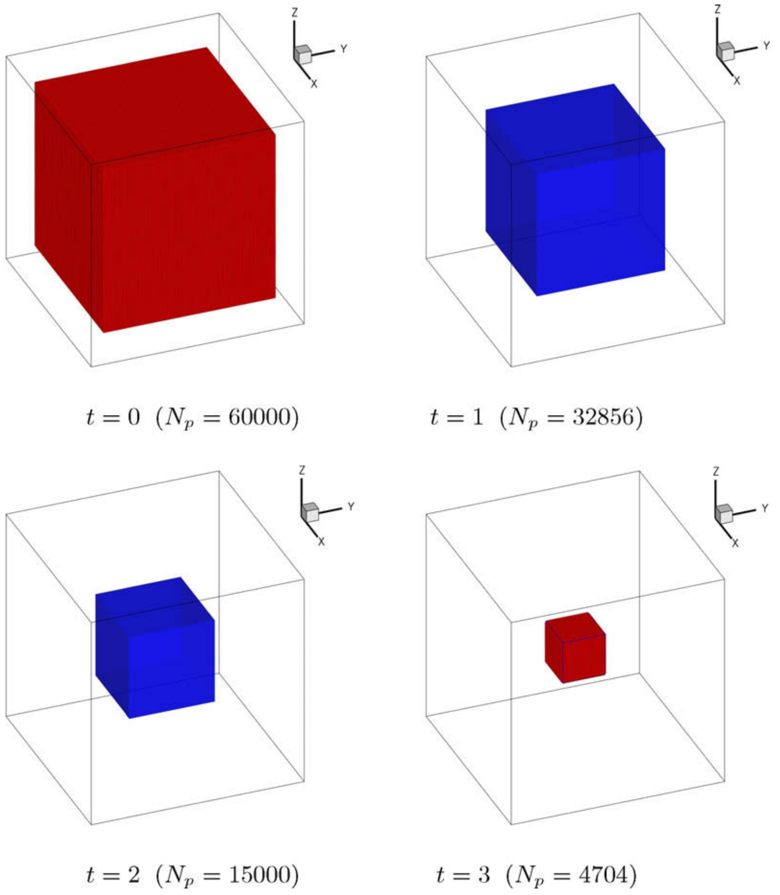


Fig. 28. The shrinking cube modeled by the self-adaptive method and the front merging de-seeding criteria given by relations (40). Each frame points out the (updated) number of particles determined by the de-seeding process and used to compute the zero level of ϕ.

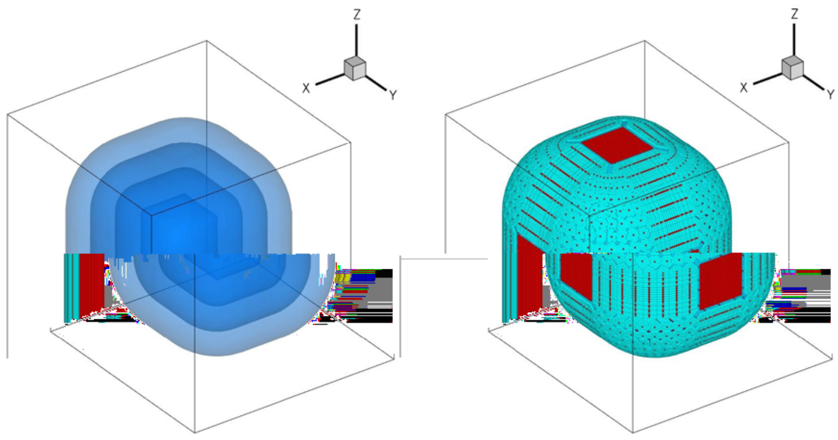


Fig. 29. The left figure shows the (inner) cube Γ_0 and the interface at $t = 1, 2, 3$ corresponding to the so-called entropy solution. On the right, the oriented particles are depicted with different colors: the red markers correspond to the starting distribution, whereas the cyan markers come from the seeding procedure. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

same cell at the previous time step, since the intersections could take place after the particles reached the cell of the Eulerian mesh. Note that the procedure requires the displacement of a particle not to exceed the cell size; then, a typical CFL condition

$$\Delta t \leqslant \frac{\Delta x}{\max(|\mathbf{u}|)} \tag{41}$$

must be enforced for the time step. Fig. 28 shows the interfaces determined for the shrinking cube test through the front merging de-seeding criteria. Of course, no particular constraint is considered here on the velocity field and each side of the cubic front simply moves normal to itself.

In the frames corresponding to $t = 0$ and $t = 3$ the (updated) particles distributions are superimposed to the zero level of ϕ: due to the de-seeding process, the simulation starts with 6×10^4 oriented particles and ends with only 4704 markers, still uniformly located on each side of the cube. As desired, no front distortion at corners or vertices appear and the shrinking motion is simulated perfectly. An entropy solution can be also determined for the expanding cube in a very simple way. In fact, the edges and vertices of the starting front can be modeled by a suitable patch of an elongated cylinder and a sphere, respectively, with a very small radius, in order to smooth the discontinuities affecting Γ_0. In these regions Eq. (18) is used to set the starting value of the shape operator (on the cube edges one component of $\mathbf{n}$ must be set to zero), while the corresponding derivatives of the velocity components can be still determined by Eq. (31) used for the (very similar) test of the expanding sphere. The initial distribution of particles on Γ_0 need not to be very fine, because the seeding procedure adds markers along the expanding front during the evolution.

The left Fig. 29 shows Γ_0 and the interfaces computed at three subsequent steps $t = 1, 2$ and 3. Still, the cube is centered in the unit computational domain (with $L_0 = 0.2$), the grid is 100^3, $|\mathbf{u}| = 0.1$ and the radius of the smoothing patches is set to $L/25$. As expected, the cube edges expands to a rounded shape whose radius grows in time. The right Fig. 33 highlights the oriented particles on Γ at the end of the simulation ($t = 3$), when the starting markers patterns (5376 particles) has been replaced by an overall distribution of 47,835 particles. These results demonstrate that the oriented particles can be successfully used to model any complex time evolution of the interface, even exhibiting a front merging characteristics.

9. Conclusions

A new method for tracking interfaces, based on a coupled use of the Level-Set ϕ function and massless particles, has been proposed. The high potentiality of a Lagrangian description provided by the markers is fully exploited, by linking the particles to the fundamental geometrical and topological properties of the interface, as the local normal vector and the corresponding shape operator. The particles enable a step-by-step evaluation of the Level-Set function by a direct solution of the Eikonal equation and to track the interface (as usual) through the ϕ zero level; in this way, no transport equation for ϕ needs to be solved and any diffusion problem related to the calculation of the spatial derivatives of ϕ and the re-initialization process is removed. The particles distribution at each step is modified, according to the deformations undergone by Γ during the motion, through an effective self-adaptive mechanism. Each marker behaves both as a seeder, able to enrich the pattern locally when the interface exhibits a notable stretching, and a de-seeder, in presence of an excessive (and useless) concentration of markers. Thus, a high accuracy to the interface reconstruction process can be achieved in a limited CPU time. The method has been successfully tested on different and standard problems, both in 2D and 3D, and will be soon inserted in a Navier–Stokes solver for complex hydrodynamic simulations.

References

[1] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface, Physics of Fluids 8 (1965) 2182–2189.
[2] S. Chen, D.B. Johnson, P.E. Raad, D. Fadda, The surface marker and micro-cell method, International Journal of Numerical Methods in Fluids 25 (1997) 749–778.
[3] C.W. Hirt, B.D. Nichols, Fundamentals of the KRAKEN Code, Report ucir-760 Lawrence Livermore National Laboratory, 1974.
[4] W. Noh, P. Woodward, SLIC – simple line interface calculation, in: A.V. Vooren, P. Zandbergen (Eds.) Proceedings of the Fifth International Conference on Fluid Dynamics, Lecture Notes in Physics, vol. 59, Springer, 1976, p. 330.
[5] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–225.
[6] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, Journal of Computational Physics 113 (1994) 134.
[7] D. Gueyffier, A. Nadim, J. Li, R. Scardovelli, S. Zaleski, Volume-of-fluid interface Trackinh with smoothed surface stress methods for three-dimensional flows, Journal of Computational Physics 152 (1998) 423–456.
[8] D.L. Youngs, Time-dependent Multi-material Flow with Large Fluid Distortion, Numerical Methods for Fluid Dynamics, Academic Press, New York, 1982.
[9] J.M. Martinez, X. Chesneau, B. Zeghmati, A new curvature technique calculation for surface tension contribution in PLIC-VOF method, Computational Mechanics 37 (2006) 182–193.
[10] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, Journal of Computational Physics 141 (1998) 112–152.
[11] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flows, Annual Review of Fluid Mechanics 31 (1999) 567–603.
[12] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithm based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[13] M. Sussman, P. Smereka, S. Osher, A level-set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159.
[14] M. Sussman, E. Fatemi, An efficient interface preserving level-set re-distancing algorithm and its application to interfacial incompressible fluid flow, SIAM Journal of Scientific Computing 20 (1999) 1165.
[15] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level-set method, Journal of Computational Physics 155 (1999) 410–438.
[16] G. Russo, P. Smereka, A remark on computing distance functions, Journal of Computational Physics 163 (2003) 51–67.
[17] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics 100 (1992) 25–37.
[18] E. Aulisa, S. Manservisi, R. Scardovelli, A mixed markers and VOF method for the reconstruction and advection of interfaces in two-phase and free-boundary flows, Journal of Computational Physics 188 (2003) 611–639.
[19] E. Aulisa, S. Manservisi, R. Scardovelli, A surface marker algorithm coupled to an area-preserving redistribution method for 3D interface tracking, Journal of Computational Physics 197 (2004) 555–584.
[20] M. Sussman, E. Puckett, A coupled level-set and VOF method for computing 3D and axisymmetric incompressible two-phase flows, Journal of Computational Physics 162 (2000) 301.
[21] M. Sussman, A second order coupled level-set and volume of fluid method for computing growth and collapse of vapour bubbles, Journal of Computational Physics 187 (2003) 110–136.
[22] M. Sussman, K.M. Smith, M.Y. Hussaini, M. Otha, R. Zhi-Wei, A sharp interface method for incompressible two-phase flows, Journal of Computational Physics 221 (2007) 469–505.
[23] D.J. Torres, J.U. Brackbill, The point-set method: front-tracking without connectivity, Journal of Computational Physics 165 (2000) 201–218.
[24] C. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (1977) 220–252.
[25] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (2005) 239–261.
[26] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and applications to non-spherical stars, Monthly Notices of the Royal Astronomical Society 181 (1977) 375–389.
[27] J.J. Monaghan, Simulating free surface flows with SPH, Journal of Computational Physics 110 (1994) 399–406.
[28] J.J. Monaghan, Smoothed particle hydrodynamics, Reports on Progress in Physics 68 (2005) 1703–1759.
[29] R. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), Journal of Computational Physics 152 (1999) 457–492.
[30] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level-set method for improved interface capturing, Journal of Computational Physics 183 (2002) 83–116.
[31] D. Enright, F. Losasso, R. Fedkiw, A fast and accurate semi-Lagrangian particle level-set method, Computers and Structures 83 (2005) 479–490.
[32] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Applied Mathematical Sciences, vol. 153, Springer, 2003.
[33] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science, Cambridge University Press, 1999.
[34] D. Adalsteinsson, J.A. Sethian, The fast construction of extension velocities in level-set methods, Journal of Computational Physics 148 (1999) 2–22.
[35] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, Journal of Computational Physics 31 (1979) 335–362.
[36] H. Zhao, A fast sweeping method for Eikonal equations, Mathematics of Computation 74 (250) (2004) 603–627.
[37] R. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM Journal of Numerical Analysis 33 (1996) 627.
[38] M. Kang, R. Fedkiw, X.D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Computational Physics 15 (2000) 323–360.