Calculation of the interface curvature and normal vector with the level-set method

Karl Yngve Lervåg[a,*], Bernhard Müller[a], Svend Tollak Munkejord[b]

[a]*Department of Energy and Process Engineering, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*
[b]*SINTEF Energy Research, P.O. Box 4761 Sluppen, NO-7465 Trondheim, Norway*

Abstract

This article addresses the use of the level-set method for capturing the interface between two fluids. One of the advantages of the level-set method is that the curvature and the normal vector of the interface can be readily calculated from the level-set function. However, in cases where the level-set method is used to capture topological changes, the standard discretization techniques for the curvature and the normal vector do not work properly. This is because they are affected by the discontinuities of the signed-distance function half-way between two interfaces. This article addresses the calculation of normal vectors and curvatures with the level-set method for such cases. It presents a discretization scheme based on the geometry-aware curvature discretization by Macklin and Lowengrub [1]. As the present scheme is independent of the ghost-fluid method, it becomes more generally applicable, and it can be implemented into an existing level-set code more easily than Macklin and Lowengrub's scheme [1]. The present scheme is compared with the second-order central-difference scheme and with Macklin and Lowengrub's scheme [1], first for a case with no flow, then for a case where two drops collide in a 2D shear flow, and finally for a case where two drops collide in an axisymmetric flow. In the latter two cases, the Navier-Stokes equations for incompressible two-phase flow are solved. The article also gives a comparison of the calculation of normal vectors with the direction difference scheme presented by Macklin and Lowengrub in [2] and with the present dis-

*Corresponding author
Email address: `karl.y.lervag@ntnu.no` (Karl Yngve Lervåg)
URL: `http://folk.ntnu.no/lervag` (Karl Yngve Lervåg)

cretization scheme. The results show that the present discretization scheme yields more robust calculations of the curvature than the second-order central difference scheme in areas where topological changes are imminent. The present scheme compares well to Macklin and Lowengrub's method [1]. The results also demonstrate that the direction difference scheme [2] is not always sufficient to accurately calculate the normal vectors.

Keywords: Level-set method, Curvature discretization, Normal-vector discretization, Curve-fitting discretization scheme, Finite differences, Ghost-fluid method.

1. Introduction

The level-set method was introduced by Osher and Sethian [3]. It is designed to implicitly track moving interfaces through an isocontour of a function defined in the entire domain. In particular, it is designed for problems in multiple spatial dimensions in which the topology of the evolving interface changes during the course of events, cf. [4].

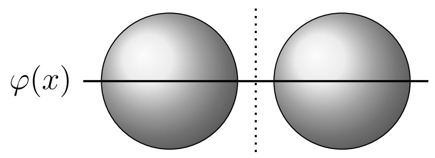
This article addresses the calculation of interface geometries with the level-set method. This method allows us to calculate the normal vector and the curvature of an interface directly as the first and second derivatives of the level-set function. These calculations are typically done with standard finite-difference methods. Since the level-set function is chosen to be a signed-distance function, in most cases it will have areas where it is not smooth. Consider for instance two colliding drops where the interfaces are captured with the level-set method, see Figure 1. The derivative of the level-set function will not be defined at the points outside the drops that have an equal distance to both drops. When the drops are in near contact, this discontinuity in the derivative will lead to significant errors when calculating the interface geometries with standard finite-difference methods. For convenience the areas where the derivative of the level-set function is not defined will hereafter be referred to as kinks.

To the authors knowledge, this issue was first described in [2], where the level-set method was used to model tumour growth. Here Macklin and Lowengrub presented a direction difference to treat the discretization across kinks for the normal vector and the curvature. They later presented an improved method where curve fitting was used to calculate the curvatures [1]. This was further expanded to include the normal vectors [5].

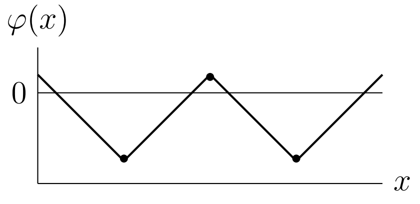
An alternative method to avoid the kinks is presented in [6], where a level-set extraction technique is presented. This technique uses an extraction algorithm to reconstruct separate level-set functions for each distinct body.

Accurate calculation of the curvature is important in many applications, in particular in curvature-driven flows. There are several examples in the literature of methods that improve the accuracy of the curvature calculations, but that do not consider the problem with the discretization across the kinks. The authors in [7] use a coupled level-set and volume-of-fluid method based on a fixed Eulerian grid, and they use a height function to calculate the curvatures. In [8] a refined level-set grid method is used to study two-phase flows on structured and unstructured grids for the flow solver. An interface-projected curvature-evaluation method is presented to achieve converging calculation of the curvature. Marchandise et. al [9] adopt a discontinuous Galerkin method and a pressure-stabilized finite-element method to solve the level-set equation and the Navier-Stokes equations, respectively. They develop a least-squares approach to calculate the normal vector and the curvature accurately, as opposed to using a direct derivation of the level-set function. This method is used by Desjardins et. al in [10], where they show impressive results for simulations of turbulent atomization.

This article is a continuation of the work presented in [11]. It applies the level-set method and the ghost-fluid method to incompressible two-phase flow in two dimensions. A curve-fitting discretization scheme is presented which is based on the geometry-aware discretization given in [1]. This scheme is mainly applied to the curvature discretization. The normal vectors are calculated both with the direction difference described in [2] and with a combination of the direction difference and the curve-fitting discretization



(a) Drops in near contact



(b) A slice of the level-set function

Figure 1: (a) Two drops in near contact. The dotted line marks a region where the derivative of the level-set function is not defined. (b) A one-dimensional slice of the level-set function $\varphi(x)$. The dots mark points where the derivative of $\varphi(x)$ is not defined.

scheme.

The main advantage of the present scheme compared to the geometry-aware discretization [1] is that it is independent of the ghost-fluid method. That is, in [1] Macklin and Lowengrub calculate the curvature values directly on the interface when it is needed by the ghost-fluid method, whereas we compute the curvature values at the global grid points, indendent of the ghost-fluid method. Because of this, the scheme can be implemented more easily into existing Navier-Stokes codes employing the level-set method, since only small parts of the existing codes need modification. It is also more generally applicable, for instance it can be used with the continuum surface-force method [15]. Further, it allows for more accurate curvature values in models that require curvature values on the grid instead of on the interface, e.g. surfactant models [12–14].

The article starts by briefly describing the governing equations for two-phase flow and the level-set method in Section 2. It continues in Section 3 with a description of the numerical methods that are used for their solution. Then the discretization schemes for the normal vector and the curvature are presented in Section 4, followed by a detailed description of the method for curvature discretization in Section 5. Section 6 gives a convergence test and a comparison of the present discretization scheme with the second-order central difference scheme and Macklin and Lowengrub's scheme [1], first on static interfaces in near contact, then on two drops colliding in a 2D shear flow, and finally on a case where two drops collide in an axisymmetric flow. The section is concluded with a comparison of the direction difference scheme [2] with a combination of the direction difference and the curve-fitting discretization schemes for calculating normal vectors. Finally in Section 7 concluding remarks are made.

2. Governing equations

2.1. Navier-Stokes equations for two-phase flow

Consider a two-phase domain $\Omega = \Omega^+ \cup \Omega^-$ where Ω^+ and Ω^- denote the regions occupied by the respective phases. The domain is divided by an interface $\Gamma = \delta\Omega^+ \cap \delta\Omega^-$ as illustrated in Figure 2. The governing equations for incompressible and immiscible two-phase flow in the domain Ω with an

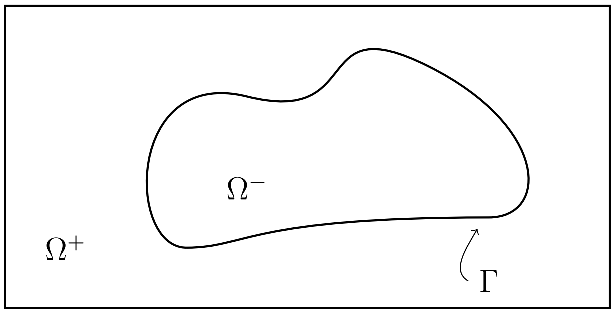


Figure 2: Illustration of a two-phase domain: The interface Γ separates the two phases, one in Ω^+ and the other in Ω^-.

interface force on the interface Γ can be stated as

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0, \tag{1}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}\right) = -\boldsymbol{\nabla}p + \boldsymbol{\nabla}\cdot\left(\mu\left(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T\right)\right) + \rho\boldsymbol{f}_b + \int_\Gamma \sigma\kappa\boldsymbol{n}\,\delta(\boldsymbol{x}-\boldsymbol{x}_I(s))\,\mathrm{d}s, \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, $\boldsymbol{f}_b$ is the specific body force, σ is the coefficient of surface tension, κ is the curvature, $\boldsymbol{n}$ is the normal vector which points to Ω^+, δ is the Dirac Delta function, $\boldsymbol{x}_I$ is a parametrization of the interface, ρ is the density and μ is the viscosity. These equations are often called the Navier-Stokes equations for incompressible two-phase flow.

It is assumed that the density and the viscosity are constant in each phase, but they may be discontinuous across the interface. The interface force and the discontinuities in the density and the viscosity lead to a set of interface conditions,

$$[\boldsymbol{u}] = 0, \tag{3}$$

$$[p] = 2[\mu]\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{n} + \sigma\kappa, \tag{4}$$

$$[\mu\boldsymbol{\nabla}\boldsymbol{u}] = [\mu]\big((\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n}\otimes\boldsymbol{n} + (\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{n}\otimes\boldsymbol{t} - (\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{t}\otimes\boldsymbol{n} + (\boldsymbol{t}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{t}\otimes\boldsymbol{t}\big), \tag{5}$$

$$[\boldsymbol{\nabla}p] = 0, \tag{6}$$

where $\boldsymbol{t}$ is the tangent vector along the interface, $\otimes$ denotes the dyadic product, and $[\,\cdot\,]$ denotes the jump across an interface, that is

$$[\mu] \equiv \mu^+ - \mu^-. \tag{7}$$

See [16, 17] for more details and a derivation of the interface conditions.

2.2. Level-set method

The interface is captured with the zero level set of the level-set function $\varphi(\boldsymbol{x}, t)$, which is prescribed as a signed-distance function. That is, the interface is given by

$$\Gamma = \{ (\boldsymbol{x}, t) \mid \varphi(\boldsymbol{x}, t) = 0 \}, \quad \boldsymbol{x} \in \Omega, \quad t \in \mathbb{R}^+, \tag{8}$$

and for any $t \geq 0$,

$$\varphi(\boldsymbol{x}, t) \begin{cases} < 0 & \text{if } \boldsymbol{x} \in \Omega^- \\ = 0 & \text{if } \boldsymbol{x} \in \Gamma \\ > 0 & \text{if } \boldsymbol{x} \in \Omega^+ \end{cases} . \tag{9}$$

The interface is updated by solving an advection equation for φ,

$$\frac{\partial \varphi}{\partial t} + \hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla} \varphi = 0, \tag{10}$$

where $\hat{\boldsymbol{u}}$ is the velocity at the interface extended to the entire domain. The interface velocity is extended from the interface to the domain by solving

$$\frac{\partial \hat{\boldsymbol{u}}}{\partial \tau} + S(\varphi) \boldsymbol{n} \cdot \boldsymbol{\nabla} \hat{\boldsymbol{u}} = 0, \quad \hat{\boldsymbol{u}}_{\tau=0} = \boldsymbol{u}, \tag{11}$$

to steady state, cf. [18]. Here τ is pseudo-time and S is a smeared sign function which is equal to zero at the interface,

$$S(\varphi) = \frac{\varphi}{\sqrt{\varphi^2 + 2\Delta x^2}}. \tag{12}$$

When equation (10) is solved numerically, the level-set function loses its signed-distance property due to numerical dissipation. The level-set function is therefore reinitialized regularly by solving

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + S(\varphi_0)(|\boldsymbol{\nabla} \varphi| - 1) = 0, \\ \varphi(\boldsymbol{x}, 0) = \varphi_0(\boldsymbol{x}), \end{aligned} \tag{13}$$

to steady state as proposed in [19]. Here φ_0 is the level-set function that needs to be reinitialized.

One of the advantages of the level-set method is that normal vectors and curvatures can be readily calculated from the level-set function, i.e.

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}, \tag{14}$$

$$\kappa = \boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|} \right). \tag{15}$$

3. Numerical methods

The Navier-Stokes equations, (1) and (2), are solved by a projection method on a staggered grid as described in [17, Chapter 5.1.1]. The spatial terms are discretized by the second-order central difference scheme, except for the convective terms which are discretized by a fifth-order WENO scheme. The temporal discretization is done with explicit strong stability-preserving Runge-Kutta (SSP RK) schemes, see [20]. A three-stage third-order SSP-RK method is used for the Navier-Stokes equations (1) and (2), and a four-stage second-order SSP-RK method is used for the level-set equations (10), (11) and (13).

The method presented in [21] is used to improve the computational speed. The method is often called the narrow-band method, since the level-set function is only updated in a narrow band across the interface at each time step.

The interface conditions are treated in a sharp fashion with the Ghost-Fluid Method (GFM), which incorporates the discontinuities into the discretization stencils by altering the stencils close to the interfaces. For instance, the GFM requires that a term is added to the stencil on the right-hand side of the Poisson equation for the pressure. Consider a one-dimensional case where $[\rho] = [\mu] = 0$ and where the interface lies between x_i and x_{i+1}. In this case,

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{\Delta x^2} = f_k \pm \frac{\sigma \kappa_\Gamma}{\Delta x^2}, \tag{16}$$

where f_k is the general right-hand side value and κ_Γ is the curvature at the interface. The sign of the added term depends on the sign of $\varphi(x_i)$. See [16] for more details on how the GFM is used for the Navier-Stokes equations and [22] for a description on how to use the GFM for a variable-coefficient Poisson equation.

The normal vector and the curvature defined by equations (14) and (15) are typically discretized by the second-order central difference scheme, cf. [4,

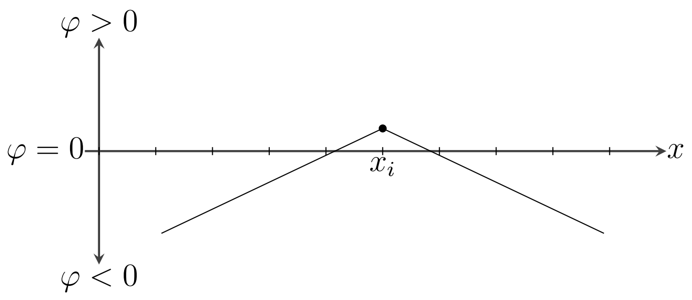


Figure 3: A level-set function that has one point where the derivative is discontinuous.

12, 16]. The curvatures are calculated on the grid nodes and then interpolated with simple linear interpolation to the interface, e.g. for κ_Γ in equation (16),

$$\kappa_\Gamma = \frac{|\varphi_i|\kappa_{i+1} + |\varphi_{i+1}|\kappa_i}{|\varphi_i| + |\varphi_{i+1}|}. \tag{17}$$

If the level-set method is used to capture non-trivial geometries, the level-set function will in general contain areas where it is not smooth, i.e. kinks. This is depicted in Figure 3, which shows a level-set function in a one-dimensional domain that captures two interfaces, one on each side of the grid point x_i. The kink at x_i will lead to potentially large errors with the standard discretization both for the curvature and the normal vector. The errors in the curvature will lead to erroneous pressure jumps at the interfaces, and the errors in the normal vector affects both the discretized interface conditions and the advection of the level-set function. If the level-set method is used to study for example coalescence and breakup of drops, these errors may severely affect the simulations.

It should be noted that the kinks that appear far from any interfaces are handled by ensuring that the denominators do not become zero, as explained in [23, Sections 2.3 to 2.4]. This works fine, since only the values of the curvature at the grid nodes adjacent to any interface are used. Also, the normal vector only needs to be accurate close to the interface due to the narrow-band approach.

4. Improved discretization of geometrical quantities

The previous section explained why it is necessary to develop new discretization schemes for the normal vector and the curvature that can handle kinks in the level-set function. This section will give an overview of the curve-fitting discretization scheme and how it is applied to calculate the curvature. It will then give a brief presentation of the direction difference which is used to calculate the normal vector. A note is finally given on how to use the curve-fitting discretization scheme to calculate the normal vector.

4.1. The curvature

The curvature is calculated with a discretization that is based on the improved geometry-aware curvature discretization presented by Macklin and Lowengrub [1]. This is a method where the curvature is calculated at the interfaces directly with the use of a least-squares curve parametrization of the interface. The curve parametrization is used to create a local level-set function from which the curvature is calculated using standard discretization techniques. The local level-set function only depends on one interface and is therefore free of kinks.

The main motivation behind the present method is to improve the curvature calculations specifically at the grid points, as these values may be important for other models. Examples of such cases are the modelling of interfacial flows with surfactants [12–14].

The main difference between the present method and that of Macklin and Lowengrub [1] is that they modify the GFM to calculate the curvature at the interface directly, whereas the present method only changes the procedure to calculate the curvature at specific grid points. In other words, Macklin and Lowengrub calculate κ_Γ in equation (16) directly with a parametrized curve, whereas the present method uses parametrized curves to calculate κ_i and κ_{i+1}. The present method is therefore independent of the GFM, which makes it easier to adopt it into existing level-set codes.

An important consequence of not calculating the curvature directly on the interface is that it becomes more important to have an accurate representation of the interface. This is due to the fact that the point $\boldsymbol{x}_i$ where the curvature is calculated is not on the interface, so the calculation becomes less local. Thus the parametrization needs to be more accurate at a larger distance from $\boldsymbol{x}_i$. The curve-fitting discretization scheme presented here uses monotone cubic Hermite splines to parametrize the curve. The least-square parametrization used in [1] is only accurate very close to the point where the curvature needs to be calculated. The Hermite spline is more accurate along the entire interface representation.

The algorithm to calculate the curvature at $\boldsymbol{x}_{i,j}$ can be summarized as follows. The details are explained in the next section.

1. If $Q_{i+n,j+m} \leq \eta$, where $n = -1, 0, 1$ and $m = -1, 0, 1$, then it is safe to use the central-difference discretization. Otherwise continue to the next step.
2. Locate the closest interface, Γ.
3. Find a set of points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \Gamma$.
4. Create a parametrization $\boldsymbol{\gamma}(s)$ of the points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$.
5. Calculate a local level-set function based on the parametrization $\boldsymbol{\gamma}(s)$.
6. Use central-difference discretization on the local level-set function to calculate the curvature.

4.2. The normal vector

This section will describe two methods to calculate the normal vector close to kinks. The direction difference [2] will be described first. Then a method based on the curve-fitting scheme will be presented.

4.2.1. Direction difference

The direction-difference scheme uses a quality function to ensure that the difference stencils never cross kinks. The basic strategy is to use a combination of central differences and one-sided differences based on the values of a quality function,

$$Q(\boldsymbol{x}) = \left| 1 - |\boldsymbol{\nabla}\varphi(\boldsymbol{x})| \right|, \tag{18}$$

which is approximated with central differences. The quality function effectively detects the areas where the level-set function differs from the signed-distance function. Let $Q_{i,j} = Q(\boldsymbol{x}_{i,j})$ and $\eta > 0$, then $Q_{i,j} > \eta$ can be used to detect kinks. The parameter η is tuned such that the quality function will detect all the kinks. The value $\eta = 0.1$ is used in the present work.

The quality function is used to define a direction function,

$$\boldsymbol{D}(\boldsymbol{x}_{i,j}) = (D_x(\boldsymbol{x}_{i,j}), D_y(\boldsymbol{x}_{i,j})), \tag{19}$$

where

$$D_x(\boldsymbol{x}_{i,j}) = \begin{cases} -1 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i+1,j} \geq \eta, \\ 1 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i,j} < \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i,j} \geq \eta \text{ and } Q_{i+1,j} \geq \eta, \\ \text{undetermined} & \text{otherwise.} \end{cases} \tag{20}$$

$D_y(\boldsymbol{x}_{i,j})$ is defined in a similar manner. If D_x or D_y is undetermined, $\boldsymbol{D}(\boldsymbol{x}_{i,j})$ is chosen as the vector normal to $\boldsymbol{\nabla}\varphi(\boldsymbol{x}_{i,j})$. It is normalized, and the sign is chosen such that it points in the direction of best quality. See [2] for more details.

The direction difference is now defined as

$$\partial_x f_{i,j} = \begin{cases} \frac{f_{i,j}-f_{i-1,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = -1, \\ \frac{f_{i+1,j}-f_{i,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = 1, \\ \frac{f_{i+1,j}-f_{i-1,j}}{2\Delta x} & \text{if } D_x(x_i, y_j) = 0, \end{cases} \tag{21}$$

where $f_{i,j}$ is a piecewise smooth function. The normal vector is calculated using the direction difference on φ, which is equivalent to using central differences in smooth areas and one-sided differences in areas close to the kinks. This method makes sure that the differences do not cross any kinks, and the normal vector can be accurately calculated even close to a kink.

4.2.2. Curve-fitting scheme

The direction difference is an elegant scheme which performs well in most cases. However, it will be shown later that in some rare cases where both direction functions are undetermined, this discretization scheme may become very inaccurate. An alternative is to use the curve-fitting discretization scheme on the normal vectors. But since this method starts by locating the closest interface with a breadth-first search (see next section), it will be slow when it is used far from any interfaces. It is therefore proposed to use a combination of the direction difference and the curve-fitting discretization scheme.

5. The curve-fitting discretization scheme

5.1. Locating the closest interface

A breadth-first search is used to to identify the closest interface, see Figure 4. Let $\boldsymbol{x}_0$ denote the starting point and $\boldsymbol{x}_1$ denote the desired point on the closest interface. The search iterates over all the eight edges from $\boldsymbol{x}_0$ to its neighbours and tries to locate an interface which is identified by a change of sign of $\varphi(\boldsymbol{x})$. If more than one interface is found, $\boldsymbol{x}_1$ is chosen to be the point that is closest to $\boldsymbol{x}_0$. If no interfaces are located the search continues at the next depth. The search continues in this manner until an interface is found. Note that this algorithm does not in general return the point on the interface which is closest to $\boldsymbol{x}_0$.

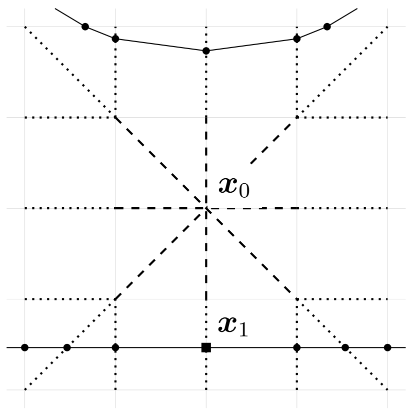


Figure 4: Sketch of a breadth-first search. The dashed lines depict the edges that are searched first, the dotted lines depict the edges that are searched next and the solid lines depict two interfaces. The circular dots mark where the algorithm finds interface points, and the rectangular dot marks the point which is returned for the depicted case.

The crossing points between the grid edges and the interfaces are found with linear and bilinear interpolation. E.g. if an interface crosses the edge between (i, j) and $(i, j+1)$ at $\boldsymbol{x}_I$, the interface point is found by linear interpolation,

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(0, \Delta x), \tag{22}$$

where

$$\theta = \frac{\varphi(\boldsymbol{x}_{i,j})}{\varphi(\boldsymbol{x}_{i,j}) - \varphi(\boldsymbol{x}_{i,j+1})}. \tag{23}$$

In the diagonal cases the interface point is found with bilinear interpolation along the diagonal. This leads to

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(\Delta x, \Delta x), \tag{24}$$

where θ is the solution of

$$\alpha_1\theta^2 + \alpha_2\theta + \alpha_3 = 0. \tag{25}$$

The α values depend on the grid cell. For instance, when searching along the diagonal between (i, j) and $(i+1, j+1)$ the α values will be

$$\alpha_1 = \varphi_{i,j} - \varphi_{i+1,j} - \varphi_{i,j+1} + \varphi_{i+1,j+1}, \tag{26}$$

$$\alpha_2 = \varphi_{i+1,j} + \varphi_{i,j+1} - 2\varphi_{i,j}, \tag{27}$$

$$\alpha_3 = \varphi_{i,j}. \tag{28}$$

5.2. Searching for points on an interface

When an interface and a corresponding point $\boldsymbol{x}_1$ on the interface are found, the next step is to find a set of points $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_n$ on the same interface. The points should be ordered such that when traversing the points from $k = 1$ to $k = n$, the phase with $\varphi(\boldsymbol{x}) < 0$ is on the left-hand side. Note that the ordering of the points may be done after all the points are found. Three criteria are used when searching for new points:

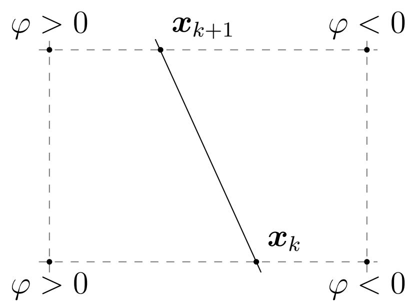
1. The points are located on the grid edges.
2. The distance between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ for all k is greater than a given threshold μ.
3. The n points that are closest to $\boldsymbol{x}_0$ are selected, where $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$ is the initial point where the curvature is to be calculated.

Let $\boldsymbol{x}_k \in \Gamma \cap [x_i, x_{i+1}) \times [y_j, y_{j+1})$ be given. To find a new point $\boldsymbol{x}_{k+1}$ on Γ, a variant of the marching-squares algorithm[1] is used. Given $\boldsymbol{x}_k$ and a search direction which is either clockwise or counter clockwise, the algorithm searches for all the points where an interface crosses the edges of the mesh rectangle $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. In most cases there will be two such points and $\boldsymbol{x}_k$ is one of them. $\boldsymbol{x}_{k+1}$ is then selected based on the search direction. If $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, the search is continued in the adjacent mesh rectangle. The search process is depicted in Figure 5(a).

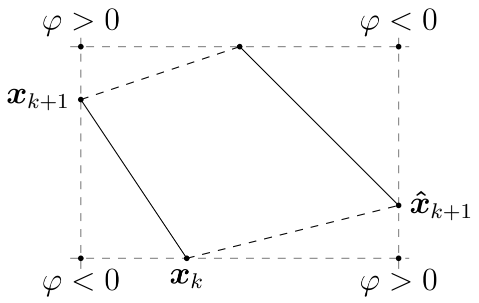
In some rare cases the algorithm must handle the ambiguous case depicted in Figure 5(b). In these cases there are four interface crossing-points and two solutions. Either solution is valid, and it is not possible to say which solution is better. The current implementation selects the first solution that it finds, which will be in all practical sense a random choice. Note that the ambiguous cases only occur when two interfaces cross a single grid cell. The ambiguity comes from the fact that the level-set method is not able to resolve the interfaces on a sub-cell resolution.

It was found that $n = 7$ points where necessary in order to ensure that the closest points on the interface with respect to the different grid points are captured with the spline parametrization.

[1]The marching-squares algorithm is an equivalent two-dimensional formulation of the well known marching-cubes algorithm presented in [24]. The algorithm was mainly developed for use in computer graphics.



(a) Marching squares



(b) Ambiguous case

Figure 5: (a) The search starts by locating the two points where the interface crosses the mesh rectangle. $\boldsymbol{x}_k$ is the starting point, and if the search is counter clockwise it will select $\boldsymbol{x}_{k+1}$ as depicted. If the search is clockwise, it will select $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, and the search continues in the adjacent mesh rectangle $[x_i, x_{i+1}] \times [y_{j-1}, y_j]$. (b) An example of an ambiguous case. The solid black lines and the dashed black lines are two equally valid solutions for how the interfaces cross the mesh rectangle. If the search starts at $\boldsymbol{x}_k$ and searches counter clockwise, then both $\hat{\boldsymbol{x}}_{k+1}$ and $\boldsymbol{x}_{k+1}$ are valid solutions.

5.3. Curve fitting

Cubic Hermite splines are used to fit a curve to the set of points

$$X_{0,m} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}. \tag{29}$$

Let the curve parametrization be denoted $\boldsymbol{\gamma}(s)$ for $0 < s < 1$. A cubic spline is a parametrization where

$$\boldsymbol{\gamma}(s) = \begin{cases} \boldsymbol{\gamma}_1(s) & s_0 \leq s < s_1, \\ \boldsymbol{\gamma}_2(s) & s_1 \leq s < s_2, \\ \vdots & \\ \boldsymbol{\gamma}_m(s) & s_{m-1} \leq s \leq s_m, \end{cases} \tag{30}$$

where $0 = s_0 < s_1 < \cdots < s_m = 1$

$$\boldsymbol{\gamma}(s_i) = \boldsymbol{x}_i, \quad 0 \leq i \leq m, \tag{31}$$

and each interpolant $\boldsymbol{\gamma}_i(s) = (x_i(s), y_i(s))$ is a third-order polynomial. A Hermite spline is a spline where each interpolant is in Hermite form, see [25, Chapter 4.5]. The interpolants are created by solving the equations

$$\boldsymbol{\gamma}_i(s) = h_{00}(s)\boldsymbol{x}_{i-1} + h_{01}(s)\boldsymbol{x}_i + h_{10}(s)\boldsymbol{m}_{i-1} + h_{11}(s)\boldsymbol{m}_i, \tag{32}$$

for $1 \leq i \leq m$, where $\boldsymbol{m}_i$ is the curve tangents and h_{00}, h_{01}, h_{10} and h_{11} are Hermite basis polynomials,

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1, \\ h_{01}(s) &= s^3 - 2s^2 + s, \\ h_{10}(s) &= -2s^3 + 3s^2, \\ h_{11}(s) &= s^3 - s^2. \end{aligned} \tag{33}$$

The choice of the tangents is non-unique, and there are several possible options for a cubic Hermite spline.

It is essential that the spline is properly oriented. This is because we require to find both the distance and the position of a point on the grid relative to the spline in order to define a local level-set function. The orientation of the spline $\boldsymbol{\gamma}(s)$ is defined such that when s increases, Ω^- is to the left.

To ensure that our curve is properly oriented, the tangents are chosen as described in [26]. This will ensure monotonicity for each component as long as the input data is monotone. The tangents are modified as follows. First the slopes of the secant lines between successive points are computed,

$$\boldsymbol{d}_i = \frac{\boldsymbol{x}_i - \boldsymbol{x}_{i-1}}{s_i - s_{i-1}} \tag{34}$$

for $1 \leq i \leq m$. Next the tangents are initialized as the average of the secants at every point,

$$\boldsymbol{m}_i = \frac{\boldsymbol{d}_i + \boldsymbol{d}_{i+1}}{2} \tag{35}$$

for $1 \leq i \leq m-1$. The curve tangents at the endpoints are set to $\boldsymbol{m}_0 = \boldsymbol{d}_1$ and $\boldsymbol{m}_m = \boldsymbol{d}_m$. Finally let k pass from 1 through $m-1$ and set $\boldsymbol{m}_k = \boldsymbol{m}_{k+1} = 0$ where $\boldsymbol{d}_k = 0$, and $\boldsymbol{m}_k = 0$ where $\operatorname{sign}(\boldsymbol{d}_k) \neq \operatorname{sign}(\boldsymbol{d}_{k+1})$.

5.4. Local level-set function

The local level-set function, here denoted as $\phi(\boldsymbol{x}_{i,j}) \equiv \phi_{i,j}$, is calculated at the grid points surrounding and including $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$. The curvature is then calculated with the standard discretization stencil where ϕ is used instead of the global level-set function, φ.

A precise definition of ϕ is

$$\phi(\boldsymbol{x}_{i,j}) = \min_s \left(\hat{d}(\boldsymbol{x}_{i,j}, \boldsymbol{\gamma}(s)) \right) \tag{36}$$

where $\hat{d}(\boldsymbol{x}, \boldsymbol{\gamma}(s))$ is the signed-distance function, which is negative in phase one and positive in phase two. This function is calculated by first finding the minimum distance between $\boldsymbol{x}$ and $\boldsymbol{\gamma}(s)$ and then deciding the correct sign. The minimum distance is found by minimizing the norm

$$d(\boldsymbol{x}, \boldsymbol{\gamma}(s)) = \|\boldsymbol{x} - \boldsymbol{\gamma}(s)\|_2. \tag{37}$$

When $\boldsymbol{\gamma}$ is composed of cubic polynomials as is the case for cubic Hermite splines, the computation of the distance requires the solution of several fifth-order polynomial equations. Sturm's method (see [27, Section 11.3] or [28, Chapter XI,§2]) is employed to locate and bracket the solutions and a combined Newton-Raphson and bisection method is used to refine them. The correct sign is found by solving

$$\operatorname{sign}(\phi(\boldsymbol{x}_{i,j})) = \operatorname{sign}\left((\boldsymbol{x}_{i,j} - \boldsymbol{\gamma}(s)) \times \boldsymbol{t}_{\boldsymbol{\gamma}}(s)\right)_z, \tag{38}$$

where $\boldsymbol{t}_{\boldsymbol{\gamma}}(s)$ is the tangent vector of $\boldsymbol{\gamma}(s)$.

6. Verification and testing

This section presents results of calculating normal vectors and curvatures with the present discretization scheme. First a convergence test is considered. Then in the following three cases simulation results are compared with the second-order central difference scheme and Macklin and Lowengrub's scheme [1]. In the final case, the direction difference [2] is compared with the curve-fitting scheme outlined in Section 4.2.2.

6.1. Convergence test

The convergence of the present curve-fitting method is measured on a simple test case depicted in Figure 6. Here a disc of radius $r = 0.25$ and curvature $\kappa_0 = -4$ is placed a distance $h = 1.1\Delta x$ over a rectangle. The grid is aligned such that a grid cell fits between the disc and the rectangle, see Figure 7. The error for a grid of size $n \times n$ is defined as the 1-norm of the difference between κ_0 and the curvature values κ_i at the disc interface,

$$E_n = \frac{1}{m} \sum_{i=1}^{m} |\kappa_0 - \kappa_i|, \tag{39}$$

where m is the number of curvature values along the interface. The curvature values κ_i are calculated with linear interpolation (17) along the interface of the disc.

The convergence results for several different grid sizes n are shown in Table 1. The curvature calculated with central differences does not converge due to an error $\mathcal{O}(1/\Delta x)$ introduced by the kink region. It is seen that the present method converges, although the convergence order jumps between 0.6 and 3. Since h depends on the grid size, the case is slightly altered for each grid refinement. This might be one of the reasons that the convergence rate is slightly sporadic. Another reason is that the accuracy depends both on the second-order discretization stencil for the curvature, and on the locally generated level-set function. It is difficult to make a rigorous analysis of the accuracy of the latter, since it depends on several steps as described in the previous section. However, it is easy to see that the accuracy of the latter depends on the alignment of the interface with respect to the grid, and in particular the distance of the interface to the initial grid point $\boldsymbol{x}_{i,j}$. This could explain the varying convergence rate.

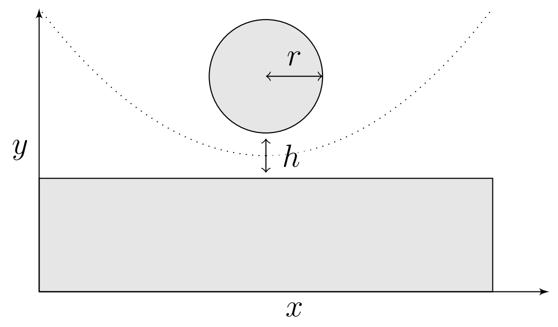


Figure 6: A disc that rests a distance h over a rectangle. The dotted line depicts the kink location.

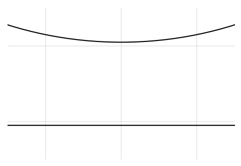


Figure 7: The grid and the rectangle are aligned on the grid such that there is one grid cell between the bodies.

Table 1: Convergence results for the curvature calculated for a disc that rests a distance $h = 1.1\Delta x$ above a rectangle. Results with the central difference on the left, and results with the present method on the right.

n	E_n	order
64	1.035	
128	1.213	-0.23
256	1.310	-0.11
512	1.351	-0.04
1024	1.401	-0.05
2048	1.394	0.01

n	E_n	order
64	$4.172 \cdot 10^{-2}$	
128	$1.123 \cdot 10^{-2}$	1.90
256	$3.950 \cdot 10^{-3}$	1.50
512	$2.583 \cdot 10^{-3}$	0.61
1024	$3.147 \cdot 10^{-4}$	3.00
2048	$1.164 \cdot 10^{-4}$	1.40

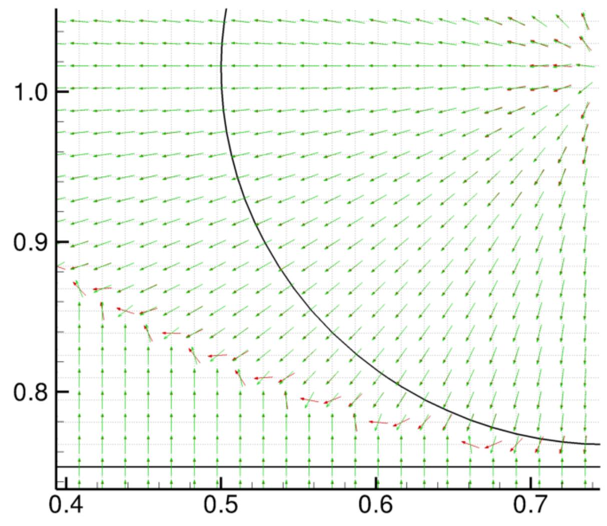


Figure 8: A comparison of the normal vectors as calculated with central differences [in red] and direction differences [in green]. The thick black lines depict the interfaces.

6.2. Disc and rectangle

Again consider the disc and rectangle, see Figure 6. It is of interest to compare the curvature and normal vector calculations, especially in the middle region close to the kink area. Only the level-set function and the geometrical quantities are considered, that is none of the governing equations is solved (equations (1), (2), (10), (11) and (13)). When h is small, the kinks along the dotted line will affect the discretization stencils as has been explained in Section 3.

The following results are obtained with $r = 0.25$ m and $h = \Delta x$. The domain is 1.5 m $\times$ 1.5m, and the straight line is positioned at $y = 0.75$ m. The grid size is 101×101.

Figure 8 shows a comparison of the calculated normal vectors. The results with central differences are depicted with red vectors and the results with direction differences are depicted with green vectors. The figure shows that the central-difference scheme yields much less accurate results for the normal vectors along the kink region than the direction-difference scheme.

Figure 9 shows a comparison of the calculated curvatures between the central-difference scheme, Macklin and Lowengrub's method [1], and the present method. Note that for Macklin and Lowengrub's method [1] the values of the curvature at the grid points are first calculated with the central-difference scheme. Then for the interface locations that need special treatment the curvature values are copied from the interface to the adjacent grid points. This is done in order to compare the results. In all three cases, the curvature is set to zero at grid points that are far from any interface.

The figure shows that the present method yields similar results as Macklin and Lowengrub's method [1], which should be expected. Further, the central-difference scheme leads to large errors in the calculated curvatures in the areas that are close to two interfaces. In particular note that the sign of the curvature becomes wrong. The analytic curvature for this case is $\kappa = -1/r = -4$, and the curvature spikes seen with the central differences is in the order of $|\kappa| \sim \frac{1}{\Delta x} \simeq 67.3$. These spikes will lead to large errors in the pressure jumps through equation (4). The effect of these errors will become more clear in the next case.

6.3. Drop collision in shear flow

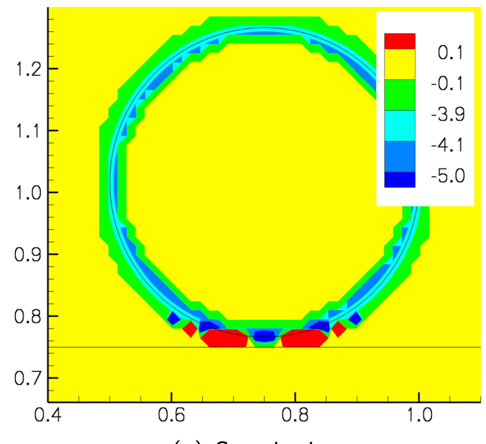
The second case considers drop collision in shear flow. The initial condition is sketched in Figure 10. Both drops have radius R and are initially placed at a distance $d = 5R$ apart in a shear flow. The initial flow velocity changes linearly from the bottom-wall velocity $u_s = -U < 0$ to the top-wall velocity $u_n = U$. The computational domain is $12R \times 8R$, and the grid size is 241×161.

The density and viscosity differences of the two phases are zero, and the shear flow is defined by the Reynolds number and the Capillary number,

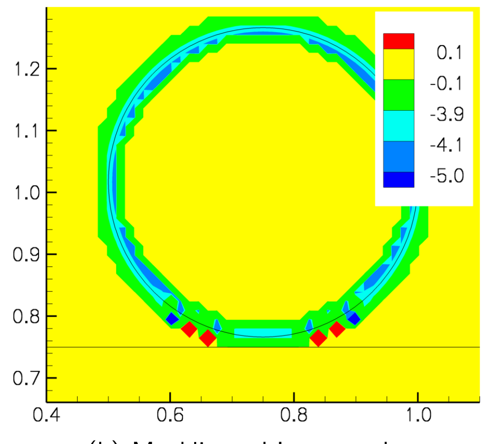
$$Re = \frac{\rho U r}{\mu}, \qquad Ca = \frac{\mu U}{\sigma}. \tag{40}$$

The following results were obtained with $Re = 10$ and $Ca = 0.025$ for $R = 0.5$ m and $h = 0.42$ m. No-slip boundary conditions are used on all walls. The evolution of the flow field and the pressure is simulated by solving the Navier-Stokes equations (1) and (2) as described in Section 3.

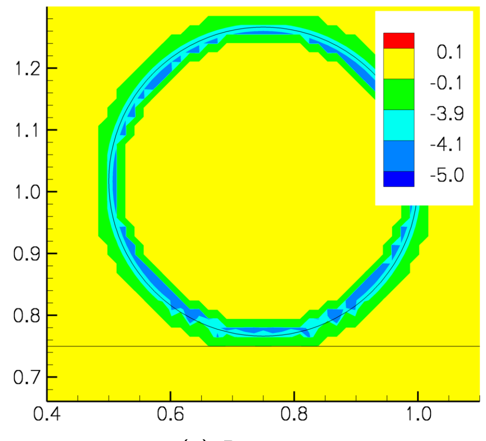
Figure 11 shows the evolution of the interfaces and the velocity field for a simulation where the present method is used. We observe that the drops are deformed before they collide and that the evolution of the drops affects the velocity field. Figure 12 shows a comparison of the interface evolution and the curvature between the central-difference scheme and the present



(a) Standard



(b) Macklin and Lowengrub



(c) Present

Figure 9: A comparison of curvature calculations between the central-difference scheme (Standard), Macklin and Lowengrub's method [1], and the present method. The central-difference scheme leads to large errors in the curvatures in areas that are close to two interfaces.

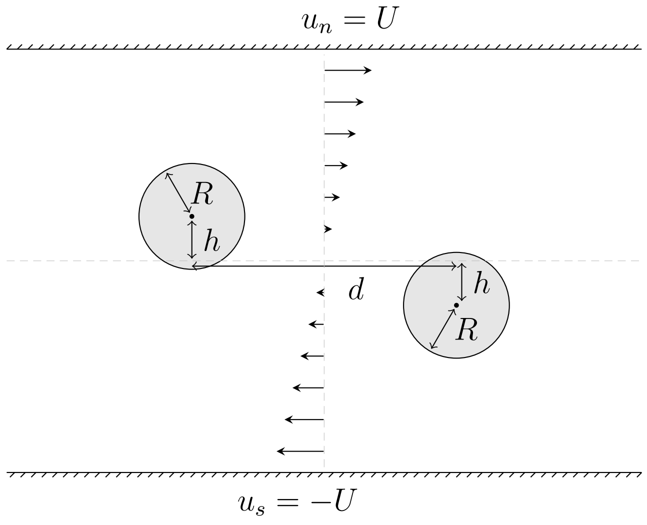


Figure 10: Sketch of the initial condition for the case with two drops in a shear flow.

method. The kinks between the drops again lead to curvature spikes for the central-difference scheme, whereas the improved discretization calculates the curvature along the kink in a much more reliable manner.

The curvature spikes in Figure 12 for the central-difference scheme are seen to prevent coalescence. This is due to the effect they have on the pressure field, cf. equation (16). Figure 13 shows the pressure field at $t = 2.75$ s. It can be seen that the pressure field for the central-difference scheme is distorted in the thin-film region. This distortion in the pressure leads to a flow in the film region which suppresses coalescence. The corresponding result for the present method shows that the pressure is not distorted. It is high in the centre of the thin-film region and lower at the edges. The pressure change induces a flow out of the region which is more as expected.

Finally, Figure 14 shows a comparison of Macklin and Lowengrub's method [1] and the present method. As can be expected, coalescence is observed also with Macklin and Lowengrub's method [1]. However, the time at which coalescence occurs is slightly different, which might be due to small differences in the flow of the thin film region just before coalescence.

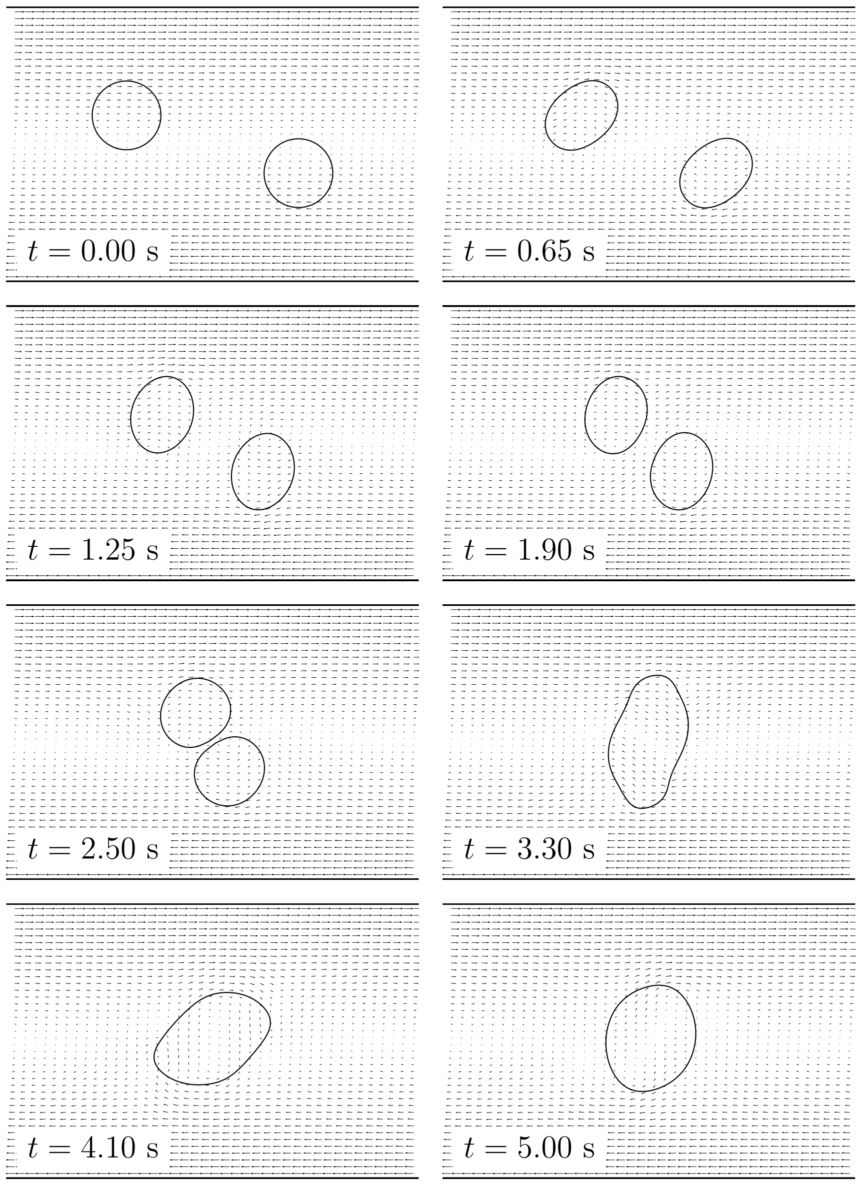


Figure 11: The evolution of the velocity field and the interfaces for drop collision in shear flow.

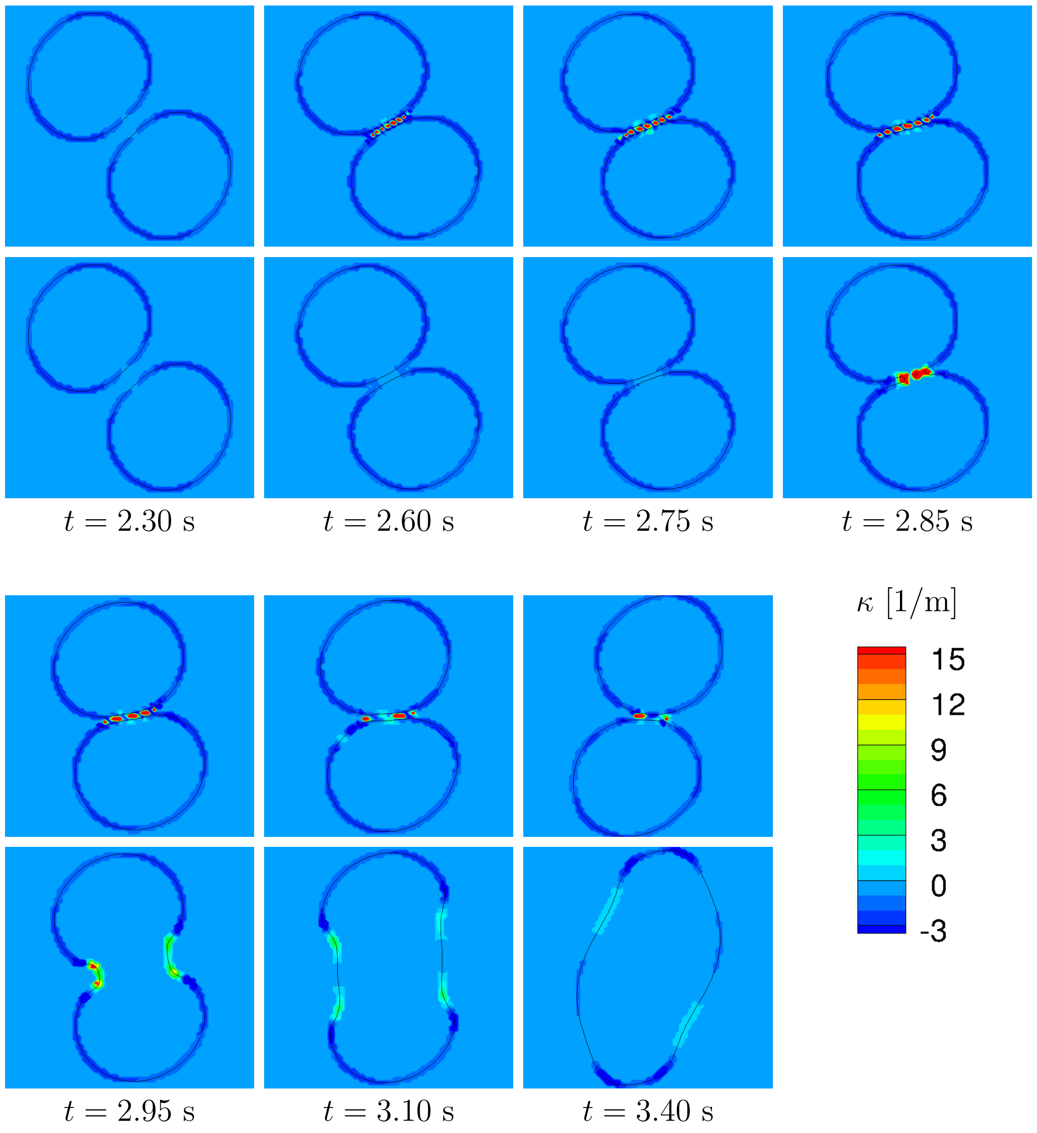


Figure 12: A comparison between the central-difference scheme (top row) and the present discretization scheme (bottom row) of the interface evolution and the curvature κ of drop collision in shear flow.

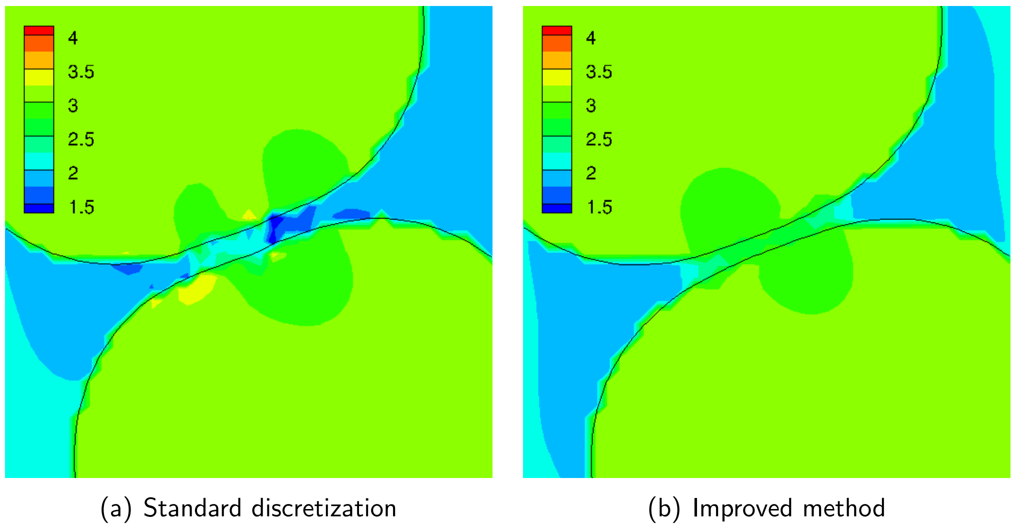


Figure 13: Comparison of the pressure field in the thin film between the drops at $t = 2.75$ s. The contour legends indicate the pressure in Pa.

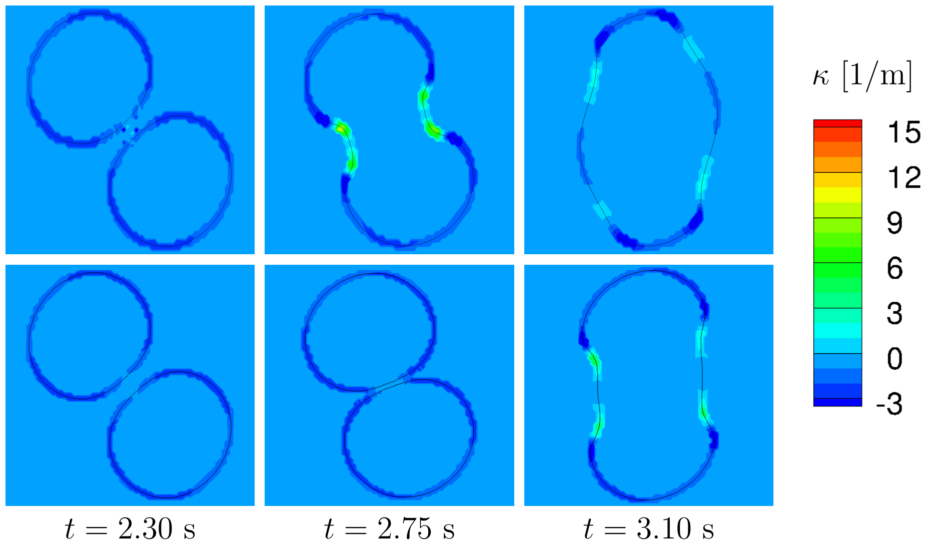


Figure 14: A comparison between Macklin and Lowengrub's method [1] (top row) and the present method (bottom row) of the interface evolution and the curvature κ of drop collision in shear flow.

6.4. Drop collision in axisymmetric flow

The third case considers the collision of two drops of radius R in an axisymmetric flow. The drops are initially placed at a distance $d = R$ apart in a linear flow field

$$\begin{aligned} u(r,z) &= \frac{r}{R} U_0, \\ v(r,z) &= -2\frac{z}{R} U_0. \end{aligned} \tag{41}$$

Here r is the radial coordinate, z is the axial coordinate, and U_0 is a scaling factor of the velocity. Figure 15 shows streamlines of the initial velocity field as well as the initial location of the drops with the centers at $z = \pm 0.75$ m.

The density and viscosity differences of the two phases are zero, and the case is defined by the Reynolds number and the Capillary number,

$$Re = \frac{\rho U_0 R}{\mu}, \qquad Ca = \frac{\mu U_0}{\sigma}. \tag{42}$$

The governing equations (1) and (2) are solved as explained in the previous sections, with some modifications: In axisymmetry the divergence and Laplacian operators become

$$\boldsymbol{\nabla} \cdot \boldsymbol{f} = \frac{1}{r}\frac{\partial}{\partial r}(r f_1) + \frac{\partial f_2}{\partial z}, \tag{43}$$

$$\boldsymbol{\Delta} g = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial g}{\partial r}\right) + \frac{\partial^2 g}{\partial z^2}, \tag{44}$$

where the subscripts indicate vector components, that is $\boldsymbol{f} = (f_1, f_2)$. In addition, one must add $-u/r^2$ to the viscous term in the radial component of the momentum equation, where u is the radial velocity component. Note that equation (43) applies to the calculation of the curvature through equation (15).

The following results were obtained with $Re = 0.5$ and $Ca = 0.025$ for $R = 0.5$ m and $U_0 = 0.5$ m/s. The computational domain was $8R \times 12R$, and the grid size was 120×160. The axis of symmetry coincides with the left boundary. At the other boundaries we specify (u, v) to match equation (41).

The case is run both with the standard discretization of curvature and with the present method. Figure 16 shows the interfaces, the curvature values, and the velocity vectors plotted at various stages of the collision process when the case is run with the standard discretization. The discretization stencil for the curvature starts to cross the kink at some time between

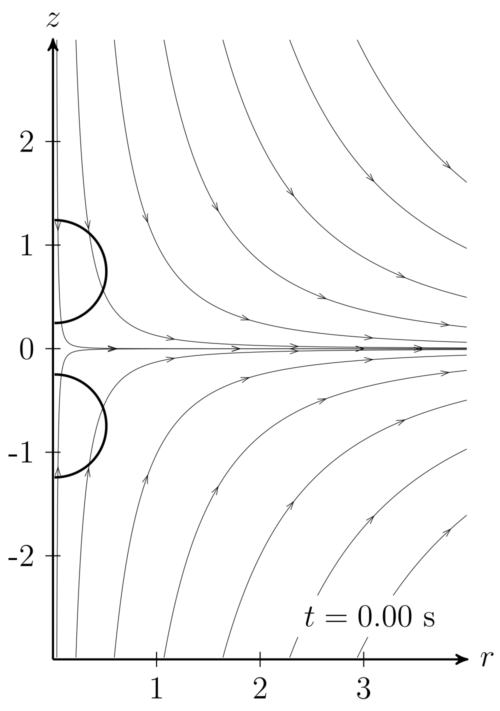


Figure 15: The initial drop interfaces and initial streamlines of the axisymmetric flow.

$t = 0.30$ s and $t = 0.042$ s, after which one can observe a spike in the calculated curvature where both the value and the sign are erroneous. This in turn affects the pressure calculation and leads to a slower coalescence process. Figure 17 shows that the spike in the curvature calculation is prevented with the present method. Note that right after the coalescence at $t = 0.43$ s, the value of the curvature in the thin filament area is and should be very high.

6.5. Normal vectors between two near discs

The final case is designed to show that the direction difference is not always sufficient to calculate the normal vectors. In this case two discs of radius r are placed at a distance h from each other as shown in Figure 18. As in the first case, only the level-set function and the geometrical quantities are considered.

The parameters for this case are $r = 0.25$ m and $h = 1.2\Delta x$. The domain is 1.5 m $\times$ 1.5m and the grid size is 101×101.

As was noted in Section 4.2.2, the curve-fitting discretization scheme may be used as an alternative. In this case the curve-fitting discretization scheme is used at the grid points that are within 1 grid cell from any interface. The direction difference is used at the other grid points.

Figure 19 shows a comparison of the calculated normal vectors. The normal vectors calculated with the direction difference are depicted with red vectors and the normal vectors calculated with a combination of the direction difference and the curve-fitting scheme are depicted with green vectors. The green vectors are on top of the red vectors, which shows that the results are almost identical. But at the centre grid point between the discs, the direction difference is not able to accurately calculate the normal vector. For more complex geometries this error may appear at more than one grid point. The error directly affects both the solutions of the level-set equations, (10), (11) and (13), and the jump conditions for the pressure and the gradient of the velocity, (4) and (5).

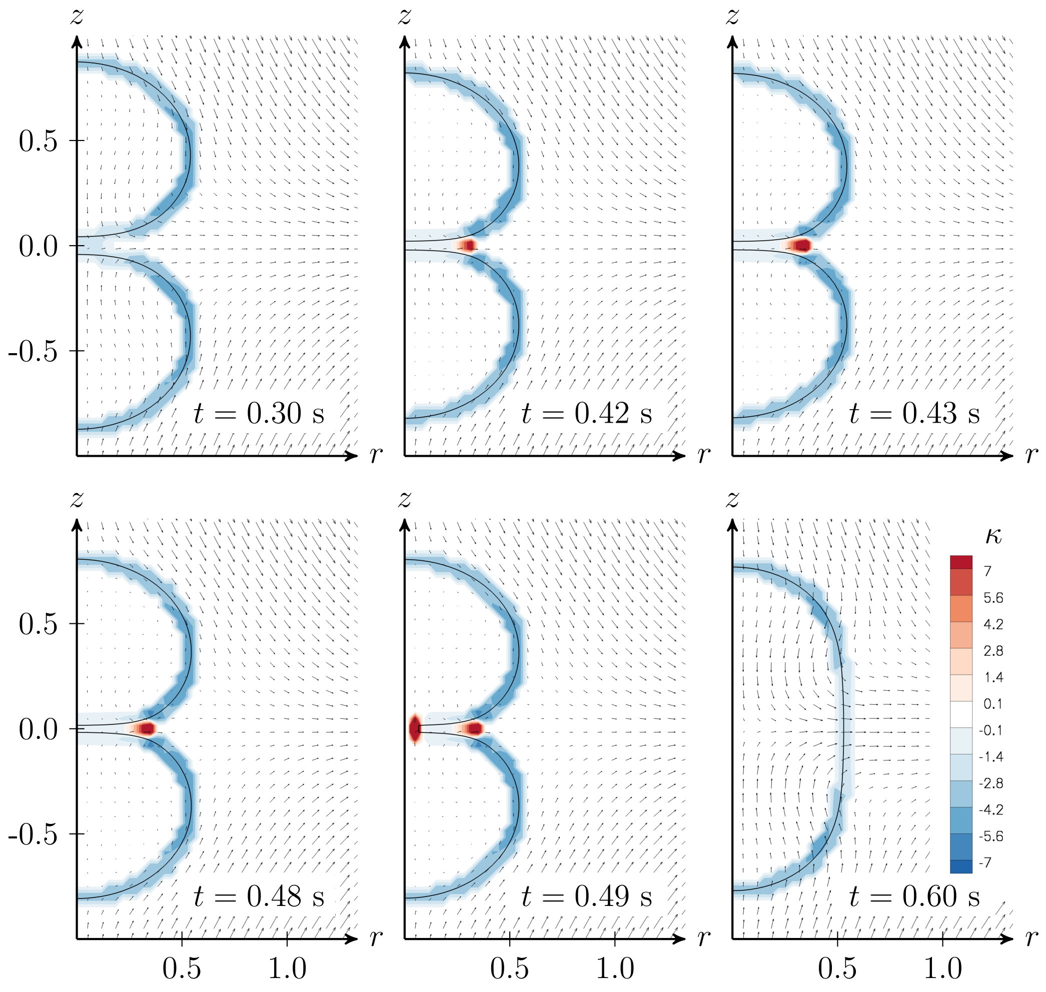


Figure 16: Drop collision in axisymmetric flow calculated with the standard method. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

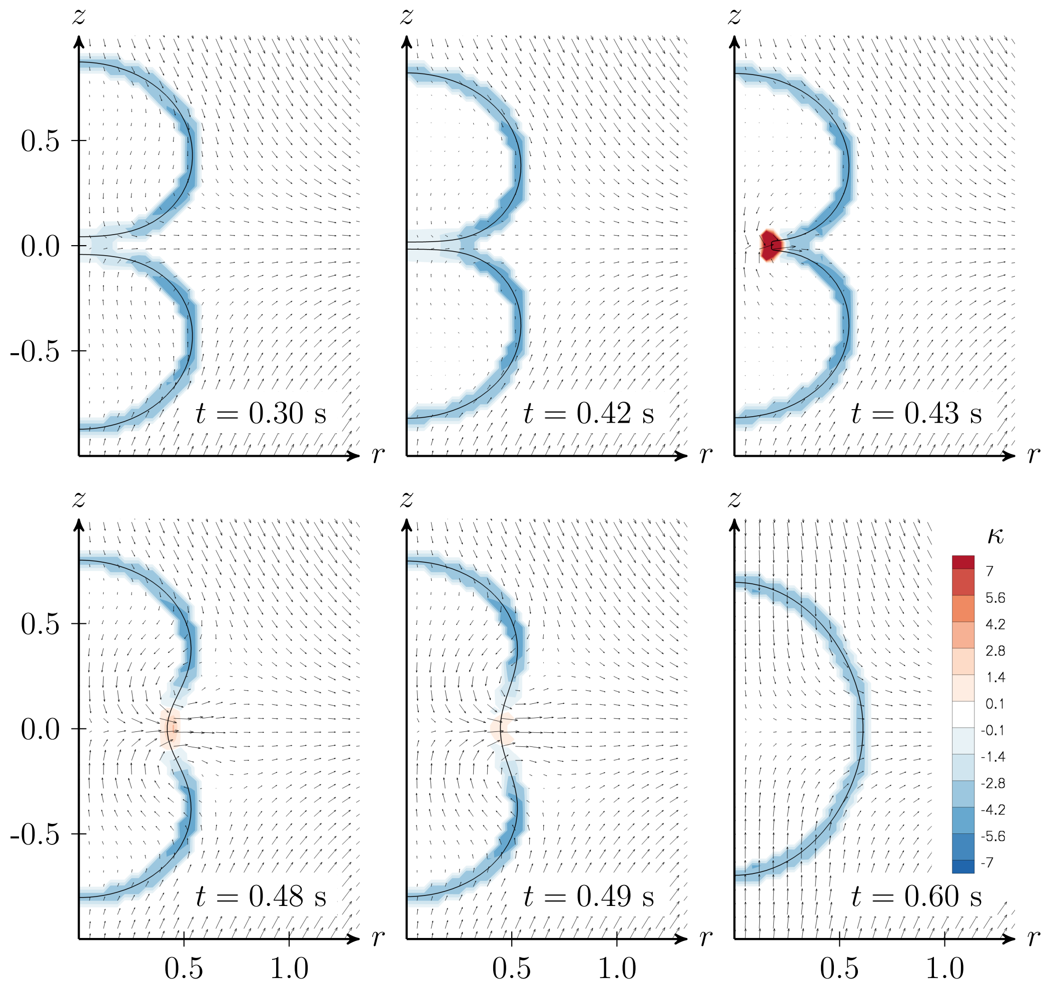


Figure 17: Drop collision in axisymmetric flow calculated with the present method. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

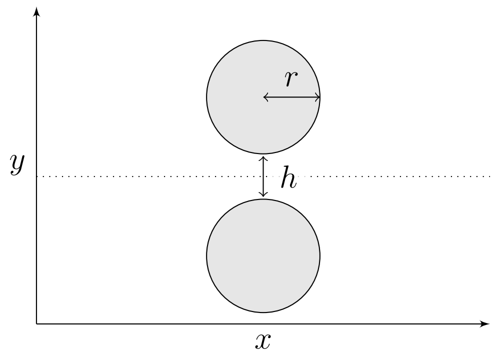


Figure 18: A sketch of the initial state for the two-disc test. The dotted line depicts the kink location.

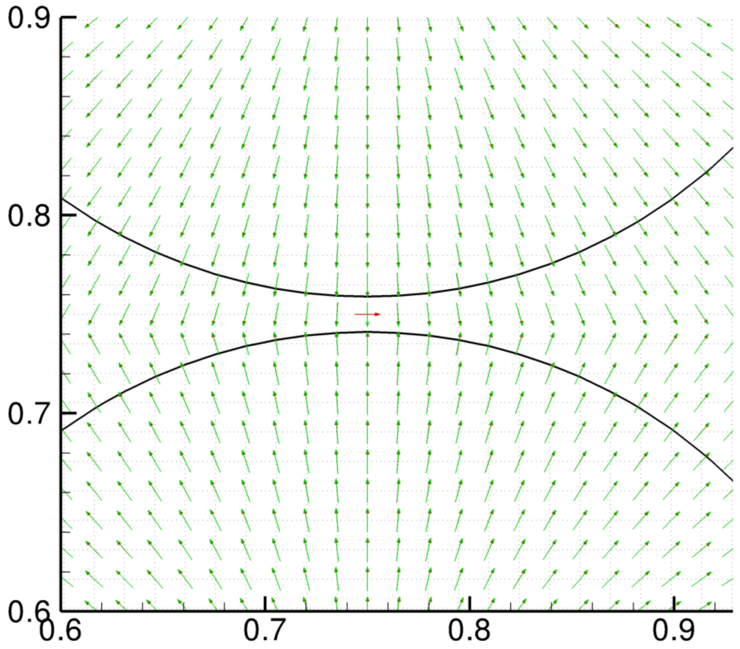


Figure 19: A comparison of the direction difference and the curve-fitting method for calculating normal vectors. The direction difference results are plotted in red below the curve-fitting method results which are depicted in green. The thick black lines depict the interfaces.

7. Conclusions

Improved discretization schemes for the normal vector and the curvature of the interface between two phases have been devised and tested. The curvature was discretized with a curve-fitting discretization scheme based on the geometry-aware discretization presented in [1]. The normal vector was discretized both by the direction difference presented in [2] and a combination of the direction difference and the curve-fitting discretization scheme. The main advantage of the present curvature discretization scheme is that it is independent of the ghost-fluid method. This makes it easier to be adopted into existing level-set codes, for instance codes that use the continuum surface-force method. In addition, it enables the use of models that require curvature values at the grid points, not just on the interface.

The present work has been restricted to two spatial dimensions. An extension to three dimensions would require bicubic parametrization of surfaces, and a local reconstruction of the level-set function based on calculating the minimum distances of parametrized curves to points on the grid. The complexity of this is much higher than for the two-dimensional problem. Note however, that the curve-fitting discretization scheme is directly applicable to axisymmetric cases, which is demonstrated in a test case.

The implementation of the curve-fitting discretization scheme has been described in detail. Our results show that the curvatures calculated with the present scheme converge when the grid size is reduced in a case where the standard scheme fails to converge.

The present discretization scheme is compared with the central-difference scheme in three different cases. The first case is a direct comparison of the schemes for a case with no flow. The second case compares the evolution of two drops colliding in shear flow. Both of these cases demonstrate that the central-difference scheme leads to erroneous behaviour at the kink locations. The second case shows that this behaviour prevents coalescence from occurring due to an erroneous pressure field. The curvature spikes at the kink regions are not observed with the present discretization scheme, and coalescence is achieved for the second case. The present scheme was also compared with Macklin and Lowengrub's method [1], and the results show that the present method gives similar results, as expected. The third case considers the collision of two drops in an axisymmetric flow. As in the previous cases, the central-difference scheme leads to erroneous curvatures at the kink, which is shown to lead to a slower coalescence.

Finally, a fourth test case demonstrates that the direction difference [2] does not always yield accurate results for calculating the normal vector. A combination of the curve-fitting discretization scheme and the direction difference is shown to remove the error in the given case. Accurate calculation of the normal vector is crucial, as it is used both to advect the level-set function (10), extrapolate the velocity vector (11), and to calculate the jumps across the interface (4) and (5). More work should therefore be done to investigate how much this error affects more complex cases.

Acknowledgements

This publication is based on results from the research project Enabling low emission LNG systems, performed under the Petromaks program. The author acknowledges the project partners; Statoil and GDF SUEZ, and the Research Council of Norway (193062/S60) for support.

The authors acknowledge Claudio Walker, Leif Amund Lie and Eirik Svanes for several good discussions.

[1] P. Macklin, J. Lowengrub, An improved geometry-aware curvature discretziation for level set methods: Application to tumor growth, Journal of Computational Physics 215 (2006) 392–401.

[2] P. Macklin, J. Lowengrub, Evolving interfaces via gradients of geometry-dependent interior Poisson problems: Application to tumor growth, Journal of Computational Physics 203 (2005) 191–220.

[3] S. Osher, J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.

[4] J. A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annual Review of Fluid Mechanics 35 (2003) 341–372.

[5] P. Macklin, J. S. Lowengrub, A new ghost cell/level set method for moving boundary problems: Application to tumor growth, Journal of Scientific Computing 35 (2008) 266–299.

[6] D. Salac, W. Lu, A local semi-implicit level-set method for interface motion, Journal of Scientific Computing 35 (2008) 330–349.

[7] Z. Wang, A. Y. Tong, A sharp surface tension modeling method for two-phase incompressible interfacial flows, International Journal for Numerical Methods in Fluids 64 (2010) 709–732.

[8] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, Journal of Computational Physics 227 (2008) 2674–2706.

[9] E. Marchandise, P. Geuzaine, N. Chevaugeon, J.-F. Remacle, A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics, Journal of Computational Physics 225 (2007) 949–974.

[10] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, Journal of Computational Physics 227 (2008) 8395–8416.

[11] K. Y. Lervåg, Calculation of interface curvature with the level-set method, in: Sixth National Conference on Computational Mechanics MekIT'11 (Trondheim, Norway), 23-24 May 2011.

[12] J.-J. Xu, Z. Li, J. Lowengrub, H.-K. Zhao, A level set method for interfacial flows with surfactants, Journal of Computational Physics 212 (2) (2006) 590–616.

[13] K. E. Teigen, K. Y. Lervåg, S. T. Munkejord, Sharp interface simulations of surfactant-covered drops in electric fields, in: Fifth European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2010, Lisbon, Portugal, 2010.

[14] K. E. Teigen, S. T. Munkejord, Influence of surfactant on drop deformation in an electric field, Physics of Fluids 22 (11) (2010) 112104. doi:10.1063/1.3504271.

[15] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (1992) 335–354.

[16] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Scientific Computing 15 (3) (2000) 323–360.

[17] E. B. Hansen, Numerical simulation of droplet dynamics in the presence of an electric field, Doctoral thesis, Norwegian University of Science and Technology, Department of Energy and Process Engineering, Trondheim, iSBN 82-471-7318-2 (Nov. 2005).

[18] H.-K. Zhao, T. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, Journal of Computational Physics 127 (1996) 179–195.

[19] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159.

[20] S. Gottlieb, C. W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review 43 (2001) 89–112.

[21] D. Adalsteinsson, J. A. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118 (1995) 269–277.

[22] X.-D. Liu, R. P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, Journal of Computational Physics 160 (2000) 151–178.

[23] S. Osher, R. P. Fedkiw, The Level-Set Method and Dynamic Implicit Surfaces, Springer, 2003.

[24] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, Computer Graphics 21 (4) (1987) 163–169.

[25] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-spline Techniques, Springer, 2002.

[26] F. N. Fritsch, R. E. Carlson, Monotone piecewise cubic interpolation, SIAM Journal of Numerical Analysis 17 (2) (1980) 238–246.

[27] B. Waerden, E. Artin, E. Noether, Algebra, no. v. 1 in Algebra, Springer-Verlag, 2003.
URL `http://books.google.com/books?id=XDN8yR8R1OUC`

[28] S. Lang, Algebra, Graduate texts in mathematics, Springer, 2002.
URL `http://books.google.com/books?id=Fge-BwqhqIYC`