A sharp cartesian method for the simulation of air-water interface

Lisl Weynans, Michel Bergmann, Francky Luddens

The Eighth International Conference on Computational Fluid Dynamics (ICCFD8), Chengdu, Sichuan,China, July 14-18, 2014

ICCFD8-2014-xxx

A sharp cartesian method for the simulation of air-water interface

Lisl Weynans[1], Michel Bergmann[1], Francky Luddens[1],

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

Corresponding author: Lisl.Weynans@math.u-bordeaux1.fr

Abstract: We firstly present a sharp cartesian method for the simulation of incompressible flows with high density and viscosity ratios, like air-water interfaces. This method is inspired from the second-order cartesian method for elliptic problems with immersed interfaces developed in [1]. Then, because a high-order interface description is necessary in this context, we present a level-set technique allowing to maintain in the course of time a third-order accuracy of the level-set itself, and thus a first order accuracy of the curvature.

Keywords: incompressible flows, air-water interface, level-set method, cartesian grid discretization, redistanciation.

1 Description of the method for air-water interface

The air-water interface evolution is described by the incompressible Navier-Stokes in each subdomain (air or water):

$$\rho(\boldsymbol{u}_t + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}) = -\nabla p + (\nabla.\tau)^T + \rho\boldsymbol{g},$$
$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0,$$

with jump conditions at the interface Γ between the two fluids, derived from:

- the velocity continuity and its material derivative : $[u] = [v] = 0$, and $\left[\dfrac{\nabla p}{\rho}\right] = \left[\dfrac{(\nabla.\tau)^T}{\rho}\right]$,
- the balance between normal stresses and surface tension: $[p] = \sigma\kappa + 2[\mu](u_n, v_n).\boldsymbol{n}$,

with κ the local curvature of the interface, σ the coefficient of the surface tension and μ the viscosity.

The interface is represented with a level-set function, and the normal to the interface and curvature are computed with the following formulas at interface points:

$$\boldsymbol{n} = \frac{\nabla\phi}{|\nabla\phi|}, \text{ and } \kappa = \nabla\cdot\boldsymbol{n}.$$

A classical predictor-corrector algorithm is used to solve the fluid equations, in a non-incremental version, which means that the guess value for the pressure is zero. This choice avoids instability issues due to the

discontinuous pressure values when the interface moves. We thus compute successively:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = \frac{[(\nabla.\tau)^T]^n}{\rho} + \boldsymbol{g}, \qquad \text{(prediction step)}$$
$$\text{and} \quad \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\frac{\nabla p}{\rho}. \qquad \text{(correction step)}$$

To compute the pressure, according to the above jump conditions, it is necessary to solve an elliptic problem with discontinuous values of the solution and its derivative across the interface:

$$\begin{aligned}
&\nabla \cdot (\frac{1}{\rho}\nabla p) = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t}, \\
&[p] = \sigma\,\kappa + 2[\mu]\,(u_n, v_n).\boldsymbol{n} \text{ on } \Gamma, \\
&\left[\frac{\nabla p}{\rho}.\boldsymbol{n}\right] = \left[\frac{(\nabla.\tau)^T}{\rho}.\boldsymbol{n}\right] \text{ on } \Gamma.
\end{aligned}$$

In practice, we take into account the viscous forces by following a continuous approach: we regularize the density and viscosity values by an harmonic averaging for the computation of the viscous term in the prediction step. This approach, as explained in [2], allows for a more straightforward and robust treatment, and has been proven to provide satisfactory accuracies for high Reynolds numbers. The elliptic problem to solve becomes therefore

$$\begin{aligned}
&\nabla \cdot (\frac{1}{\rho}\nabla p) = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t}, \\
&[p] = \sigma\,\kappa \text{ on } \Gamma, \\
&\left[\frac{\nabla p}{\rho}.\boldsymbol{n}\right] = 0 \text{ on } \Gamma,
\end{aligned}$$

avoiding the computation of the terms $(u_n, v_n).\boldsymbol{n}$ and $[\frac{(\nabla.\tau)^T}{\rho}.\boldsymbol{n}]$.

This elliptic problem with discontinuous values across an interface is solved with the second order method developed in [1]. The originality of this method lies in the use of unknowns located at the interface. These interface unknowns are used to discretize the flux jump conditions and the elliptic operator accurately enough to get a second order convergence in maximum norm. Actually, near the interface, the elliptic operator needs to be discretized with a first order truncation error, while the fluxes have to be discretized with a second-order truncation error. For a visual explanation of the discretization see figure 1. The advantage of using this method, compared to the reference work of [3] is that the jump conditions for the corrective step are solved more accurately.

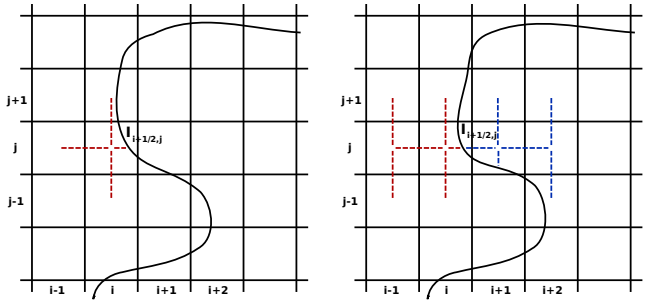

Figure 1: Examples of stencils used for the discretization of elliptic operator (left) and fluxes on each side of the interface (right). $I_{i+1/2,j}$ is an interface point where an interface value is created.

The term $\nabla \cdot \boldsymbol{u}^*$ is discretized near the interface with a decentered stencil, because the value of $\boldsymbol{u}^*$ at the interface is not known and has not evident physical meaning. The corrective term for the velocity, that is, the gradient of p, is then also computed with an adapted decentered stencil near the interface, in order to keep a consistent discretization.

2 Numerical validation of the sharp cartesian method for air-water interface

2.1 Rising of air bubble in water

We computed the evolution of air bubbles rising in water. We considered two cases: a small bubble, and a large one. In the first case, the surface tension plays an important role in the evolution of the interface because of the small size of the water subdomain. In the second case, the surface tension has less influence, and larger deformations occur.

We use the following physical values: $g = -9.81m/s^2$, $\sigma = 0.0728kg/s^2$, $\rho_{air} = 1000$ kg$/m^3$, $\mu_{air} = 1,137.10^{-3}$ kg/ms, $\rho_{water} = 1$kg$/m^3$, $\mu_{water} = 1,78.10^{-5}$ kg/ms. The computations are made with 80×120 points.

Small air bubble: The computational domain is $[-0.01m, 0.01m] \times [-0.01m, 0.02m]$. It is filled with water excepted a circular air bubble of radius $\frac{1}{300}m$, centered at the origin and initially at rest. We present the interface evolution and the velocity at times $t = 0, 0.02, 0.035$ and 0.05 on figures 2 and 3.

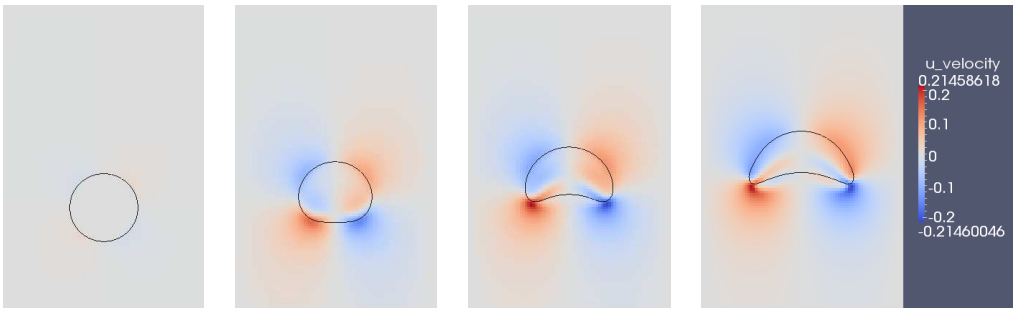

Figure 2: Evolution of the interface and horizontal velocity for a small air bubble rising in air, from left to right at times $t = 0, 0.02, 0.035, 0.05$.

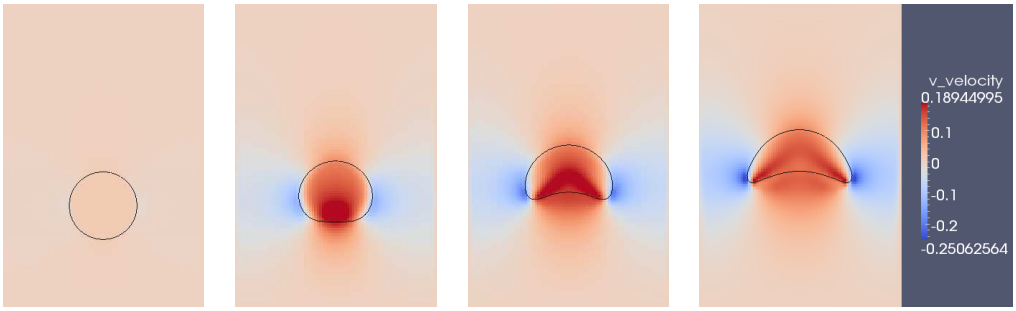

Figure 3: Evolution of the interface and vertical velocity for a small air bubble rising in air, from left to right at times $t = 0, 0.02, 0.035, 0.05$.

Large air bubble: The computational domain is $[-1m, 1m] \times [-1m, 2m]$. It is filled with water excepted a circular air bubble of radius $\frac{1}{3}m$, centered at the origin and initially at rest. We present the interface evolution and the velocity at times $t = 0, 0.2, 0.35$ and 0.5 on figures 4 and 5.

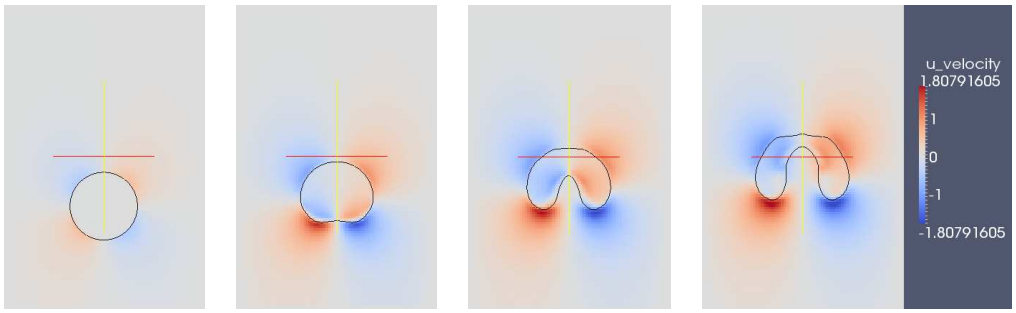

Figure 4: Evolution of the interface and horizontal velocity for a large air bubble rising in air, from left to right at times $t = 0, 0.2, 0.35, 0.5$.

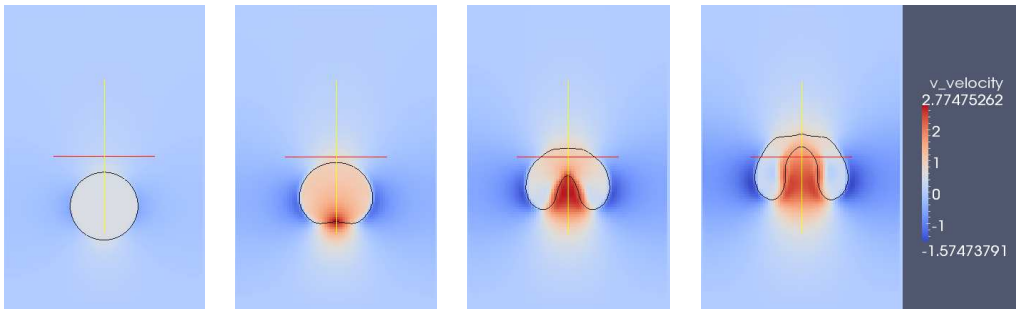

Figure 5: Evolution of the interface and vertical velocity for a large air bubble rising in air, from left to right at times $t = 0, 0.2, 0.35, 0.5$.

In both cases, the results are in good agreement with those presented in [3].

2.2 Collapse of a water column (dam break)

This test case is introduced in [2] and [4], and based on experiments conducted in [5]. The initial configuration is a water column at rest in air. For more details, we refer to [2]. The physical constants are the same than for the rising bubble in the previous subsection, excepted the air density that is slightly different: $\rho_{air} = 1.226 kg/m^3$. We present on figure 6 the interface evolution at non-dimensional times $T = t\sqrt{g/h} = 0, 1, 2, 3$, with h the initial height of the water column. The computations are performed with 60×240 points.

The front propagation is in agreement with the results in [2], which means that the errors in momentum transfer across the interface are negligible enough, though the method is not strictly speaking conservative. It is not the case for instance for the method of Kang et al. [3], as reported in [2].

3 Level-set strategy

In order to use a sharp method for air-water interfaces, it is necessary to accurately compute geometric properties (e.g. curvature) in the vicinity of the interface. Most common strategies in the litterature do not ensure the consistency for these quantities. The level-set approach we use is based on two key features:

1. reduce the error on the curvature of interface,

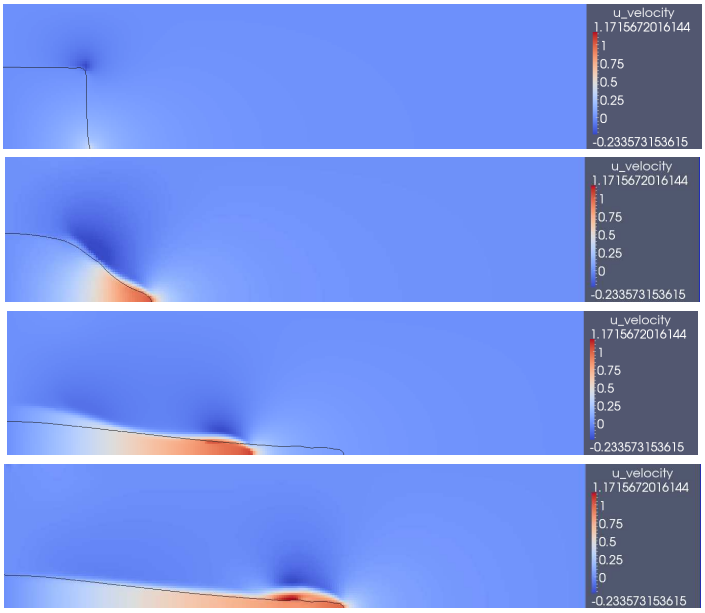

Figure 6: Evolution of the interface and horizontal velocity for the dam break problem from top to bottom at non-dimensional times $T = t\sqrt{g/h} = 0, 1, 2, 3$.

2. ensure at least a first order convergence for the curvature.

The first point concerns the amplitude of the error, while the second involves the consistency of the approximation of the curvature. As we show in the next section, even a first order approximation of the curvature might lead to a large error in the curvature. In the case of air-water interfaces, this may cause numerical issues when the surface tension plays a significant role.

3.1 Motivation : keeping the level-set function close to a distance function

When the level-set function is far from a distance function, small and/or large gradients may appear in the vicinity of the interface. Even with a first order approximation of the curvature, this may lead to a loss of accuracy. To illustrate this fact, we compute the curvature of a circle (center $(0, 0)$, radius 0.6) in three different manners:

- computing $\nabla \cdot \left(\dfrac{\nabla \phi_0}{|\nabla \phi_0|} \right)$ with

$$\phi_0(x, y) = \left(\frac{\sqrt{x^2 + y^2}}{0.6} - 1 \right) \left(\epsilon + (x - x_0)^2 + (y - y_0)^2 \right),$$

  $\epsilon = 0.02$, $x_0 = 0.7$ and $y_0 = 0.4$, and we interpolate on the interface,

- we use the signed distance function $d(x, y) = \sqrt{x^2 + y^2} - 0.6$ and we interpolate $\nabla \cdot \left(\frac{\nabla d}{|\nabla d|} \right)$ on the interface,

- we use the signed distance function and we interpolate Δd on the interface.

Results are reported in Figure 7 and show that using the distance function significantly reduce the error (by a factor of 100). This example shows that whenever possible, it is best to prevent the level-set function to have small and/or large gradients near the interface. One possibility that we investigate here is the use of a reinitialization step, in order to compute the distance function. The price to pay is that the reinitialization introduces some errors on the level-set function (e.g. on the position of the interface). It is then necessary to combine high-order computations of distance functions and a small number of reinitializations.

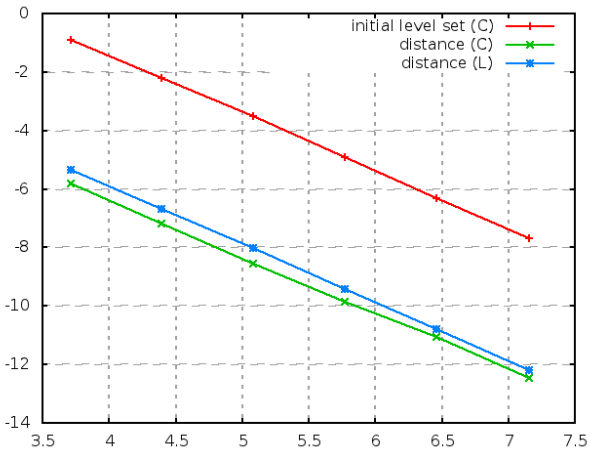

Figure 7: L^∞ error on the curvature (on the interface) vs $1/\Delta x$ (log-log representation) : with original level-set (red), distance function (green), using the laplacian of the distance function (blue).

3.2 High-order scheme for computing distance functions

Working on cartesian grids eases the parallelization of the numerical simulations. For the distance function, we try to find a good balance between computational cost and easy parallelization. In the litterature, several methods are described to compute distance functions. One may cite the Fast-Marching method (cf. [6] e.g.), the Fast-Sweeping method (cf. [7] e.g.) or relaxation algorithms (cf. [8] e.g.). The first two are very fast solvers, essentially of order 1 on the level-set. The latter is much slower, but higher-order schemes are more tractable. The Fast-Marching method is the less easy to parallelize, so we use a hybrid relaxation/fast sweeping method to compute distance function. The idea is to use a high-order relaxation method near the interface, and fast-sweeping method away from the interface.

Relaxation step. The equation we want to solve is the following: given ϕ_0, find ϕ such that

$$|\nabla\phi| = 1, \qquad \phi = 0 \text{ on } \{\phi_0 = 0\}. \tag{1}$$

Relaxation methods are based on the resolution of the PDE (with fictitious time τ)

$$\partial_\tau\phi + sgn(\phi_0)\frac{\nabla\phi}{|\nabla\phi|}\cdot\nabla\phi = sgn(\phi_0), \tag{2}$$

until the steady state is reached. Due to its analogy to a transport equation, this happens for $\tau \sim L_{\max}$ where $L_{\max}$ is the maximum distance between a boundary point and the interface. The pseudo-time discretization is usually $\Delta\tau = \Delta x/2$, so the number of iterations needed increases when refining the mesh. In [9] a third order method to solve this equation is presented. In order to get a good approximation of the distance function near the interface for the Fast-Sweeping step, we only need to solve (2) in the vicinity of the interface. We consider the band $B_n := \{d(x,\Gamma) \leq 5\Delta x\}$ and we compute:

$$\phi^{(n+1)} = \phi^{(n)} - \Delta\tau\ \mathrm{sgn}(\phi_0)\left(|\nabla\phi^{(n)}| - 1\right), \tag{3}$$

$$\phi^{(0)} = \phi_0, \tag{4}$$

with the numerical method of [9]. We stop the computation when

$$\|\phi^{(n+1)} - \phi^{(n)}\|_{L^1(B_n)} \leq (\Delta x)^4.$$

This stopping criterion ensures that the error on ϕ in the band B_n is of order 3, thus that the curvature is computed with first order accuracy. Then we apply a Fast-Sweeping step.

Fast-Sweeping step. Fast-Sweeping methods ([7] or [10] for high-order schemes) are based on Gauss-Seidel-like iterations to solve (1). The procedure to update ϕ at node (i,j) is the following: freeze all values of ϕ (except $\phi_{i,j}$ and find $\phi_{i,j}$ such that

$$\left[\left(\frac{\phi_{i,j}-\phi^x_{\min}}{\alpha\Delta x}\right)^+\right]^2+\left[\left(\frac{\phi_{i,j}-\phi^y_{\min}}{\alpha\Delta x}\right)^+\right]^2=1,$$

with

$$\phi^x_{\min}:=\begin{cases}\frac{4}{3}\phi_{i-1,j}-\frac{1}{3}\phi_{i-2,j} & \text{if } \phi_{i-1,j}\le\phi_{i+1,j}\\ \frac{4}{3}\phi_{i+1,j}-\frac{1}{3}\phi_{i+2,j} & \text{otherwise}\end{cases}\qquad \phi^y_{\min}:=\begin{cases}\frac{4}{3}\phi_{i,j-1}-\frac{1}{3}\phi_{i,j-2} & \text{if } \phi_{i,j-1}\le\phi_{i,j+1}\\ \frac{4}{3}\phi_{i,j+1}-\frac{1}{3}\phi_{i,j+2} & \text{otherwise}\end{cases},$$

$\alpha=2/3$ and $(x)^+:=\max(x,0)$. The scheme is as follows:

- *Initialization*: set values in the vicinity of the interface (i.e. in the band B_n). These values remain untouched after.
- *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.

This method is only second order accurate away from the interface, but the first relaxation step ensures the third order where needed. Besides, all high-order method in [10] have an increasing number of iterations when refining the mesh, while this second order method keeps it stable.

Coupling with transport. In practice, the common use is to reinitialize the distance function every 5 or 10 time steps. Since the reinitialization procedure sligthly moves the interface, this leads to a loss of accuracy on the distance function, and the consistency of the curvature is not necessarily ensured. To circumvent the problem, we choose to fix the number of reinitializations during the computations, irrespective of Δt or Δx. Recalling that we need to avoid formation of small and/or large gradients, we introduce the quantity $r_g:=\||\nabla\phi|-1\|$, that has to remain small. We choose a threshold ϵ and we proceed as follows: we transport the level-set function without changing it until $r_g>\epsilon$. When $r_g>\epsilon$, we compute the distance function and replace the level-set. Thus, for any space and time discretization, the numerical scheme approximates the same (continuous) problem defined by:

- *Initialization:* start with ϕ the signed distance function to the interface
- *Transport:* while $r_g<\epsilon$, evolve ϕ according to

$$\partial_t\phi+\boldsymbol{u}\nabla\phi=0.$$

- *Reinitialization:* when $r_g=\epsilon$, set $\phi_0=\phi$ and recompute the distance function to the interface (by solving (1)). Go to transport.

Then it makes sense to investigate a convergence order, and the expected high-order accuracy is shown in the next section.

4 Numerical validation of the level-set method

4.1 Reinitialization only

We first illustrate the performance of the reinitialization strategy. The computational domain is $\Omega=[-1,1]^2$. The interface Γ is the circle with center $(0,0)$ and radius 0.6. The initial level-set ϕ_0 and the distance function

are given by

$$\phi_0(x,y) = \left(\frac{\sqrt{x^2+y^2}}{0.6} - 1 \right) \left(\epsilon + (x-x_0)^2 + (y-y_0)^2 \right), \tag{5}$$

$$d(x,y) = \sqrt{x^2+y^2} - 0.6, \tag{6}$$

with $\epsilon = 0.02$, $x_0 = 0.7$ and $y_0 = 0.4$. The L^∞ error (on Ω) for ϕ and the $^\infty$ error on the curvature on the interface are reported in Table 1. The error for ϕ is taken over Ω so that we expect a global second order accuracy (third order is achieved in the vicinity of the interface). The first order accuracy for the curvature is recovered. The total number of iterations (relaxation iterations + fast-sweeping iterations) needed is not increasing too rapidly. Note that, using only the numerical scheme of [9] (relaxation method over the whole),

mesh	$\|d-\phi_h\|$	coc	$\|\kappa-\kappa_h\|$	coc	number of it.
40	3.78E-03	-	5.97E-02	-	23
80	1.81E-03	1.06	2.65E-02	1.17	28
160	6.62E-04	1.45	1.25E-02	1.08	32
320	2.06E-04	1.69	6.29E-03	1.00	36
640	5.77E-05	1.83	3.17E-03	0.99	41
1280	1.53E-05	1.92	1.57E-03	1.01	48

Table 1: Errors and computed order of convergence (coc) for the mixed method. Errors on ϕ (left), on κ (middle) and total number of iterations.

the computation on the 1280×1280 mesh would have required 2560 instead of 48 here.

4.2 Coupling transport & reinitialization

We illustrate the good behaviour of the method on the case with transport. We assume that the interface is passively advected by a given velocity field. The computational domain is $\Omega = [0,1]^2$. The initial interface Γ_0 is the circle with center $(\frac{1}{2}, \frac{3}{4})$ and radius 0.15. The velocity field is given by the scalar potential :

$$\boldsymbol{u} = \nabla^\perp \omega, \qquad \omega = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \cos\left(\frac{\pi t}{T}\right), \tag{7}$$

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 1/640, and perform simulations for $t \in [0,4]$. We use the threshold $\epsilon = 0.15$ for reinitialization. The computation of the signed distance is enforced at $t = 4$ and for each mesh. We compute the curvature of the interface at $t = 2$ (which is not a reinitialization time) and $t = 4$. Results are reported in Table 2. As expected with this strategy, the total

mesh	L^∞ error	coc	L^∞ error	coc
40	0.919E+00	-	0.108E+01	-
80	0.291E+00	1.66	0.363E+00	1.57
160	0.934E-01	1.64	0.921E-01	1.98
320	0.301E-01	1.64	0.343E-01	1.43
640	0.369E-02	3.03	0.422E-02	3.02

Table 2: L^∞-errors on the curvature of the interface: at $t = 2$ (left) and $t = 4$ (right).

number of computations of the signed distance on the fly is 16, in each simulation.

5 Conclusion

We have presented two new tools for the simulation of air-water interface (or any kind of flows with discontinuous densities and viscosities) on cartesian grids:

- a sharp method to compute the evolution of the fluid, with a second order treatment of the correction term,
- a high order level-set technique which allow us to compute consistently the curvature of the interface even for long times.

There are very few references to cartesian methods for incompressible flows with discontinuous densities and viscosities in the literature: to our knowledge, only the pioneering work of [3], the conservative method of [2] where a more sophisticated treatment of the convective terms, in the "cut-cell" spirit, is used, and [11], where the interface does not move. From this point of view, our new method provides another alternative, yielding satisfactory results for moving interfaces, without any interface reconstruction, and however maintaining a small level of conservation errors.

The usual level-set strategies involve either a low order approximation of the distance function or a too large number of reinitialization steps, leading to a loss of consistency for geometrical properties. The technique we presented here provides a consistent computation of the curvature of the interface, yet with a reasonable computational cost. It is then suitable if for long time simulations, and might be used in a more general framework than air-water interfaces.

References

[1] Marco Cisternino and Lisl Weynans. A parallel second order Cartesian method for elliptic interface problems. *Commun. Comput. Phys.*, 12(5):1562–1587, 2012.

[2] M. Raessi and H. Pitsch. Consistent mass and momentum transport for simulating incompressible interfacial flows with large density ratios using the level set method. *Computers and Fluids*, 63:70–81, 2012.

[3] Myungjoo Kang, Ronald P. Fedkiw, and Xu-Dong Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 15(3):323–360, 2000.

[4] V. Le Chenadec and H. Pitsch. A monotonicity preserving conservative sharp interface flow solver for high density ratio two-phase flows. *J. Comput. Phys.*, 249:185–203, 2013.

[5] J. C. Martin and W. J. Moyce. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philos. Trans. R. Soc. London, Ser. A*, 244:312–324, 1952.

[6] J. A. Sethian. *Level set methods and fast marching methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, second edition, 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.

[7] Yen-Hsi Richard Tsai, Li-Tien Cheng, Stanley Osher, and Hong-Kai Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41(2):673–694, 2003.

[8] M. Smereka M. Sussman and S. Osher. A level-set approach for computing solutions to incompressible two-phase flows. *Journal of Computational Physics*, 114:146–159, 1994.

[9] Antoine du Chéné, Chohong Min, and Frédéric Gibou. Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes. *J. Sci. Comput.*, 35(2-3):114–131, 2008.

[10] Yong-Tao Zhang, Hong-Kai Zhao, and Jianliang Qian. High order fast sweeping methods for static Hamilton-Jacobi equations. *J. Sci. Comput.*, 29(1):25–56, 2006.

[11] Y. C. Zhou, J. Liu, and D. Harry. A matched interface and boundary method for solving multi-flow navier-stokes equations with applications to geodynamics. *J. Comput. Phys.*, 231:223–242, 2012.