INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2015; **79**:654–675
Published online 30 July 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/fld.4070

Enablers for high-order level set methods in fluid mechanics

Francky Luddens[1,2,*,†], Michel Bergmann[1,2] and Lisl Weynans[1,2,3]

[1]*INRIA, F-33400 Talence, France*
[2]*IMB, Univ. Bordeaux, UMR 5251, F-33400 Talence, France*
[3]*CNRS, IMB, UMR 5251, F-33400 Talence, France*

SUMMARY

In the context of numerical simulations of multiphysics flows, accurate tracking of an interface and consistent computation of its geometric properties are crucial. In this paper, we investigate a level set technique that satisfies these requirements and ensures local third-order accuracy on the level set function (near the interface) and first-order accuracy on the curvature, even for long-time computations. The method is developed in a finite differences framework on Cartesian grids. As in usual level set strategies, reinitialization steps are involved. Several reinitialization algorithms are reviewed and mixed to design an accurate and fast reinitialization procedure. When coupled with a time evolution of the interface, the reinitialization procedure is performed only when there are too large deformations of the isocontours. This strategy limits the number of reinitialization steps and shows a good balance between accuracy and computational cost. Numerical results compare well with usual level set strategies and confirm the necessity of the reinitialization procedure, together with a limited number of reinitialization steps. Copyright © 2015 John Wiley & Sons, Ltd.

Received 26 November 2014; Revised 24 June 2015; Accepted 27 June 2015

KEY WORDS: level set; two-phase flows; immersed boundary; finite differences; Eulerian; fluid–solid interactions

1. INTRODUCTION

In multiphysics flows (multiphase flows, fluid–solid interactions), it is essential to be able to accurately track and represent the evolution of interfaces. For example, in a Lagrangian framework, particle methods (such as front tracking algorithms [1]) or conforming meshes [2] might be used. These techniques allow a very accurate localization of the interface but might require possibly complex reseeding or remeshing algorithms, especially for large deformations of the interface. They can be mixed with Eulerian methods and the ALE method introduced by Hirt *et al.* [3]. In an Eulerian framework, an efficient way of capturing interfaces is the level set method, introduced by Osher and Sethian [4]. It consists of the capture of an interface through the 0-isocontour of a higher-dimensional function. This eases the treatment of topological changes and can also easily be carried on in parallel codes, using domain decomposition, for example.

In several applications, an accurate computation of the geometric properties of the interface is required: for example, the computation of the normal to the interface for methods with second-order penalty terms in fluid–solid interactions [5] or for computations on numerical domains that do not fit the physical domain [6]. For the cases of multiphysics flows where the surface tension plays an important role, accurate computation of the curvature is a key stone. In general, distortions of the level set function ϕ might lead to significant errors in computing the geometric properties. One possible way to avoid large errors is the use of the distance function instead of a generic level set function. Unfortunately, even with a distance function as an initial state, ϕ can derive far

*Correspondence to: Francky Luddens, INRIA Bordeaux Sud-Ouest, Team MC2, 33405 Talence, France.
†E-mail: Francky.luddens@inria.fr

from the distance function. To circumvent the problem, Sussman *et al.* [7] introduced a reinitialization algorithm; it consists of an algorithm capable of recomputing the distance function. The current level set function is then replaced by this simpler function for the advection part. The reinitialization procedure may also be used in the context of the continuum surface force method developed by Brackbill *et al.* [8], where the interface is diffuse: it allows to keep a constant width for the interface.

Keeping the level set function close to a distance function is a key point in many level set algorithms. Besides the reinitialization procedure, other methods have been developed over the years. One alternative is the use of the so-called extension velocity: the velocity field is modified in the domain to ensure that the transport step will keep the distance property (e.g. [9, 10]). Another way to keep the distance property is the use of an extra term in the transport equation, similar to the reinitialization procedures. The idea is to couple advection and reinitialization in a single step (e.g. [11–15]). In this paper, we focus on the standard reinitialization strategy, where advection and reinitialization are treated separately, so that the evaluation of the errors can be performed independently.

Several methods have been developed over the years to perform the reinitialization step. It is based on the resolution of the eikonal equation

$$|\nabla\phi| = 1,$$

by either searching a stationary solution to a time-dependent Hamilton–Jacobi equation [7] or using a Gauss–Seidel-based method as in fast marching methods [16, 17] or fast sweeping methods [18]. The original method from [7] can be modified to ensure at least first-order accuracy on the curvature of the interface, but it is CPU consuming. The fast marching or fast sweeping methods are, on the contrary, much quicker but might suffer from a lack of accuracy.

The usual level set strategy for an evolving interface is the following:

- Use a transport equation to update ϕ.
- From time to time, reinitialize ϕ with the signed distance function: this step is useful for geometric properties computation, and it also eases the transport step, as the advected function is simpler and smoother.

Although the reinitialization procedure may improve mass conservation [7], it also introduces some error by slightly moving the interface [19]. It is then unclear what the frequency of these reinitialization steps should be.

In this paper, we investigate a (not too expensive) level set technique that ensures a good representation of the interface, as well as its geometrical properties, even for long times. The method is developed in a finite differences framework on Cartesian grids. It has been developed in such a way that parallelization should be easily and efficiently available. The strategy is based on a limited number of reinitialization steps. These steps are chosen according to the deformation of the isocontours of the level set function. They are necessary to prevent creation of small and/or large gradients near the interface, which could lead to numerical instabilities or increasing error on geometric computations. The method presented in this paper has been developed to handle smooth interfaces only. In some applications in fluid mechanics, singularities may appear, such as breakups (e.g. [20]) or corners, where the curvature becomes infinite. In those cases, the accuracy of the method would drop because of the limiters on the numerical curvature. A better representation of such phenomena would require special treatments of the singularities and is out of the scope of the present paper.

The paper is organized as follows: In Section 2, we present the framework and the objectives we want to reach using a level set technique. In Section 3, we discuss high-order reinitialization schemes, and we design a method with a balance between accuracy and computational cost. The strategy for the coupling transport/reinitialization is discussed in Section 4. We conclude in Section 5 with a discussion about the numerical results we obtained.

2. MOTIVATIONS

2.1. Framework

Let $\Omega = [x_{\min}; x_{\max}] \times [y_{\min}; y_{\max}]$ be a 2D rectangular domain in which we want to track the evolution of an interface Γ. Let us denote by Γ_0 the initial interface. We choose to capture Γ through the zero isocontour of a level set function ϕ. $\boldsymbol{U}(\mathbf{x}, t)$ denotes the velocity field for the advection of Γ. In order to mimic multiphysics flows, we assume that $\boldsymbol{U}$ is available in the whole domain Ω, either with explicit expressions or obtained by separate computations. As we are mainly interested in incompressible fluid mechanics, we will often use divergence-free velocity fields. In this context, the evolution of Γ is recovered from the evolution of ϕ, which satisfies the non-conservative equation

$$\partial_t \phi + \boldsymbol{U} \cdot \nabla \phi = 0, \tag{2.1}$$

$$\phi_{|t=0} = d_0, \tag{2.2}$$

or the conservative form, when $\nabla \cdot \boldsymbol{U} = 0$,

$$\partial_t \phi + \nabla \cdot (\phi \boldsymbol{U}) = 0, \tag{2.3}$$

$$\phi_{|t=0} = d_0, \tag{2.4}$$

where d_0 is the signed distance function to Γ_0. Similarly, we will use the notation d for the signed distance to Γ. Finally, we denote by $\boldsymbol{n}$ (resp. κ) the normal vector (resp. curvature) on the interface Γ.

The governing equations are discretized using a Cartesian mesh: we denote by N_x and N_y the number of nodes in each direction. The mesh size in each direction is defined by

$$\Delta x = \frac{x_{\max} - x_{\min}}{N_x - 1}, \qquad \Delta y = \frac{y_{\max} - y_{\min}}{N_y - 1}.$$

The nodes are denoted by $\boldsymbol{x}_{i,j} := (x_i, y_j)$ for $i = 1, \cdots, N_x$ and $j = 1, \cdots, N_y$. We choose a uniform spacing for the nodes, that is,

$$x_i = x_{\min} + (i-1)\Delta x, \qquad y_j = y_{\min} + (j-1)\Delta y.$$

For a given field F, we obviously denote by $F_{i,j}$ its approximate value at node $\boldsymbol{x}_{i,j}$. For the sake of simplicity, we often assume that $\Delta x = \Delta y$, but the ideas presented here can be carried out with different mesh sizes. We sometimes also denote by h the typical mesh size (i.e. $h = \Delta x$).

We work on this Cartesian grid in a finite differences framework. We then introduce the notations $D_x^{\pm}$ and $D_y^{\pm}$ for the one-sided derivatives of a field. The definition of these quantities will differ depending on the equation we are solving and the accuracy we want to reach.

2.2. Geometric properties computation

In the level set framework, the computation of the normal and the curvature of Γ is straightforward. Recall that the normal $\boldsymbol{n}$ and the curvature κ are given by

$$\boldsymbol{n} := \frac{\nabla \phi}{|\nabla \phi|}, \qquad \kappa := \nabla \cdot \boldsymbol{n}. \tag{2.5}$$

If ϕ is the signed distance function, owing to the property $|\nabla \phi| = 1$, these expressions read in an even simpler way $\boldsymbol{n} = \nabla \phi$ and $\kappa = \nabla \phi$. Owing to the definitions of $\boldsymbol{n}$ and κ, we rewrite (2.5) as

$$\kappa = \frac{\phi_x^2 \phi_{yy} + \phi_y^2 \phi_{xx} - 2\phi_x \phi_y \phi_{xy}}{\left(\phi_x^2 + \phi_y^2\right)^{3/2}}, \tag{2.6}$$

where ϕ_x is the first derivative of ϕ with respect to the first variables (and similar definitions for ϕ_y, ϕ_{xx}, ϕ_{xy} and ϕ_{yy}).

For numerical computations, the amplitude of the error depends on the chosen level set function. Let us illustrate this fact on a simple circle of radius $R = 0.6$. We consider the following setting:

$$\Omega := [-1;1]^2, \tag{2.7}$$

$$\Gamma := \left\{x^2 + y^2 = R^2\right\}, \tag{2.8}$$

$$d(x,y) = \sqrt{x^2+y^2} - R, \tag{2.9}$$

and we introduce a level set function ϕ_0 that vanishes on Γ,

$$\phi_0(x,y) := \frac{d(x,y)}{R}\left(\epsilon + (x-x_p)^2 + (y-y_p)^2\right), \tag{2.10}$$

with $\epsilon = 0.02$, $x_p = 0.7$ and $y_p = 0.4$. We compute the curvature on Γ in three different ways:

1. computing $\nabla\left(\frac{\nabla\phi_0}{|\nabla\phi_0|}\right)$, using (2.6),
2. computing $\nabla\left(\frac{\nabla d}{|\nabla d|}\right)$, using (2.6), and
3. computing Δd.

The last two computations are supposed to be equivalent, but the numerical approximations differ. We use centered second-order finite differences formulas to compute the derivatives, that is,

$$\phi_{x,i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \qquad \phi_{y,i,j} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}, \tag{2.11}$$

$$\phi_{xx,i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2}, \qquad \phi_{yy,i,j} = \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2}, \tag{2.12}$$

$$\phi_{xy,i,j} = \frac{\phi_{i+1,j+1} + \phi_{i-1,j-1} - \phi_{i-1,j+1} - \phi_{i+1,j-1}}{4\Delta x\Delta y}. \tag{2.13}$$

The L^∞ error on the curvature for different mesh sizes is represented in Figure 1. All three computations exhibit the expected second-order accuracy, but the amplitude of the error is much lower in the case of d (reduction by a factor of 100). This example shows that if the level set function has strong and/or weak gradients near the interface, the accuracy of the computation of the curvature is deteriorated. Even though the three computations show a second-order convergence, it confirms that it is interesting to work with distance functions, as it provides the *lowest amplitude* on the error.

Remark 1
In the advection step and in the reinitialization step, there is a need to use decentered stencils in order to respect the direction of propagation of the information. This is not the case for the computation of the curvature alone. Moreover, the curvature is a scalar value that we want to evaluate, but we do not use it as part of the schemes. Therefore, we can use a different stencil, and we choose (2.11)–(2.12)–(2.13) to keep a good accuracy with a compact stencil.

The rest of this paper is devoted to the design of a fast level set strategy involving reinitialization steps that satisfies two requirements:

- ensure at least first-order accuracy for the computation of κ and
- provide a small amplitude of the error.

The first step is the design of a fast and accurate reinitialization procedure for static interfaces. The second step is the development of a strategy for the coupling transport/reinitialization, in order to accurately track moving interfaces. The challenge is to carefully control the gradient of the numerical level set function without degrading the accuracy of the numerical solution.

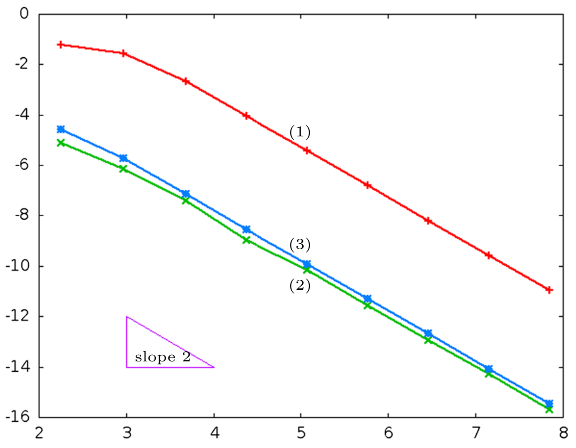

Figure 1. L^{∞} error on the curvature of the interface versus $1/\Delta x$ (log–log representation): curvature of the original level set (1), curvature of the distance function (2) and Laplacian of the distance function (3).

3. HIGH-ORDER REINITIALIZATION SCHEMES

In order to ensure numerical stability of the computations, reinitialization procedures are commonly used [7]: it consists in replacing the current level set function by the signed distance function. Given an original level set function ϕ_0, the aim is to find ϕ such that the eikonal equation is satisfied, that is,

$$|\nabla\phi| = 1, \tag{3.1}$$

$$\phi = 0 \text{ on } \{\phi_0 = 0\}\,. \tag{3.2}$$

A high-order accuracy near the interface is required to ensure good approximations of the geometric properties. We want at least first-order accuracy on κ, which means that we need at least third-order accuracy on ϕ (for a classical scheme) near Γ. Several methods, such as relaxation methods, fast sweeping methods or fast marching methods, have been designed to solve the eikonal equation. We start by reviewing the first two and develop a mixed method to obtain a fast reinitialization procedure. The fast sweeping step might be replaced by a fast marching step, but this issue is out of the scope of the present paper.

3.1. *Relaxation method*

Relaxation methods are based on the resolution of the unsteady PDE

$$\partial_\tau\phi + \operatorname{sign}(\phi_0)\left(|\nabla\phi| - 1\right) = 0, \tag{3.3}$$

where τ is a fictitious time, until the steady state is reached [7]. This equation is analogous to a non-linear transport equation with source term:

$$\partial_\tau\phi + \operatorname{sign}(\phi_0)\frac{\nabla\phi}{|\nabla\phi|}\cdot\nabla\phi = \operatorname{sign}(\phi_0). \tag{3.4}$$

The velocity for the transport is of magnitude 1, which means that the steady state should be reached for $\tau \sim L_{\max}$, where $L_{\max}$ is the maximum distance between the domain boundary and the interface. This also implies a CFL condition for iteratively solving (3.3), which is $\Delta\tau \leqslant \Delta x$. This condition implies that the number of iterations needed for convergence is multiplied by 2 when the mesh size is divided by 2. This method is then *a priori* rather slow. In practice, we use $\Delta\tau = \Delta x/2$ and a third-order Runge–Kutta total variation diminishing (TVD) scheme for pseudo-time evolution [21].

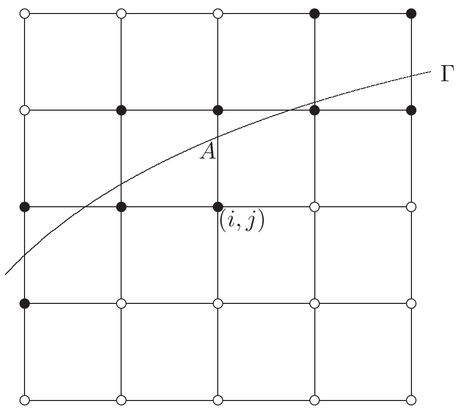

Figure 2. Illustration for the subcell fix. Plain circles stand for nodes close to the interface, and empty circles are nodes away from the interface. For example, the stencil used for $D_y^{\pm}\phi_{i,j}$ is $(\boldsymbol{x}_{i,j-2}; \boldsymbol{x}_{i,j-1}; \boldsymbol{x}_{i,j}; \boldsymbol{x}_A; \boldsymbol{x}_{i,j+1})$.

The relaxed eikonal Eq. (3.3) is a particular case of a Hamilton–Jacobi equation; numerical schemes for such equations might be found in [22]. In our particular case, we rewrite (3.3) in the semi-discrete form

$$\partial_\tau \phi + S(\phi_0) H_G(D_x^+\phi, D_x^-\phi, D_y^+\phi, D_y^-\phi) = 0,$$

where H_G is the Godunov flux defined as

$$H_G(a,b,c,d) := \begin{cases} \sqrt{\max(a,-b,0)^2 + \max(c,-d,0)^2} & \text{if } \operatorname{sign}(\phi_0) \leqslant 0, \\ \sqrt{\max(-a,b,0)^2 + \max(-c,d,0)^2} & \text{if } \operatorname{sign}(\phi_0) > 0. \end{cases}$$

$S(\phi_0)$ is a smoothed sign function defined by

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + |\nabla\phi_0|^2(\Delta x)^2}}.$$

The accuracy of the algorithm depends on the computation of the one-sided derivatives $D_x^{\pm}\phi$ and $D_y^{\pm}\phi$ used in the Godunov flux. No matter which method is used for the computation, the scheme slightly moves the interface away from its original position. It has been showed by Russo and Smereka [19] that the use of a standard WENO-5 scheme leads to a displacement that increases with the number of iterations. To improve the accuracy of the method, it is important to be able to limit this displacement. As most of it comes from the stencil crossing the interface, they introduced the so-called subcell fix that ensure the displacement will stop after a certain number of iterations. Using the subcell fix, they obtained a globally second-order accurate method. This idea has been carried on in [23] where a third-order scheme is developed. The choice of $D_x^{\pm}\phi$ and $D_y^{\pm}\phi$ is as follows:

- if the point is close to the interface (Figure 2), use a non-uniform stencil with five points and an ENO-3 scheme, detailed in [23];
- otherwise, use the standard WENO-5 scheme, see [24] for the original scheme or [25] for finite differences WENO schemes.

3.2. *Fast sweeping methods*

Fast sweeping methods ([18] for the original method or [26] for high-order schemes) are based on Gauss–Seidel-like iterations to solve the eikonal equation. The grid is swept in four alternating directions, namely

$$\begin{aligned}
&(1)\ i = 1, \cdots n_x,\ j = 1, \cdots n_y,\\
&(2)\ i = n_x, \cdots 1,\ j = 1, \cdots n_y,\\
&(3)\ i = n_x, \cdots 1,\ j = n_y, \cdots 1,\\
&(4)\ i = 1, \cdots n_x,\ j = n_y, \cdots 1,
\end{aligned}$$

and the values of ϕ are updated on the fly. Although fast sweeping methods are not suited for unstructured grids, they are very efficient on Cartesian grids and likely to be efficiently used in a parallel environment [27]. The idea is similar to fast marching methods (e.g. [17]), but instead of following characteristics flowing from Γ, the nodes are updated when the characteristics are in the direction of sweeping. The standard first-order method, denoted here by `fs1`, to update $\phi_{i,j}$ is the following: freeze all values of ϕ and find $\phi_{i,j}^{new}$ such that

$$\left[\left(s\left(\frac{\phi_{i,j}^{new} - \phi_{\min}^{x}}{\Delta x}\right)\right)^{+}\right]^{2} + \left[\left(s\left(\frac{\phi_{i,j}^{new} - \phi_{\min}^{y}}{\Delta x}\right)\right)^{+}\right]^{2} = 1, \tag{3.5}$$

where $(x)^+ := \max(x, 0)$, $s = \mathrm{sign}(\phi_{i,j})$ and $\phi_{\min}^x$ is defined by

$$\phi_{\min}^x := s \min(s\phi_{i-1,j}\ ,\ s\phi_{i+1,j}).$$

$\phi_{\min}^y$ is defined in a similar way. This first-order scheme is purely upwind and monotone, so that it requires a number of iterations that is independent of the mesh size [28]. In [26], higher-order methods are achieved, denoted here by `fs3Z`, using

$$\phi_{\min}^x := s \min(s(\phi_{i,j} - \Delta x D_x^- \phi_{i,j}); s(\phi_{i,j} + \Delta x D_x^+ \phi_{i,j})),$$

and a WENO-3 scheme for $D_x^\pm$ and $D_y^\pm$. Unfortunately, it requires a number of iterations that increases with the number of degrees of freedom.

We designed a second-order method, `fs2`, that turned out to have a stable number of iterations, by replacing the expressions in (3.5) by

$$\left[\left(s\left(\frac{\phi_{i,j}^{new} - \phi_{\min}^{x}}{\alpha\Delta x}\right)\right)^{+}\right]^{2} + \left[\left(s\left(\frac{\phi_{i,j}^{new} - \phi_{\min}^{y}}{\alpha\Delta x}\right)\right)^{+}\right]^{2} = 1, \tag{3.6}$$

with $\alpha = \frac{2}{3}$ and

$$\phi_{\min}^x := \begin{cases} \frac{4}{3}\phi_{i-1,j} - \frac{1}{3}\phi_{i-2,j} & \text{if } s\phi_{i-1,j} \leqslant s\phi_{i+1,j} \\ \frac{4}{3}\phi_{i+1,j} - \frac{1}{3}\phi_{i+2,j} & \text{otherwise} \end{cases}$$

$$\phi_{\min}^y := \begin{cases} \frac{4}{3}\phi_{i,j-1} - \frac{1}{3}\phi_{i,j-2} & \text{if } s\phi_{i,j-1} \leqslant s\phi_{i,j+1} \\ \frac{4}{3}\phi_{i,j+1} - \frac{1}{3}\phi_{i,j+2} & \text{otherwise} \end{cases}.$$

The complete fast sweeping scheme is as follows:

- *Initialization near* Γ: Set values in the vicinity of the interface, for example, in the band $B_n := \{d(x,\Gamma) \leqslant 5\Delta x\}$. These values remain untouched after.
- *Initialization of other values*: For the method `fs1`, the other values are set to $\pm 10^8$ depending on the sign of d [18]. For higher-order methods `fs2` and `fs3Z`, we use the first-order approximation given by `fs1` as the initial state.
- *Iteration*: Update ϕ elsewhere by sweeping the grid in four alternating directions,
- *Convergence*: Stop the computation when the L^1 norm between two successive iterations is small enough.

Table I. Comparison of three fast sweeping methods (Γ is a circle).

	fs1			fs3Z			fs2		
$\frac{1}{\Delta x}$	Error	coc	N_{it}	Error	coc	N_{it}	Error	coc	N_{it}
20	1.21E−02	—	3	3.85E−04	—	17	1.56E−03	—	7
40	6.29E−03	0.91	3	5.25E−05	2.77	28	5.44E−04	1.47	7
80	3.18E−03	0.97	3	7.41E−06	2.77	27	1.58E−04	1.75	8
160	1.59E−03	0.99	3	1.04E−06	2.81	33	4.27E−05	1.87	8
320	8.02E−04	0.99	3	1.44E−07	2.83	43	1.11E−05	1.94	7
640	4.01E−04	1.00	3	1.99E−08	2.85	63	2.82E−06	1.97	7

The error is $\|\phi - d\|_{L^1(\Omega)}$. ‘coc’ stands for computed order of convergence, and we report the total number of iterations of the fast sweeping algorithm N_{it}: 1 fast sweeping iteration corresponds to 1 sweep in *each* direction.

This method fs2 is only second order accurate, as it consists in the resolution of

$$\max(sD_x^-\phi_{i,j}; -sD_x^+\phi_{i,j}; 0)^2 + \max(sD_y^-\phi_{i,j}; -sD_y^+\phi_{i,j}; 0)^2 = 1,$$

where $D_x^{\pm}\phi_{i,j}$ and $D_y^{\pm}\phi_{i,j}$ are defined with a decentered second-order finite differences formula, that is,

$$D_x^-\phi_{i,j} = \frac{3\phi_{i,j} - 4\phi_{i-1,j} + \phi_{i-2,j}}{2\Delta x}, \qquad D_y^-\phi_{i,j} = \frac{3\phi_{i,j} - 4\phi_{i,j-1} + \phi_{i,j-2}}{2\Delta y},$$
$$D_x^+\phi_{i,j} = \frac{-3\phi_{i,j} + 4\phi_{i+1,j} - \phi_{i+2,j}}{2\Delta x}, \quad D_y^+\phi_{i,j} = \frac{-3\phi_{i,j} + 4\phi_{i,j+1} + \phi_{i,j+2}}{2\Delta y}.$$

Methods fs3Z and fs2 are not based on the same idea, and it is not clear whether a third-order method built similarly to fs2 is tractable or not, although similar methods have been used in the context of fast marching algorithms [29, 30]. The three methods fs1, fs2 and fs3Z can be written in the form (3.6). Unlike fs1 and fs2, the computation of $\phi^x_{\min}$ and $\phi^y_{\min}$ in fs3Z requires the current value $\phi_{i,j}$. This kind of relaxation effect might explain why the number of iterations increases when h decreases in fs3Z, while it remains constant for fs1 and fs2.

We illustrate the performances of this second-order fast sweeping on the following example: Ω is the square $[-1; 1]$, and Γ is the circle with centre $(0, 0)$ and radius 0.5. We initialize the fast sweeping algorithm by setting the distance function in the narrow band B_n, and we recall that these values remain fixed during the computation. We compare the results obtained with the methods fs1, fs3Z and fs2. Note that for second-order and third-order methods, we first apply fs1 in order to have a suitable initial state. We compute the L^1 error on ϕ on the whole domain Ω and show the results in Table I.

As expected, this method almost reaches a globally second-order accuracy. Compared with the high-order method in [26], the number of iterations needed to achieve convergence is relatively small and stable, that is, does not depend on the mesh size. As the second-order scheme is not very expensive compared with the first order, it will be handful to quickly (and rather accurately) compute the distance function in regions far enough from the interface.

3.3. Mixed method

The main goal we want to achieve is the construction of a fast method to solve the eikonal equation, with high-order accuracy in the vicinity of the interface, where geometric properties are of interest.

We have showed that the relaxation algorithm may be very accurate near the interface, but it is slower than the fast sweeping approach. However, the fast sweeping method requires a good initial guess near the interface. It is then natural to couple the two methods in the following way:

- *Initialization:* We use the relaxation method to fix the values of ϕ in the narrow band B_n; for this step, we apply the relaxation algorithm on the entire domain and stop the computation when the accuracy near the interface is good enough.

- *Fast sweeping:* We use the second-order fast sweeping method to compute the distance function far from the interface.

In order to obtain a good approximation of the distance function near the interface during the initialization step, we only need to control the accuracy on B_n. The relaxation method has been shown to be third order accurate near the interface [23], so we stop the computation when

$$\|\phi^{(n+1)} - \phi^{(n)}\|_{L^1(B_n)} \leqslant C(\Delta x)^3, \tag{3.7}$$

where C is a user defined constant. C should depend on the curvature (or its derivatives), but we usually use the numerical value $C = 1$ in the numerical simulations. We define the norm as follows:

$$\|\phi\|_{L^1(B_n)} := \frac{\sum_{(i,j)/\boldsymbol{x}_{i,j} \in B_n} |\phi_{i,j}|}{\#\{(i,j)/\boldsymbol{x}_{i,j} \in B_n\}}. \tag{3.8}$$

Assuming uniformly distributed errors, this stopping criterion ensures that the error on ϕ in the band B_n is of order 3; thus, the curvature is computed with at least first-order accuracy. Using an explicit Euler method for the time discretization of (3.3), we infer that the stopping criterion (3.7) should be equivalent to requiring

$$\||\nabla\phi^n| - 1\|_{L^1(B_n)} \leqslant C(\Delta x)^2. \tag{3.9}$$

Again, C is a curvature-dependent constant (possibly different from the previous one), but we usually choose $C = 1$. Numerical simulations (not shown here) have been performed with both criteria (3.7) and (3.9) and give comparable results.

Remark 2
At this stage, the fast sweeping step is performed to guarantee a valid distance function in the whole domain. When coupled with the transport equation, the distance function is usually not required away from the interface. However, we keep it because it ensures that the discrete problems we solve (for any Δx or Δt) are approximations of the same continuous problem (see Section 4.3). Moreover, the computational cost of the fast sweeping step is low compared with the cost of the partial relaxation step. If one is only interested in having the distance function near the interface, it should be possible to use the strategy of Section 4.3 with a local criterion and the relaxation step only. We performed some tests that suggest the same kind of behaviour for both approaches.

Remark 3
In practice, it is sufficient to replace B_n by a neighbourhood of Γ containing all the points needed for the computation of geometric properties. In particular, the choice of (2.11)–(2.12)–(2.13) to compute geometric properties allows a rather small B_n, thus reducing the number of relaxation steps.

3.4. Numerical illustrations

We illustrate the behaviour of the mixed method for reinitialization on two cases: a closed interface and an interface crossing the boundary of Γ.

Example 1: Closed interface. The setting is given by (2.7)–(2.10), but with a different radius $R = 0.5$. This is a slight modification of the test case of in [19] where Γ is an ellipse. We choose a circle so as to have exact solutions for the distance function and the curvature.

The results of the mixed reinitialization algorithm are presented in Table II. We illustrate the second-order accuracy of the method, using the L^1 norm over the whole domain Ω. This second order is due to the second-order fast sweeping step. We compute the L^∞ error on ϕ in the narrow band B_n. We observe the expected third order. We show two series of results for the L^∞ error on κ on the interface. The first one uses the second-order finite differences formulas (2.11)–(2.13). The second ones uses a first-order formula for the approximation of κ. The latter exhibits a clear first-order convergence, whereas it is not that clear for the usual computation of κ. However, a linear regression on the results gives an average rate of convergence of 0.996. Even if the error seems to deteriorate for fine meshes, it is still better to use the second-order centred finite difference formula because the amplitude of the error is reduced by a factor of at least 10 for reasonable meshes.

Table II. Convergence errors for example 1 (circle).

	$\|\phi_h - d\|_{L^1(\Omega)}$		$\|\phi_h - d\|_{L^\infty(B_n)}$		$\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$		$\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$			
$\frac{1}{h}$	Error	coc	Error	coc	Error	coc	Error	coc	N_{it}	CPU time
20	3.37E−03	—	3.12E−02	—	1.81E−02	—	8.95E−02	—	24	0.048
40	4.25E−04	2.88	1.51E−03	4.22	5.15E−03	1.75	4.01E−02	1.12	26	0.160
80	1.10E−04	1.92	2.48E−04	2.56	1.28E−03	1.98	1.91E−02	1.05	28	0.604
160	3.19E−05	1.77	2.85E−05	3.09	4.65E−04	1.44	9.38E−03	1.01	31	2.560
320	9.43E−06	1.75	3.37E−06	3.06	1.60E−04	1.53	4.52E−03	1.05	34	11.09
640	2.57E−06	1.87	6.09E−07	2.46	1.97E−04	−0.30	2.34E−03	0.95	36	46.88
1280	6.82E−07	1.91	6.98E−08	3.12	1.51E−04	0.38	1.18E−03	0.98	38	206.1

Errors on the curvature are relative errors. N_{it} stands for the total number of iterations required by the reinitialization algorithm (relaxation + fast sweeping).

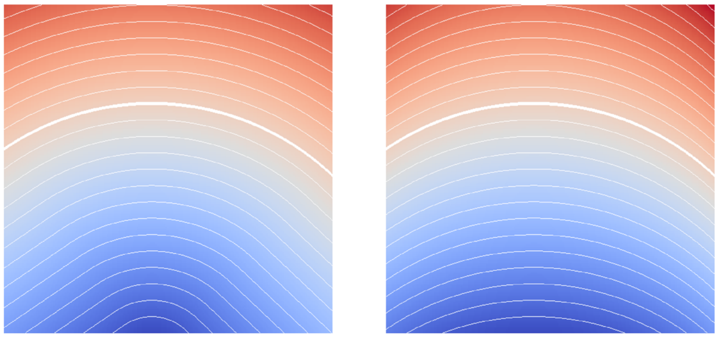

Figure 3. Left: distance function associated to the 'wave-like' interface (example 2). Right: initial level set. The original interface Γ_0 is the thick line. Other lines are isolines of ϕ.

Example 2: Wave-like interface. Now, we turn our attention to a case where Γ crosses the boundary of Ω. We use $\Omega = [-1; 1]^2$ and the interface

$$\Gamma := \left\{(x - x_0)^2 + (y - y_0)^2 = r_0^2\right\},$$

with $(x_0, y_0) = (-0.1; -1.2)$ and $r_0 = 1.7$; see Figure 3. Γ is the graph of a $\mathcal{C}^1$ function, and we start with

$$\phi_0(x, y) = y - y_0 - \sqrt{r_0^2 - (x - x_0)^2}.$$

The distance function we want to compute is depicted in Figure 3.

Note that when the interface touches the boundary of the domain, we need to extend Γ outside Ω. For this particular case, one could choose the natural extension by the circle $\mathcal{C}_0$ with centre (x_0, y_0) and radius r_0. In order to have a method that can be easily adapted for any kind of interfaces, we choose an extension with straight lines, tangent to Γ. Thus, ϕ_0 is the distance to $\mathcal{C}_0$ except in the bottom-left and bottom-right corners.

With this kind of extension, we may encounter inflows for the gradient of the level set. Thus, we have to handle part of the boundary using Dirichlet-type boundary conditions. The main difference with usual Dirichlet boundary conditions is that the boundary values we want to enforce are *a priori* unknown. Using linear extensions allows to accurately know the boundary values, so that ideas from [31] might be considered, that is, the use of Richardson extrapolation or a Lax–Wendroff procedure. However, the extension introduces a discontinuity line for $D^2\phi$ (cf. Figure 4), so that none of these ideas prevent the loss of the third-order accuracy in a neighbourhood of Γ. Thus, we choose a more naive approach to enforce Dirichlet conditions:

- Relaxation step: at the inflow boundary only, we add a Dirichlet step where we replace ϕ by the distance to the extension of Γ.

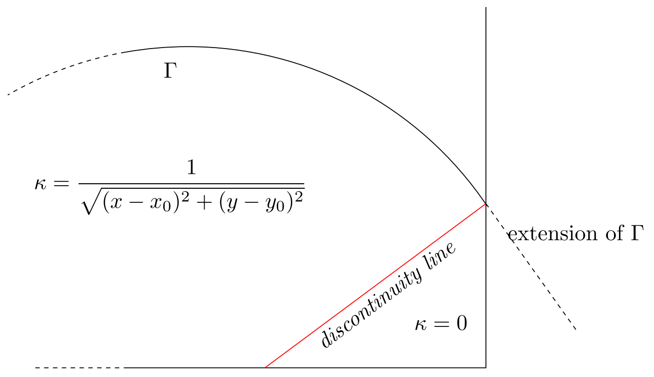

Figure 4. The extension of ϕ introduces the red line as a discontinuity line for $D^2 d$ (hence also for κ). Here, Γ is part of the circle $(x-x_0)^2+(y-y_0)^2 = r_0^2$ and crosses the boundary of Ω.

Table III. Convergence errors for example 2.

	$\lVert\phi_h - d\rVert_{L^1(\Omega)}$		$\lVert\phi_h - d\rVert_{L^\infty(B_n)}$		$\lVert\kappa_h - \kappa\rVert_{L^1(\Gamma)}$		$\lVert\kappa_h - \kappa\rVert_{L^\infty(\Gamma)}$			
$\frac{1}{h}$	Error	coc	Error	coc	Error	coc	Error	coc	N_{it}	CPU time
40	4.84E−04	—	4.40E−03	—	7.53E−03	—	8.36E−02	—	19	0.143
80	5.50E−05	3.08	3.40E−04	3.63	3.44E−03	1.11	8.44E−02	−0.01	23	0.599
160	9.01E−06	2.59	3.94E−05	3.08	2.38E−03	0.53	8.99E−02	−0.09	25	2.468
320	2.11E−06	2.09	4.84E−06	3.01	1.14E−03	1.05	1.06E−01	−0.24	28	10.51
640	5.32E−07	1.98	9.54E−07	2.34	6.21E−04	0.87	8.44E−02	0.33	30	43.79
1280	1.40E−07	1.93	2.55E−07	1.90	3.14E−04	0.98	5.96E−02	0.50	33	186.6

- Fast sweeping step: The conditions are enforced *in a band near* the inflow boundary, and not only on the boundary.

In our case, the bottom-left and bottom-right parts are Dirichlet boundaries, more precisely, denoting $\partial\Omega_D$ the inflow boundary, we have

$$\partial\Omega_D = \left\{(x,y)\ /\ (x = x_{\min} \text{ or } x = x_{\max}) \text{ and } y \leqslant y_0 + \sqrt{r_0^2 - (x-x_0)^2}\right\}.$$

We present the results in Table III. Again, the global second-order accuracy is reached. The linear extension of Γ introduces shocks near the intersection between Γ and the boundary of Ω; this leads to a loss of accuracy in B_n. As a result, the L^∞ error on ϕ is only second order accurate, and the computation of the curvature is also altered. We do not have the first-order accuracy in the L^∞ norm on Γ, but we still have it for the L^1 norm. This implies that κ is well computed away from the boundary points only. Note also that the number of iterations needed for convergence is different than for the previous example, but it is still increasing very slowly. As a comparison, for both examples, the resolution of the eikonal equation using only the relaxation method would have required more than 2000 iterations on the finest mesh (and then much more CPU time).

4. COUPLING TRANSPORT AND REINITIALIZATION

In this part, we investigate the coupling between the transport of the interface and the computation of the distance function. We present results where the interface is passively advected by a velocity field. We choose divergence-free velocity fields to emulate an incompressible flow. The challenge is to find a strategy that avoids too many reinitialization steps, because this procedure may introduce some errors in the position of the interface.

4.1. Advection

Recall that we have at hand a velocity field defined on Ω. In some applications, a different velocity is used for the advection step. This so-called extension velocity is supposed to keep ϕ as close as possible to the signed distance function and helps reduce the number of reinitialization steps. However, distortions of Γ might lead to a loss of smoothness, that is, the original velocity might be smoother than the extension velocity, and we infer that it could lead to higher errors during the advection part. We choose to work with the original velocity, and to solve

$$\partial_t \phi + \boldsymbol{U} \cdot \nabla \phi = 0. \tag{4.1}$$

Note that we use (4.1) even in the cases where $\boldsymbol{U}$ is divergence free. A WENO-5 scheme is used to compute spatial derivatives, and the time evolution is performed with an RK3-TVD scheme. For this advection, we use a time step that satisfies the CFL condition

$$\Delta t < \frac{\Delta x}{\max |\boldsymbol{U}|}.$$

4.2. Number of reinitialization steps

In the usual level set strategies, it is claimed that reinitialization is necessary but should not be performed too many times. This is again due to the slight displacement of the interface induced by the reinitialization algorithms. Even with a high-order reinitialization, the computation of the signed distance function at each time step might lead to significant errors in the position of the interface for long times. It is then necessary to limit the number of reinitialization steps. For a given final time, one might think of setting a maximum number of reinitialization steps and distribute them uniformly in time. However, it is unclear how to choose the maximum number of reinitialization steps. Our approach is based on the fact that, during the transport of Γ, ϕ slowly deviate from a distance function. We choose to perform the reinitialization when ϕ is 'too far away' from the signed distance function.

4.3. Strategy

Recalling that the distance function has the property $|\nabla d| = 1$, we introduce the quantity $r_g(\nabla\phi) := \| |\nabla\phi| - 1 \|_{L^1(\Omega)}$. We choose a threshold $\delta > 0$, and we consider the following continuous problem:

- *Initialization:* Start with $\phi_0 = d_0$, the signed distance function to Γ_0.
- *Transport:* While $r_g(\nabla\phi) < \delta$, evolve ϕ according to (4.1).
- *Reinitialization:* When $r_g(\nabla\phi) = \delta$, set $\phi_0 = \phi$ and recompute the signed distance function d. Set $\phi = d$ and go to *Transport*.

The method we want to use is based on a discrete approximation of this problem. The only modification is that the reinitialization steps occur when $r_g(\nabla\phi) \geqslant \delta$. Using the continuous problem as a reference, we see that the solution we want to approximate does not depend on the mesh size or the time step. This is a major difference compared with usual level set strategies where the reinitialization steps are performed every 5 or 10 time iterations and/or the reinitialization procedure is performed only in a narrow band, typically of length $5\Delta x$, near the interface. It is then reasonable to check if there is a convergence order. Numerical simulations tend to assess the expected high-order accuracy.

4.4. Numerical illustrations

We illustrate the performances of the new strategy on several examples. We start by showing that the coupling with the transport equation preserves the first-order accuracy on κ. This is not the case for usual level set strategies, and the results obtained with the new strategy compare well with standard approaches. We then illustrate the necessity of the reinitialization procedure when steep gradients are likely to be created. We also investigate the influence of the threshold δ in the scheme, and we

Table IV. Example 3 (vortex case).

	$\|\phi_h - d\|_{L^1(\Omega)}$		$\|\phi_h - d\|_{L^\infty(B_n)}$		$\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$ $(t = 4)$			$\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$ $(t = 2)$	
$\frac{1}{h}$	val	coc	val	coc	val	coc	N_{red}	val	coc
40	8.62E−04	—	2.44E−03	—	2.11E−01	—	17	1.47E−01	—
80	8.60E−05	3.27	3.04E−04	2.95	5.59E−02	1.88	17	4.63E−02	1.64
160	3.08E−05	1.47	2.23E−05	3.73	1.41E−02	1.97	16	1.51E−02	1.60
320	9.30E−06	1.72	1.97E−06	3.48	5.22E−03	1.43	16	4.60E−03	1.70
640	2.52E−06	1.88	1.15E−07	4.09	3.65E−04	3.83	17	3.16E−04	3.86
1280	6.59E−07	1.94	1.07E−08	3.43	9.73E−05	1.91	16	2.17E−04	0.54

Errors on ϕ and κ at $t = 4$ (first three columns). Total number of reinitialization steps (N_{red}). Error on κ at $t = 2$ (last column).

end the section by showing that the method depends very weakly on the time step, which makes it robust if coupled with problems where Δt is modified during the simulation.

4.4.1. Closed interface and vortex. We first present some results on a moving closed interface.

Example 3: Vortex case. This example is a slight modification of the original vortex example in [32]. The difference is that we impose a periodic velocity so as to recover the initial interface after a given time. The computational domain is $\Omega = [0,1]^2$. The initial interface Γ_0 is the circle with centre $(\frac{1}{2}, \frac{3}{4})$ and radius 0.15. The velocity field is given by the scalar potential:

$$\boldsymbol{U} = \cos\left(\frac{\pi t}{T}\right)\nabla^{\perp}\omega, \qquad \omega = \frac{1}{2\pi}\sin^2(\pi x)\sin^2(\pi y), \tag{4.2}$$

with $T = 2$. With this choice of $\boldsymbol{U}$, the interface is $\Gamma = \Gamma_0$ for $t = 0$, $t = 2$ and $t = 4$. The initial level set is the signed distance to Γ_0. We use meshes with typical length from 1/40 to 1/1280 and perform simulations for $t \in [0,4]$. We use the threshold $\delta = 0.1$ for reinitialization. The computation of the signed distance is enforced at $t = 4$ for each mesh. We report the results in Table IV at $t = 4$, as well as the error on κ at $t = 2$. We also report the total number of reinitialization steps during the whole computation.

These results confirm the good behaviour of the method, even when coupled with the transport of the interface. As expected, the number of reinitialization steps needed is almost uniform in h, because the strategy is based on the approximation of the same continuous problem.

The results on κ at $t = 2$ are presented because it is not a reinitialization time, but Γ is the original circle. As we enforce the computation of the signed distance function at $t = 4$, it is natural to wonder whether the good approximations properties that we observe is due to that last reinitialization or not. Results at $t = 2$ confirm that we have the first-order accuracy on κ, even for intermediate times.

Note that for this example, we obtain better results than the third-order accuracy expected for the L^∞ error on ϕ in the narrow band. This behaviour explains the higher order for the curvature.

4.4.2. Comparison with usual level set strategies. When accuracy is needed near the interface, usual level set strategies consists of using the relaxation method. The reinitialization procedure is performed at each time step (or every 5 or 10 time step), and a fixed number of iterations (3 or 5, e.g.) is used. We use the same setting as before and perform the computation until $t = 2$. We compare four cases:

1. five iterations of the relaxation method, every five time steps;
2. three iterations of the relaxation method at every time step;
3. our new method, with $\delta = 0.1$; and
4. our new method, with $\delta = 0.01$.

Results are presented in Tables V and VI. We compare the computation of the curvature at $t = 2$, the position of the interface and the volume loss. As $\nabla \cdot \boldsymbol{U} = 0$, the volume should be constant, that is, Table V. L^∞ error on the curvature at $t = 2$ for the four cases.

	Case 1		Case 2		Case 3		Case 4	
$\frac{1}{h}$	Error	coc	Error	coc	Error	coc	Error	coc
40	1.82E−01	—	1.18E−01	—	1.47E−01	—	1.22E−01	—
80	5.53E−02	1.69	6.35E−02	0.87	4.63E−02	1.64	4.16E−02	1.53
160	7.07E−02	−0.35	7.62E−02	−0.26	1.51E−02	1.61	1.30E−02	1.66
320	6.56E−02	0.11	9.56E−02	−0.33	4.58E−03	1.71	3.87E−03	1.74
640	1.10E−01	−0.75	2.01E−01	−1.07	3.27E−04	3.80	3.05E−04	3.66

Table VI. Error on the position of Γ and volume loss at $t = 2$.

	Case 1		Case 2		Case 3		Case 4	
$\frac{1}{h}$	pos.	vol. loss	pos.	vol. loss	pos.	vol. loss	pos.	vol. loss
40	2.19E−03	6.82E−03	7.89E−04	1.06E−03	2.20E−03	5.67E−03	1.03E−03	2.50E−03
80	2.99E−04	9.39E−04	1.58E−04	2.72E−04	2.90E−04	4.06E−04	1.77E−04	1.53E−04
160	6.45E−05	1.86E−04	9.20E−05	2.19E−04	2.07E−05	1.88E−05	1.80E−05	1.30E−05
320	3.19E−05	7.38E−05	4.27E−05	1.15E−04	1.59E−06	9.66E−07	1.44E−06	9.25E−07
640	1.63E−05	3.66E−05	2.25E−05	6.21E−05	7.83E−08	4.30E−08	8.02E−08	3.50E−08

$$\frac{d}{dt}\int_{\{\phi(\boldsymbol{x})\leq 0\}} 1\, d\boldsymbol{x} = 0.$$

The total number of reinitialization steps for the finest mesh is respectively 256, 1240, 10 and 86 for each case. The results show that usual reinitialization strategies do not ensure an accurate computation of κ, while the new method is first order accurate on the curvature. We also notice that both the position of Γ and the volume are computed more accurately with the new method. For cases (1) and (2), they are approximated with first-order accuracy, whereas the new method exhibits an average rate of convergence of almost 4.

The maximum deformation of Γ is for $t = 1$. In this example, the distortion is not too severe, so that cases (3) and (4) (i.e. the new method with $\Delta = 0.1$ and $\Delta = 0.01$) give similar results. Table V indicates that a rather small number of reinitialization steps is needed to obtain accurate results. In this particular case, it is even better to use the transport equation without any reinitialization (cf. also Section 4.4.4). However, as shown in the next section, reinitialization steps are sometimes necessary to obtain accurate approximations.

4.4.3. Formation of steep gradients near the interface. On the previous example, because the distortion of Γ is not too severe, there is no creation of (very) steep gradients near the interface. Reinitialization might be critical to accurately compute the curvature when the flow tends to create steep gradients, for instance, in the case of the merging of two bubbles [7]. We illustrate this fact and the behaviour of our method on two cases.

Example 4: We use the following setting:

$$\Omega = [-1;1]^2, \tag{4.3}$$

$$\Gamma_0 = \{y = 0\}, \tag{4.4}$$

$$\boldsymbol{U} = \nabla^\perp \omega, \tag{4.5}$$

$$\omega(x, y) = \frac{1}{2\pi}\sin(\pi x)^2 \sin(\pi y)^2. \tag{4.6}$$

With this choice, the interface does not move, that is, $\Gamma = \Gamma_0$. The final time is $t = 6$. Without reinitialization, the isocontours of ϕ are greatly distorted, and steep gradients appear, even in the neighbourhood of Γ (cf. Figure 5).

Example 5: Flow around a circular cylinder. In the case of a flow around a motionless cylinder, the usual form for the components of the velocity is given in polar coordinates by

$$V_r = \left(U_\infty - \frac{K}{r^2}\right)\cos(\theta), \tag{4.7}$$

$$V_\theta = -\left(U_\infty + \frac{K}{r^2}\right)\sin(\theta). \tag{4.8}$$

We choose $U_\infty = 1$, $K = 1$, Γ_0 is the circle with radius 1, centred at the origin, and the numerical domain is $\Omega = [-3;3]^2$. We slightly change the velocity field inside the cylinder to avoid the singularity at the origin, by setting

$$U_r = \alpha c(r) V_r, \qquad U_\theta = \alpha c(r) V_\theta,$$

where $c(r) = \min(1, \frac{r}{0.5})^3$ and α is chosen so that $\|\boldsymbol{U}\|_{L^\infty(\Omega)} = 1$. With this setting, the interface does not move (even if $\boldsymbol{U} \neq 0$ on Γ). Again, without reinitialization, the isocontours of ϕ are severely distorted, leading to numerical issues for the transport itself (cf. Figure 5). As for

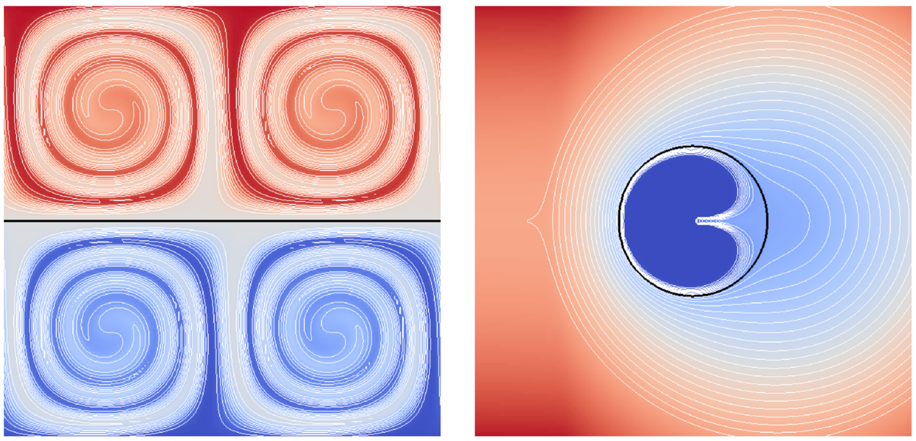
Figure 5. Isocontours of ϕ for example 4 (left) and example 5 (right) at $t = 6$ without reinitialization. Γ is the dark thick line.

Table VII. Example 4: $\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$ at $t = 6$ and computed order of convergence.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$	
$\frac{1}{h}$	Error	coc	Error	coc	Error	coc
40	1.03E−02	—	1.65E−02	—	7.20E+00	—
80	2.68E−03	1.90	1.10E−03	3.84	4.11E−01	4.06
160	4.13E−04	2.68	1.54E−04	2.81	1.55E−02	4.69
320	1.50E−04	1.46	6.70E−05	1.20	8.37E−04	4.19
640	3.59E−05	2.05	5.30E−05	0.34	5.04E−05	4.04

The number of reinitialization steps for the last mesh is 175, 40 and 1, respectively.

Table VIII. Example 5: $\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$ at $t = 6$ and computed order of convergence.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$	
$\frac{1}{h}$	err.	coc	err.	coc	err.	coc
40	7.24E−02	—	8.35E−02	—	1.40E+00	—
80	3.91E−02	0.87	1.68E−02	2.27	2.15E+00	−0.61
160	9.71E−03	1.99	5.09E−03	1.71	3.53E+00	−0.71
320	3.66E−03	1.40	2.51E−03	1.01	8.22E+01	−4.52
640	2.51E−03	0.54	1.88E−03	0.42	1.57E+02	−0.93

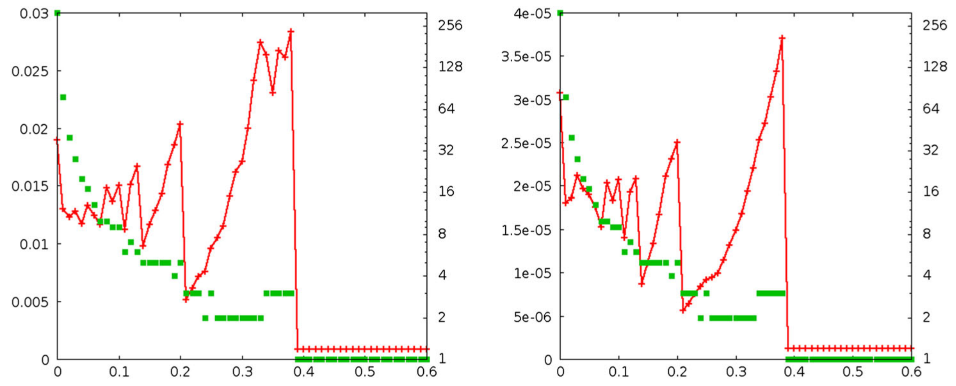

Figure 6. Example 3 (vortex case): error on the curvature (left) and the position (right) of Γ, depending on the value of δ. The squares represent the number of reinitialization steps required (logarithmic scale).

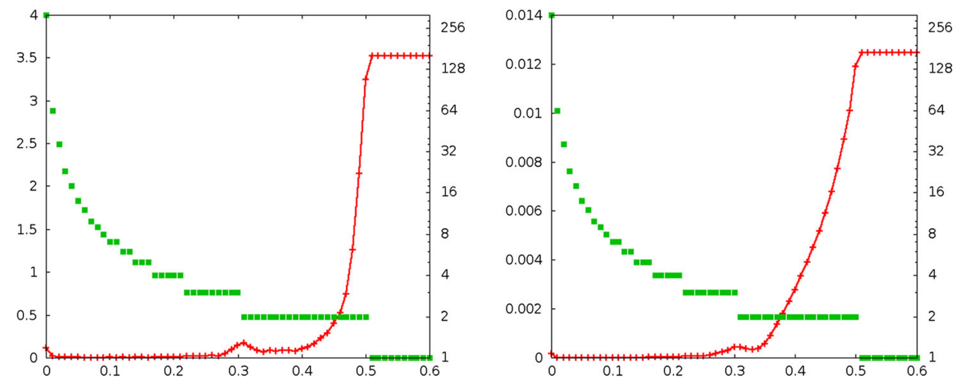

Figure 7. Example 5 (flow around cylinder): error on the curvature (left) and the position (right) of Γ, depending on the value of δ. See Figure 8 for a zoom on the region $\delta \in [0; 0.3]$. The squares represent the number of reinitialization steps required (logarithmic scale).

example 4, the final time is $t = 6$.

For both examples 4 and 5, we compute the curvature at the final time, with three different values of δ. We enforce a reinitialization step at the end. For $\delta = 0.01$, the distortion of the isocontours of ϕ between two reinitialization steps is mild, whereas for $\delta = 1$, there is no reinitialization until the final time. The L^∞ error on the curvature (on Γ) is reported in Tables VII and VIII.

In both cases, results are improved thanks to the reinitialization procedure. In example 4, a fine mesh (640×640) is necessary to obtain comparable results. In the case of the flow around a cylinder (example 5), reinitialization is even *necessary* to obtain a consistent computation of κ. The computation of other quantities (not shown here), such as the position of the interface or the volume loss, shows the same behaviour. This example confirms that the reinitialization steps *are indeed necessary* in some cases to obtain stable and accurate enough numerical simulations.

4.4.4. Impact of the threshold δ. We now compare the results we obtain with several values of the threshold δ that controls the reinitialization steps. The limit $\delta \to 0$ corresponds to perform the reinitialization procedure at every time step, for which we expect a loss of accuracy on κ. The limit $\delta \to +\infty$ corresponds to the case of transport alone. We test the influence of δ on two examples: the case given in Section 4.4.2 and the flow around a cylinder (4.7)–(4.8). The computations are performed on a 160×160 mesh. For each case, we compute the curvature and the position of Γ at the final time, respectively $t = 2$ and $t = 6$.

Results are presented in Figures 6 and 7, respectively. For each case, there seems to be an optimal value of δ that gives the lowest amplitude on both the curvature and the position of Γ. The error decreases rapidly when δ increases (Figure 8): it confirms that it is best not to choose a too small threshold. This behaviour is more explicit for example 5: the error on the curvature for $\delta = 0$ is almost 0.12 while the minimum error for $\delta \sim 0.08$ is almost $2.2\ 10^{-3}$. For values of δ that are not too small, there is a range for which the number of reinitializations (and hence the results) is comparable. Finally, in the range of bigger δ, that is, very small number of reinitialization steps, the behaviour is different in the two examples. In the first case, the slight distortion of ϕ and the reversibility of the test case allow better results without reinitialization. On the contrary, in the second case, the distortion is more severe and increases in time, so that transport alone leads to huge errors compared with computations with reinitializations. In general cases, the behaviour of the isocontours of ϕ is not known *a priori*; the use of a rather small threshold δ seems to be a good compromise to obtain accurate results.

4.4.5. Sensitivity to the time step. We have shown that, for rather small values of δ, the numerical method we have presented gives good results compared with usual level set strategies. Recall that in these level set strategies, a reinitialization step is performed every N time steps, where the typical values of N are 1, 5 or 10; thus, decreasing Δt increases the number of reinitialization steps. It may lead to significant differences in the accuracy. We investigate the behaviour of our new strategy with

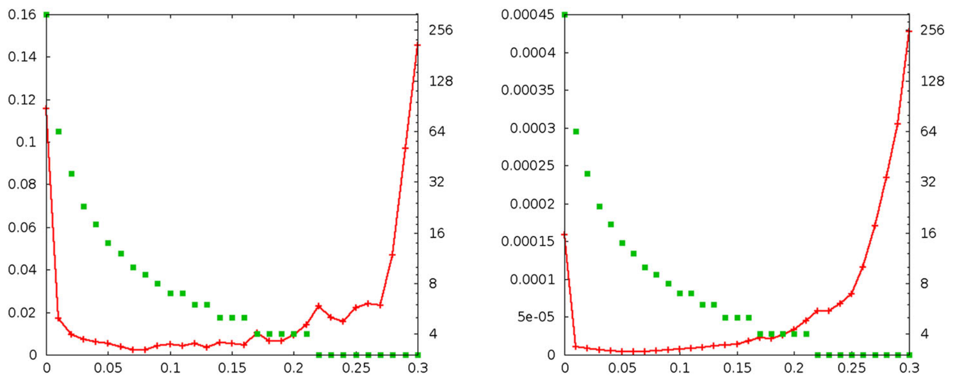

Figure 8. Example 5 (flow around cylinder): error on the curvature (left) and the position (right) of Γ, depending on the value of δ. The squares represent the number of reinitialization steps required (logarithmic scale).

Table IX. Example 3 with $T = 4$.

$\frac{\Delta t}{\Delta x}$	$\delta = 0$ Error	$\delta = 0$ N_{red}	$\delta = 0.01$ Error	$\delta = 0.01$ N_{red}	$\delta = 0.1$ Error	$\delta = 0.1$ N_{red}	$\delta = 1$ Error	$\delta = 1$ N_{red}
1	3.69E−01	640	3.35E−01	180	2.96E−01	17	1.76E−02	1
1/2	1.4337	1280	3.31E−01	208	2.86E−01	17	1.76E−02	1
1/4	27.158	2560	3.28E−01	220	2.82E−01	17	1.76E−02	1
1/8	31.058	5120	3.69E−01	224	2.92E−01	17	1.76E−02	1

Error $\|\kappa_h - \kappa\|_{L^\infty(\Gamma)}$ and total number of reinitialization steps at $t = 4$ for several values of δ and Δt.

respect to the time step. The setting is that of example 3 with $T = 4$. Simulations are performed on a 160×160 mesh, for several values of Δt. The L^∞ error on κ at $t = 4$ is represented in Table IX, with different values of δ. Other quantities such as the position of the interface or the volume have been computed (results not shown) and exhibit a similar behaviour. A reinitialization step is enforced at $t = 4$.

As expected, the case $\delta = 0$ shows the same behaviour as usual level set strategies, that is, the results become worse and worse as Δt decreases. On the contrary, for $\delta > 0$, the results weakly depend on Δt. Thus, the method is insensitive to Δt; in many simulations, it might be useful to change Δt according to $\boldsymbol{U}$ to improve computational efficiency. The use of $\delta > 0$ ensures that there will be no loss of accuracy.

4.4.6. Comparison with moment of fluid methods. In this section, we give some details about the behaviour of our strategy, compared with some other high-order methods. We quantitatively compare our method with the coupled level set/moment of fluid (CLSMOF) method in [33] and give some remarks about the comparison with gradient augmented level set methods (GALSM) in [34]. In order to compare results with the CLSMOF methods, we focus on the symmetric difference error, that is, the sum over all the cells of the difference between the real area enclosed in the cell and the recomputed area. An illustration is given in Figure 9. Note that we use piecewise linear reconstruction of the interface. As we have an accurate level set function, it should be possible to obtain better estimations by using information on the derivatives of ϕ and computing quadratic reconstruction for example. Denoting Ω_E (resp. Ω_C) the exact (resp. computed) domain enclosed by the interface, we compute the local and global symmetric difference errors with

$$E_{i,j} := \Big| |\Omega_E \cap C_{i,j}| - |\Omega_C \cap C_{i,j}| \Big| \qquad\qquad E_{sym} := \sum_{i,j} E_{i,j}, \tag{4.9}$$

where $C_{i,j}$ denotes the cell with bottom-left corner $\boldsymbol{x}_{i,j}$. We compare the methods on the following example.

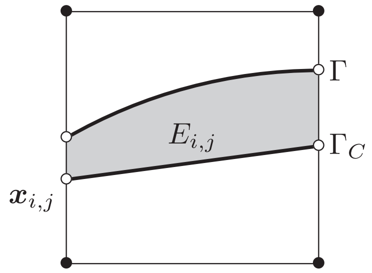

Figure 9. Illustration of the symmetric difference error (grey area). Γ_C is the reconstructed interface.

Table X. Example 6 with $T = 0.5$ (left) and $T = 8$ (last column). Three different advection scheme are used: a non conservative scheme (NC), a conservative scheme (C) and GALSM. The symmetric difference error is not normalized by the actual area of the circle, which is 7.069 10^{-2}.

	$T = 0.5$		$T = 8$ (C)		$T = 8$ (NC)		$T = 8$ (GALSM)	
$\frac{1}{\Delta x}$	E_{sym}	coc	E_{sym}	coc	E_{sym}	coc	E_{sym}	coc
32	4.53E−04	-	1.67E−01	-	7.07E−02	-	2.15E−01	-
64	1.30E−04	1.76	6.82E−02	1.29	7.07E−02	0.00	5.97E−02	1.81
128	3.26E−05	1.98	2.33E−02	1.55	3.26E−02	1.12	1.08E−02	2.43
256	7.65E−06	2.08	4.78E−03	2.29	6.57E−03	2.31	2.74E−03	1.98
512	1.90E−06	2.00	8.94E−04	2.42	1.12E−03	2.55	1.18E−03	1.21

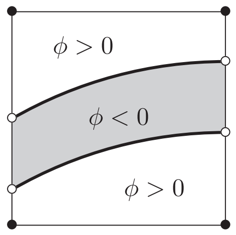

Figure 10. Under-resolved grid. The interface crosses twice the left and right boundaries of the cell. The four blank circles cannot be detected and are not used in the reinitialization procedure. Thus, information is lost and cannot be recovered afterwards.

Example 6: Weakly and strongly distorted vortex case. We use the same geometric setting as in example 3, but we use

$$\boldsymbol{U} := 2\cos\left(\frac{\pi t}{T}\right)\omega.$$

We perform numerical simulations with $T = 0.5$ and $T = 8$, in order to compare results with that in [33] and compute E_{sym} at $t = T$. The case $T = 0.5$ is quite smooth, and numerical results are expected to be good. On the contrary, the case $T = 8$ is challenging because the interface is distorted and filaments are created. The numerical results are reported in Table X.

As expected, the results are very good for $T = 0.5$ even though we have used a non-conservative scheme for the advection of ϕ. They quantitatively compare with the result of [33, Table 3]. The results for $T = 8$ are worse. We present results with a non-conservative scheme, with a conservative scheme and for a coupling with GALSM [34] for the advection part. The use of a non-conservative scheme for the advection equation leads to a total loss of volume on coarse grids. In this respect, the method does not perform as well as CLSMOF on coarse grids. However, once the grid is fine enough, we have an expected order of convergence of 2 for the symmetric difference error. The results with GALSM on the finest mesh are not as good as expected, which might be explained by the fact that we did not use any gradient limiter (e.g. [35]). However, for the cases (C) and (NC), the method behaves correctly when the grid is fine enough to detect the thinnest structures. As a comparison, the reported error on a 256×256 mesh is here around $4 \cdot 10^{-3}$, while the CLSMOF (using adaptive mesh refinement) is able to give comparable results with a 64×64 mesh (effective mesh resolution [33]). There are two features that may explain the behaviour of the method on coarse grids:

- When the inner domain enclosed by Γ is too thin (e.g. Figure 10), the reinitialization procedure loses information that cannot be recovered afterwards.
- The method separates the advection from the reinitialization, so that a non-conservative advection scheme is likely to lose volume more quickly and thus amplifies the previous point.

In the case of example 6 with $T = 8$, both phenomena occur. Even though the conservative scheme or the GALSM might help improve a little the results on a coarse grid, the first phenomenon is likely to give the major contribution to the error. So far, the intersection between Γ and the mesh is computed only when the value of ϕ changes sign between two adjacent nodes. Thus, configurations depicted in Figure 10 are not detected. It should be possible to improve the accuracy of the method on coarse grids with better tools to capture the intersection between Γ and the mesh, but this issue is beyond the scope of this paper.

5. DISCUSSION

5.1. Numerical simulations

We have described a level set strategy that allows accurate computations of geometric properties of an evolving interface. The strategy is based on a limited number of reinitialization and may be coupled with any third-order accurate method to solve the transport equation. The computational cost is limited thanks to the reinitialization algorithm and the limited number of reinitialization steps needed:

- The cost of one reinitialization step is quite low, but with a good accuracy for the computation of geometric properties. For example, we have shown examples where the total number of iterations during the reinitialization procedure does not exceed 40, even on a 1280×1280 mesh, while the relaxation method alone would have required nearly 2000 iterations.
- The number of reinitialization steps is controlled by the distortion of the level set itself, so that reinitialization only occurs when flat or steep gradients are created. In other cases, the amplitude of the error on κ remains reasonable.

Compared with usual level set strategies that involves either an incomplete reinitialization or a number of reinitialization steps that depend on the mesh size or the time step, this method exhibits a third-order accuracy on ϕ and then a first-order accuracy on κ. This approximation result allows the coupling between this level set strategy and a numerical scheme involving $\boldsymbol{n}$ or κ. On a computational point of view, it also allows to anticipate the behaviour of the numerical codes when meshes are refined. These results confirm that it is interesting to have a method for which the total number of reinitialization steps does not depend on Δx or Δt. We performed other simulations with the new reinitialization procedure, but we uniformly distributed (in time) reinitialization steps. For the same number of iterations, the results are comparable, yet a little worse on κ than the strategy presented in Section 4.3.

Although the method is third order accurate for (smooth enough) closed interfaces, the performance is altered when the interface crosses the boundary of the domain. The main issue is that we have artificially introduced a discontinuity by extending ϕ with straight lines. The discontinuity line is crossing the interface at the boundary, leading to a loss of accuracy in the computation of geometrical properties (cf. Figure 4). It should be possible to recover the first-order accuracy on κ in several ways:

- use a different extension for Γ, to ensure that the regions where the second derivative of d is discontinuous does not touch Γ; and
- use a different stencil to compute κ near the boundary, to ensure that only points in the 'smooth regions' are involved.

Note that it is sufficient to replace B_n (used in the relaxation step) by a neighbourhood of Γ containing all the points needed for the computation of geometric properties.

Finally, the method involves a criterion to perform the reinitialization, namely, $r_g(\nabla\phi)$, and a threshold δ. We have defined r_g on the whole domain Ω, but it is possible to use only a neighbourhood of Γ, provided it is uniform in h. The threshold can be taken as small as desired; decreasing δ increases the number of reinitialization steps needed. In general, there is no way to know if strong deformations of Γ will occur, or if they could be reversible. In such cases, we have shown that it is indeed preferable to use reinitialization, with a threshold that is not too small, in order to obtain better results. Unfortunately, it remains not clear how to make that choice. In practice, we often use a threshold $\delta \in [0.01; 0.1]$, for which there seems to be only a few differences in the results.

5.2. *Extensions of the method*

Although the work presented here is only for 2D it is clear that the same strategy could be carried out in a 3D framework without modifications. We have also chosen methods that should be easily parallelized. In the Cartesian framework, the parallelization might be performed using domain decomposition. This should be tractable for the relaxation step, with a small loss of scaling due to the large WENO stencil. If domain decomposition is not efficient enough for the fast sweeping step, the strategy developed by Detrixhe *et al.* [27] may be used; it is based on the fact that, when sweeping in one direction, nodes in the orthogonal diagonal are independent of one another.

The method has been illustrated on passively advected interfaces with a given velocity field, but it might be used for other applications, provided the transport step is accurate enough. It is suitable in the case of fluid mechanics where the velocity field is available on the whole domain. In the cases where $\boldsymbol{U}$ is only given on Γ (e.g. for surfaces moving with curvature-dependent speed, see [4]), it still might be used with a suitable extension velocity construction [9, 36].

Finally, the method has been developed in the context of smooth interfaces, so that the high-order accuracy of the two separate steps (advection and reinitialization) ensure a global high-order accuracy. In the case of non-smooth interfaces, for example, interfaces with corners, non-diffusive advection schemes have to be considered. For the reinitialization procedure, the algorithm shown here is only first order accurate in rarefaction regions, as many other methods (e.g. [26]). However, it is reasonable to guess that a special treatment of the corners coupled with this strategy could improve the level set representation and is currently in progress.

REFERENCES

1. Womble DE. A front-tracking method for multiphase free boundary problems. *SIAM Journal of Numerical Analysis* 1989; **26**(2):380–396.
2. Magnaudet J, Rivero M, Fabre J. Accelerated flows past a rigid sphere or a spherical bubble. I. Steady straining flow. *Journal of Fluid Mechanics* 1995; **284**:97–135.
3. Hirt CW, Amsden AA, Cook JL. An arbitrary Lagrangian–Eulerian computing method for all flow speeds. *Journal of Computational Physics* 1974; **14**(3):227 –253.
4. Osher S, Sethian JA. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics* 1988; **79**(1):12–49.
5. Bergmann M, Hovnanian J, Iollo A. An accurate Cartesian method for incompressible flows with moving boundaries. *Commun. Comput. Phys.* 2014; **15**(5):1266–1290.
6. Gorsse Y, Iollo A, Telib H, Weynans L. A simple second order Cartesian scheme for compressible Euler flows. *Journal of Computational Physics* 2012; **231**(23):7780–7794.
7. Sussman M, Smereka P, Osher S. A level-set approach for computing solutions to incompressible two-phase flows. *Journal of Computational Physics* 1994; **114**:146–159.
8. Brackbill JU, Kothe DB, Zemach C. A continuum method for modeling surface tension. *Journal of Computational Physics* 1992; **100**(2):335–354.
9. Zhao HK, Chan T, Merriman B, Osher S. A variational level set approach to multiphase motion. *Journal of Computational Physics* 1996; **127**(1):179–195.
10. Adalsteinsson D, Sethian JA. The fast construction of extension velocities in level set methods. *Journal of Computational Physics* 1999; **148**(1):2–22.
11. Coupez T. Convection of local level set function for moving surfaces and interfaces in forming flow, Materials Processing and Design, Modeling, Simulation and applications, NUMIFORM '07: 9th International Conference on Numerical Methods In Industrial Forming Processes, Porto, Portugal, Jun. 2007; 61–66.
12. Ville L, Silva L, Coupez T. Convected level set method for the numerical simulation of fluid buckling. *International Journal for Numerical Methods in Fluids* 2011; **66**(3):324–344.
13. Olsson E, Kreiss G. A conservative level set method for two phase flow. *Journal of Computational Physics* 2005; **210**(1):225–246.
14. Olsson E, Kreiss G, Zahedi S. A conservative level set method for two phase flow. II. *Journal of Computational Physics* 2007; **225**(1):785–807.
15. McCaslin JO, Desjardins O. A localized re-initialization equation for the conservative level set method. *Journal of Computational Physics* 2014; **262**:408–426.
16. Rouy E, Tourin A. A viscosity solutions approach to shape-from-shading. *SIAM Journal of Numerical Analysis* 1992; **29**(3):867–884.
17. Sethian A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 93, 1996; 1591–1595.
18. Tsai YHR, Cheng LT, Osher S, Zhao HK. Fast sweeping algorithms for a class of Hamilton–Jacobi equations. *SIAM Journal of Numerical Analysis* 2003; **41**(2):673–694.
19. Russo G, Smereka P. A remark on computing distance functions. *Journal of Computational Physics* 2000; **163**(1): 51–67.
20. Shi XD, Brenner MP, Nagel SR. A cascade of structure in a drop falling from a faucet. *Science* 1994; **265**(5169): 219–222.
21. Gottlieb S, Shu CW. Total variation diminishing Runge–Kutta schemes. *Mathematical Computations* 1998; **67**(221):73–85.
22. Jiang GS, Peng D. Weighted ENO schemes for Hamilton–Jacobi equations. *SIAM Journal of Science Computations* 2000; **21**(6):2126–2143 (electronic).
23. du Chéné A, Min C, Gibou F. Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes. *Journal of Science Computations* 2008; **35**(2-3):114–131.
24. Liu XD, Osher S, Chan T. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics* 1994; **115**(1):200–212.
25. Jiang GS, Shu CW. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics* 1996; **126**(1):202–228.

26. Zhang YT, Zhao HK, Qian J. High order fast sweeping methods for static Hamilton–Jacobi equations. *Journal of Science Computations* 2006; **29**(1):25–56.
27. Detrixhe M, Gibou F, Min C. A parallel fast sweeping method for the eikonal equation. *Journal of Computational Physics* 2013; **237**:46–55.
28. Zhao H. A fast sweeping method for eikonal equations. *Mathematical Computations* 2005; **74**(250):603–627.
29. Sethian JA. Fast marching methods. *SIAM Rev.* 1999; **41**(2):199–235.
30. Chopp DL. Some improvements of the fast marching method. *Journal of Scientific Computing* 2001; **23**(1):230–244 (electronic).
31. Huang L, Shu CW, Zhang M. Numerical boundary conditions for the fast sweeping high order WENO methods for solving the eikonal equation. *Journal of Computational Mathematics* 2008; **26**(3):336–346.
32. Bell JB, Colella P, Glaz HM. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1989; **85**(2):257–283.
33. Jemison M, Loch E, Sussman M, Shashkov M, Arienti M, Ohta M, Wang Ys. A coupled level set-moment of fluid method for incompressible two-phase flows. *Journal of Scientific Computing* 2013; **54**(2–3):454–491.
34. Nave JC, Rosales RR, Seibold B. A gradient-augmented level set method with an optimally local, coherent advection scheme. *Journal of Computational Physics* 2010; **229**(10):3802–3827.
35. Bøckmann A, Vartdal M. A gradient augmented level set method for unstructured grids. *Journal of Computational Physics* 2014; **258**:47–72.
36. Peng D, Merriman B, Osher S, Zhao H, Kang Ms. A PDE-based fast local level set method. *Journal of Computational Physics* 1999; **155**(2):410–438.