Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 217 (2006) 364–394

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids

Xiaofeng Yang [a], Ashley J. James [a,*], John Lowengrub [b], Xiaoming Zheng [b], Vittorio Cristini [c]

[a] *Department of Aerospace Engineering and Mechanics, University of Minnesota, 107 Akerman Hall, 110 Union St SE, Minneapolis, MN 55455, USA*

[b] *School of Mathematics, University of California, Irvine, USA*

[c] *Biomedical Engineering, University of California, Irvine, USA*

Received 10 January 2005; received in revised form 4 January 2006; accepted 5 January 2006
Available online 2 March 2006

Abstract

We present an adaptive coupled level-set/volume-of-fluid (ACLSVOF) method for interfacial flow simulations on unstructured triangular grids. At each time step, we evolve both the level set function and the volume fraction. The level set function is evolved by solving the level set advection equation using a discontinuous Galerkin finite element method. The volume fraction advection is performed using a Lagrangian–Eulerian method. The interface is reconstructed based on both the level set and the volume fraction information. In particular, the interface normal vector is calculated from the level set function while the line constant is determined by enforcing mass conservation based on the volume fraction. Different from previous works, we have developed an analytic method for finding the line constant on triangular grids, which makes interface reconstruction efficient and conserves volume of fluid exactly. The level set function is finally reinitialized to the signed distance to the reconstructed interface. Since the level set function is continuous, the normal vector calculation is easy and accurate compared to a classic volume-of-fluid method, while tracking the volume fraction is essential for enforcing mass conservation. The method is also coupled to a finite element based Stokes flow solver. The code validation shows that our method is second order and mass is conserved very accurately. In addition, owing to the adaptive grid algorithm we can resolve complex interface changes and interfaces of high curvature efficiently and accurately.
© 2006 Elsevier Inc. All rights reserved.

Keywords: VOF; Level set; Interface; Unstructured grid

1. Introduction

Flows involving two or more different fluids are very common in many natural and industrial processes, for example, rain drops in the air, free surface flows in the ocean, the dispersion of two immiscible fluids into each

* Corresponding author. Tel.: +1 612 625 6027; fax: +1 612 626 1558.
E-mail address: ajames@aem.umn.edu (A.J. James).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.01.007

other to create emulsions, polymer blending, and so on. Numerical simulations of such flows are difficult because the interface separating different fluid phases must be accurately tracked or captured simultaneously with the flow field evolution. Many methods have been developed for this purpose in the last two decades. Typical methods are the marker particle (MP) method [1], the level set (LS) method [2–4], the volume-of-fluid (VOF) method [5–7], the front tracking method [8,9], and the phase field method [10–14]. Each of these methods has its own advantages and disadvantages, and has been developed into various versions with steady improvements. A detailed introduction and a comparison study of several variants of the MP method, the LS method, and the VOF method were given by Rider and Kothe [15].

In an LS method, the interface is captured by a level set function ϕ, which is zero on an interface, is positive in one fluid and is negative in the other fluid. The interface is thus represented by the zero level sets. The level set function is advected by the velocity $\mathbf{u}$ of the flow field as

$$\phi_t + \mathbf{u} \cdot \nabla\phi = 0. \tag{1}$$

Usually, ϕ is initialized as a signed distance from each grid point to the interface, and it is desirable to maintain ϕ as a signed distance function as the interface evolves since, otherwise, high gradients or even a jump in ϕ can develop, for example, when interfaces merge. The high gradients in ϕ could result in excessive errors when, for example, derivatives of ϕ are taken to find the interface normal vectors. In addition, maintaining ϕ as a distance function is important for providing the interface a width fixed in time [16,17] because in numerical implementations the discontinuous surface delta function in the surface tension force term needs to be smoothed over a finite, but small width to avoid numerical instability, and the measure of the width is based on the level set function, which is assumed to be a signed distance function. However, when ϕ is advected according to Eq. (1), it does not remain a distance function in general. Therefore, a redistancing or reinitialization procedure is needed to modify the advected level set function such that the modified level set function is a signed distance function while the zero level sets remain the same before and after the redistancing, that is, the interface remains unchanged during the redistancing. Notable previous works on redistancing have been done by Sussman and his coworkers [17,18]. Their previous paper [16] also provides a profound understanding of their works. A review of the LS method with an emphasis on applications in fluid interfaces was given by Sethian and Smereka [3]. See also Osher and Fedkiw [4]. The advantage of an LS method is that it handles merging and breaking of the interface automatically, the interface never has to be explicitly reconstructed, and since ϕ is a smooth function, the unit normal $\mathbf{n}$ and curvature κ of the interface can be easily calculated from ϕ as

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad \text{and} \quad \kappa = \nabla \cdot \mathbf{n}. \tag{2}$$

A well-known drawback of the LS method is mass loss/gain, i.e., the mass (or volume for incompressible flow) of the fluid is not conserved. For example, in a drop collision simulation the average mass error with the method described in [18] is 0.5% and the error with the method described in [16] is 1.3%. Even though the zero level sets of ϕ can remain unchanged with some order of accuracy during the redistancing, the volume enclosed by the zero level sets is not conserved as ϕ evolves according to Eq. (1). An open problem is to find more conservative schemes to solve (1) [18].

In the VOF method, a volume fraction function f is defined. The value of the volume fraction in each grid cell is equal to the ratio of the volume of one of the fluids in this cell, called fluid 1, to the total volume of the grid cell. Thus, f is unity in a cell that lies completely in fluid 1, and is zero if the cell lies completely in the other fluid, called fluid 2. For cells that include an interface (called the interfacial cells), and thus contain both fluid 1 and fluid 2, f is between zero and unity. Inversely, given the volume fraction in each grid cell, one can reconstruct an approximate interface, which is called "interface reconstruction". The f field is advected by the flow field, that is

$$f_t + \mathbf{u} \cdot \nabla f = 0. \tag{3}$$

However, meaningful solution of Eq. (3) is not easy. Because f is a discontinuous function, using standard numerical schemes such as an upwind finite difference method to solve Eq. (3) can easily diffuse the interface, which should, however, remain sharp. One way to overcome this problem is to advect f based on a reconstructed interface determined by the f field. Interface reconstruction is thus a key part of any VOF method, and has been improved from the original simple line (piecewise constant) interface reconstruction (SLIC) to piecewise linear interface reconstruction (PLIC). A detailed review of some SLIC and PLIC interface reconstruction methods from a historical perspective was given by Rider and Kothe [6]. Piecewise circle [19], piecewise parabolic [20,21], and piecewise spline [22] interface reconstructions have also been developed. Two new interface reconstruction methods based on a least squares fit were presented by Scardovelli and Zaleski [23]. Based on the reconstructed interface, several schemes have been developed to advect f. Most of these works are based on Eulerian advection of f on structured grids, either with a split scheme or an unsplit scheme. These kind of schemes are described by Rider and Kothe [6]. However, the implementation of these schemes on triangular unstructured grids is difficult. In particular, it is very complicated to compute the fluid volume fluxes for triangular cells. A promising method is the Lagrangian–Eulerian advection method, which consists of three typical stages: a Lagrangian projection, an interface reconstruction, and a remapping. This method is independent of the grid type. It is suitable for both rectangular structured grids and triangular unstructured grids. Typical works on this method are Shahbazi et al. [24] and Ashgriz et al. [25]. A new mixed split Eulerian implicit—Lagrangian explicit advection scheme was presented by Scardovelli and Zaleski [23].

Compared to the LS method, a notable merit of the VOF method is that it can conserve mass very accurately. A drawback of the VOF method is that it is hard to compute accurate interface normals and curvatures because of the discontinuity in f. Several algorithms have been developed for normal vector calculations. However, most of them are less than second order accurate. A detailed comparison of several normal calculation methods was given by Rider and Kothe [6]. Calculations of curvature are even more difficult because they involve taking second order derivatives of f. Usually, the f field is smoothed by some kernel function when used for curvature calculations. However, the optimum type and extent of the kernel operation may very well depend on the interfacial flow being modeled [26]. A detailed comparison of several kernels and some principles on how to choose a kernel function were given by Williams et al. [27].

Recently, methods that couple two different schemes have been developed. Examples are the coupled level set and volume-of-fluid method (CLSVOF) [28,29], the hybrid particle level set method [30], and the mixed markers and volume-of-fluid method [31]. A coupled method takes advantage of the strengths of each of the two methods, and is therefore superior to either method alone. In a CLSVOF method, both ϕ and f are evolved according to Eqs. (1) and (3), respectively. ϕ is used to calculate the unit normal and curvature of the interface. f, together with the unit normal calculated from ϕ, is used to reconstruct a new interface. Thus, ϕ can finally be reset to a signed distance function based on the reconstructed interface. Volume can be conserved very accurately by tracking f. Since ϕ is continuous, the calculation of the interface normal and curvature is natural, easy and accurate, in contrast to the normal and curvature calculation in a pure VOF method, which requires unphysical smoothing of f. However, most of the previous works, including the CLSVOF method by Sussman and Puckett [28], are for structured grids. A few works have been done on unstructured grids [24,25,32,19]. Very recently, Sussman et al. [33,34] have developed a CLSVOF method for an adaptive Cartesian grid.

The goal of the present paper is to present an adaptive coupled level set and volume-of-fluid (ACLSVOF) volume tracking method for unstructured triangular grids. Using adaptive unstructured grids, we can cluster the grid near the interface, enabling us to efficiently and accurately resolve complex changes in interface topology, regions of high interface curvature, and the near contact regions between two impacting interfaces, which is in general hard to achieve on structured grids. In our work, we use the adaptive mesh algorithm developed in [35,36], which is based on the earlier work of Cristini et al. [37]. For the advection of the level set function, we solve Eq. (1) using a discontinuous Galerkin finite element method. For the evolution of the volume fraction, we use a Lagrangian–Eulerian advection scheme. Different from the works by Shahbazi et al. [24] and Ashgriz et al. [25], our algorithm for volume fraction advection is built in the framework of our ACLSVOF method, where the normal and the curvature of the interface will be calculated from the continuous level set function. For interface reconstruction, different from the previous works, we have developed an analytic piecewise linear interface reconstruction for triangular grids [38]. The method is also coupled to a finite element based Stokes solver. We use the Stokes solver for illustration. The method can be generalized to Navier–Stokes and viscoelastic flows. Solid body translation, rotation, Zalesak disk rotation and a single vortex flow problem are simulated to test our ACLSVOF volume tracking method. Drop deformation in an extensional flow is simulated with the volume tracking method coupled to the Stokes solver.

The rest of the paper is organized as follows. In Section 2 we describe the numerical methods for the level set function evolution, the volume fraction evolution, grid adaption, and the Stokes flow solver. Code validation and results are presented in Section 3. A conclusion is given in Section 4.

2. Computational methods

2.1. The Eulerian evolution of the level set function

The evolution of the level set function is governed by Eq. (1). In practice, we reformulate Eq. (1) as

$$\phi_t + \nabla \cdot (\mathbf{u}\phi) = \phi \nabla \cdot \mathbf{u}. \tag{4}$$

Theoretically, we have $\nabla \cdot \mathbf{u} = 0$ for incompressible flow. Numerically, however, this is not true in general. So, the right hand side is retained in Eq. (4) such that the level set function is only convected by the velocity field even when the numerical velocity field is not exactly divergence free.

Eq. (4) is solved using a Runge–Kutta discontinuous Galerkin finite element method (RKDG) [39,40] on a fixed Eulerian grid (in contrast to the projected Lagrangian grid used in the next section). See also [36]. Let $P^1(T)$ be the set of all polynomials of degree at most 1 on the triangular element T, and $\mathscr{T}_h$ a triangulation of the computational domain Ω with a characteristic grid size h. The finite element space is defined as

$$W_h = \{v \in L_2(\Omega) : v|_T \in P^1(T),\ \forall T \in \mathscr{T}_h\}. \tag{5}$$

Then, our numerical method is to find an approximate solution $\phi_h \in W_h$, such that in each element $T \in \mathscr{T}_h$

$$\int_T \frac{\partial \phi_h}{\partial t} v - \int_T \phi_h \mathbf{u}_h \cdot \nabla v + \sum_{e \in \partial T} \int_e \mathbf{u}_h \cdot \hat{\mathbf{n}} \hat{\phi}_h v^- = \int_T \nabla \cdot \mathbf{u} \phi_h v, \quad \forall v \in P^1(T), \tag{6}$$

where $\mathbf{u}_h$ is the discrete velocity, v^- denotes the value of a test function taken from within element T, $\hat{\phi}_h$ is a single-valued flux through an element boundary e, and $\hat{\mathbf{n}}$ is the outward unit normal of T. Note that $\mathbf{u}_h$ is continuous across any element boundary (we employ a continuous velocity space; see Section 2.4.), but ϕ_h and the test function v are discontinuous across an element boundary. Specifically, $\hat{\phi}_h$ is defined as,

$$\hat{\phi}_h = \begin{cases} \phi_h^- & \text{if } \mathbf{u}_h \cdot \hat{\mathbf{n}} \geqslant 0, \\ \phi_h^+ & \text{if } \mathbf{u}_h \cdot \hat{\mathbf{n}} < 0, \end{cases} \tag{7}$$

where ϕ_h^- is the value of ϕ_h on boundary e taken from within element T, and ϕ_h^+ is the value of ϕ_h on boundary e taken from outside element T.

The time integration is performed using a second order Runge–Kutta scheme. For efficiency, the narrow band/local level set strategy is utilized so that Eq. (6) is solved only in the neighborhood of the interface (see also [36]).

Since the numerical level set function ϕ_h obtained from the above discontinuous Galerkin method is discontinuous, we project (in the L_2 sense) the discontinuous ϕ_h onto a continuous piecewise linear space before we use it to calculate the interface normal vector. Specifically, we define the continuous piecewise linear space as

$$W_h^* = \{v \in C^0(\Omega) \cap L_2(\Omega) : v|_T \in P^1(T),\ \forall T \in \mathscr{T}_h\}. \tag{8}$$

Let ϕ_h^* denote the continuous level set function obtained by the projection. Then, the L_2 projection of $\phi_h \in W_h$ to $\phi_h^* \in W_h^*$ is simply the solution of the following weak formulation:

$$\int_\Omega \phi_h^* v = \int_\Omega \phi_h v, \quad \forall v \in W_h^*. \tag{9}$$

In the framework of our ACLSVOF method, this continuous level set function will be used for interface normal calculations during interface reconstruction, and for convenience the superscript $*$ will be omitted from now on. At the end of each time step, the level set function will be reinitialized to a signed distance function based on the reconstructed interface.

2.2. The Lagrangian–Eulerian volume fraction evolution

For the volume fraction advection, we employ a Lagrangian–Eulerian advection scheme, which consists of three stages: a Lagrangian projection stage, a reconstruction stage, and a remapping stage. Similar works can be found in [24,25]. Fig. 1 shows a schematic of the method. During the Lagrangian projection stage, we treat each grid cell as a material element. All fluid elements (or equivalently the grid cells) are advected/projected using a Lagrangian approach. For example, in Fig. 1, by projecting fluid element ABC we get the corresponding projected Lagrangian element $A'B'C'$. Because the volume fraction is solely transported with the flow, the volume fraction remains the same during the projection. Notice that for a linear velocity field triangles remain triangles, no matter how they deform. All projected elements form a new grid, which we call the Lagrangian grid. In contrast, the original grid is called the Eulerian grid.

As the shape of each element deforms, the interface position in each interfacial cell also changes. During the reconstruction stage, we reconstruct the interface on the Lagrangian grid. The interface is approximated using piecewise linear segments based on both the level set field and the volume fraction field. The reconstructed interface concretely defines the configuration of each fluid phase on the Lagrangian mesh, as illustrated in Fig. 1, where the two fluids in cell $A'B'C'$ are separated by a reconstructed interface segment $E'F'$. $A'B'E'F'$ is full of fluid 1, and $C'E'F'$ is full of fluid 2. Based on this information, we map each fluid element on the Lagrangian mesh back to the original Eulerian mesh (or an adapted mesh when the grid is remeshed) during the remapping stage, which gives us the advected volume fraction on the Eulerian grid and completes the volume fraction advection. Details of the method are as follows.

2.2.1. Lagrangian volume fraction projection

The projection of the volume fraction is equivalent to solving the volume fraction advection equation (3) using a Lagrangian approach. Let us consider each grid cell a material element, and let f_T^n be the volume fraction in element T at time t^n, then we have from Eq. (3) discretely that

$$\tilde{f}_T^{n+1} = f_T^n, \quad \forall T \in \mathcal{T}_h, \tag{10}$$

where $\tilde{f}_T^{n+1}$ denotes the volume fraction in element T after a Lagrangian advection. Eq. (10) is equivalent to saying that the volume fraction in each element is solely transported with the flow, which is consistent with the differential Eq. (3).

In the meantime, each material element transports with the flow. So, we have that

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}, \tag{11}$$

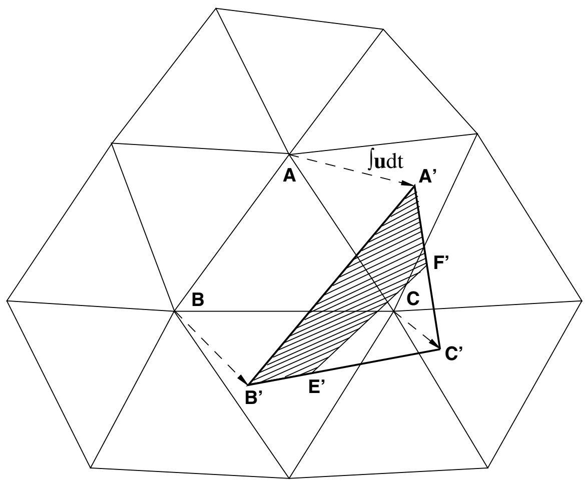


Fig. 1. Schematic of the Lagrangian–Eulerian volume fraction evolution.

where $\mathbf{x}$ is the location of a material point on the boundary of an element. Numerically, if we assume that the velocity is piecewise linear over every finite element, then finding the Lagrangian position of each element $T \in \mathscr{T}_h$ is equivalent to finding the Lagrangian position of each vertex of T. This can be achieved by solving Eq. (11) at each vertex of the grid. These projected elements form a new projected grid that we call the Lagrangian grid. Using a second order Runge–Kutta (RK) scheme, we can obtain the coordinates of each grid node of the Lagrangian grid as the following

$$\tilde{\mathbf{x}}_i^{n+\frac{1}{2}} = \mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{u}(\mathbf{x}_i^n, t^n), \quad \text{for } i = 1, \ldots, MT, \tag{12}$$

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t\mathbf{u}(\tilde{\mathbf{x}}_i^{n+1/2}, t^{n+\frac{1}{2}}), \quad \text{for } i = 1, \ldots, MT, \tag{13}$$

where MT is the total number of grid nodes, and $\mathbf{u}(\tilde{\mathbf{x}}_i^{n+1/2}, t^{n+\frac{1}{2}})$ on the Lagrangian mesh has been interpolated from the Eulerian mesh.

In the first step of the RK integration a triangular Eulerian grid cell is projected using the linear velocity field in that cell, so it remains a triangle, although it may deform. To perform the second step of the RK integration, the velocity is updated on the Eulerian mesh, then interpolation is used to determine the velocity at the vertices of the Lagrangian grid cells. It is assumed that these velocities define a linear velocity field in each Lagrangian cell, so that in this second RK step we are again projecting a triangle with a linear velocity field, so it remains a triangle. If the velocity field on the Eulerian mesh were used directly then cell edges would be projected with a velocity field that is only *piecewise* linear, so the projected cell edges would be *piecewise* linear.

Theoretically, the above Lagrangian projection conserves the volume of fluid in each element, because the volume of each element does not change due to the incompressibility of the fluid. Numerically, however, the volume of fluid can not be *exactly* conserved, because: (i) the numerical velocity field delivered from a flow solver is in general not *exactly* divergence free; (ii) the numerical velocity field is *approximated* by piecewise linear basis functions, and a triangular element remains a triangle during a projection only for linear velocities; (iii) in the second RK step the velocity on the Lagrangian mesh is interpolated from the Eulerian mesh; and (iv) the numerical integration of Eq. (11) is, of course, not exact. In other words, because of these four reasons, the volume of a material element (or grid cell) is not exactly constant during the above numerical Lagrangian projection, and thus, the volume of each single fluid phase may not be conserved exactly during the projection.

However, as we will show in Section 3, the volume loss/gain due to the last three reasons is extremely small. The magnitude of the volume loss/gain due to velocity divergence depends on the quality of the flow solver. In addition, we emphasize that the volume loss/gain can be significantly decreased by refining the grid. This will be seen by comparing the mass error on grids of different resolutions (see results in Section 3).

2.2.2. *Interface reconstruction*

After the Lagrangian projection, we know the advected volume fractions in the Lagrangian cells. To map the volume fraction from the Lagrangian mesh back to the Eulerian mesh, we need to know explicitly on the Lagrangian grid where fluid 1 is and where fluid 2 is, or equivalently, where the interface is. In most previous works, the interface in a cell is reconstructed as a line segment of the form $\mathbf{n} \cdot \mathbf{x} = \alpha$, where $\mathbf{n}$ is the unit normal vector of the interface in this cell, $\mathbf{x}$ is the location of a point on the interface, and α is the line constant. In a classic VOF method, $\mathbf{n}$ is calculated as the gradient of the volume fraction. Since volume fraction is discontinuous across interfaces, most of these methods are less than second order accurate. Normal vector calculations based on least squares minimization are second order [6]. However, this method can be "prohibitively expensive, especially in 3D" [6]. The parameter α is obtained by enforcing volume conservation, and is usually calculated iteratively using a numerical method such as Brent's method [6]. During each iterative step, the volume truncated by the line with the most recent estimate of α is calculated and compared to the given volume of fluid in the cell. A final α is declared when the discrepancy between these two volumes is within some prescribed tolerance.

In the present work, we use a different form to represent the linear interface, in which each linear interface segment is denoted by its two end points. Thus, the interface reconstruction algorithm must find the coordinates of the two end points for each interface segment. To do this, we first compute the normal vector $\mathbf{n}$ of the interface in each interfacial cell. Then, the coordinates of the two end points of each interface segment are finally determined by requiring that the normal vector of the interface is $\mathbf{n}$ and that the interface truncates the cell with the given volume fraction. Different from many previous works, we have developed an analytic method, instead of an iterative method, for the second part (see also [38]). The method is analytic, efficient, and conserves volume of fluid exactly. For brevity, the details of this method are given in Appendix A. In the following, we describe in detail how we calculate the interface normal vector.

In the framework of our ACLSVOF method, interface normal vectors are calculated from the gradient of the *continuous* level set function via Eq. (2). The level set function associated with the Lagrangian mesh is obtained by interpolating the level set function associated with the Eulerian mesh onto the Lagrangian mesh. To calculate the normal vector in a cell, we fit a quadratic function for ϕ using a least squares method over a stencil that consists of the vertices of the cell and of all the cells that share at least a vertex with the cell, as shown in Fig. 2. To simplify the calculation, a local Cartesian coordinate system $x' - y'$ is defined for each normal calculation (see Fig. 2). The origin of the local coordinate system is located at the center of the cell under consideration, at which the normal is located. The local coordinate axes x' and y' are parallel to the global coordinate axes x and y, respectively. Let (x_c, y_c) be the global coordinates of the center of the cell. Then, the local and global coordinates of a point are related to each other via $x' = x - x_c$ and $y' = y - y_c$.

In each local coordinate system, the quadratic function for ϕ has the generic form

$$\phi = ax'^2 + bx'y' + cy'^2 + dx' + ey' + g, \tag{14}$$

where a, b, c, d, e, and g are coefficients to be determined using the least squares method, that is, a, b, c, d, e, and g are the least squares solution of the over-constrained linear system

$$Q_s = r, \tag{15}$$

where

$$Q = \begin{pmatrix} x_1'^2 & x_1'y_1' & y_1'^2 & x_1' & y_1' & 1 \\ x_2'^2 & x_2'y_2' & y_2'^2 & x_2' & y_2' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N'^2 & x_N'y_N' & y_N'^2 & x_N' & y_N' & 1 \end{pmatrix}, \quad s = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ g \end{pmatrix}, \quad r = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{pmatrix},$$

(x_i', y_i'), $i = 1, \ldots, N$, are the local coordinates of the ith node in the stencil, N is the total number of nodes in the stencil, and ϕ_i, $i = 1, \ldots, N$, is the level set value at the ith node. The least squares solution of this linear system is $s = (Q^tQ)^{-1}Q^tr$, where the superscript t denotes matrix transpose and -1 denotes matrix inversion. Since

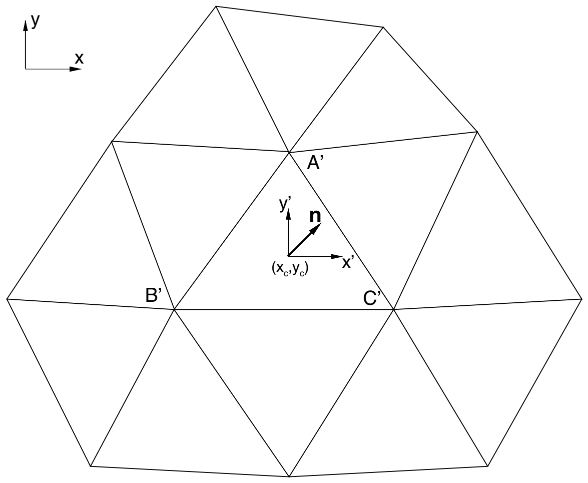


Fig. 2. The stencil for calculating the interface normal vector in element $A'B'C'$. The normal vector and the origin of the local coordinate system $x' - y'$ are located at the center of element $A'B'C'$.

Q^tQ is symmetric positive definite, it can be efficiently inverted using Cholesky decomposition. Once the coefficients are known, the unit normal vector, which is located at the origin of the local coordinate system, is simply

$$\mathbf{n} = \left(\frac{d}{\sqrt{d^2+e^2}}, \frac{e}{\sqrt{d^2+e^2}}\right).$$

Once the normal vector is known the endpoints of the interface segment are computed analytically as described in Appendix A.

2.2.3. Volume fraction remapping

By reconstructing the interface on the Lagrangian grid, we know exactly where fluid 1 is and where fluid 2 is. Then, the new advected volume fraction f_T^{n+1} on the original Eulerian grid (or an adapted grid when the grid is adaptive) is obtained by mapping each projected Lagrangian fluid element back to the Eulerian grid. More specifically, to find the new volume fraction in an Eulerian cell, we first find the volume of fluid 1 that overlaps or maps into this cell. This is achieved by performing polygon-polygon clippings, for which fluid elements and grid cells are treated as polygons, and each Lagrangian fluid region that is occupied by fluid 1 is clipped against each Eulerian grid cell. For example, as illustrated in Fig. 1, the two fluids in the Lagrangian cell $A'B'C'$ are separated by the interface segment $E'F'$. Fluid 1 occupies the entire volume $A'B'E'F'$. By clipping $A'B'E'F'$ against cell ABC, we can find the portion of fluid 1 in $A'B'E'F'$ that maps into the Eulerian cell ABC. Notice that the rest of fluid 1 in $A'B'E'F'$ is mapped into the adjacent cells of ABC. The clipping algorithm (see Appendix B for details) renders the polygonal intersection between each fluid region and each Eulerian grid cell. In particular, it returns the vertex coordinates, (x_i,y_i), $i = 0,\dots,m$, of the intersection polygon, where we assume that $x_m = x_0$ and $y_m = y_0$. The area of the polygon is computed as [41]

$$S = \frac{1}{2}\left|\sum_{i=0}^{m-1}(x_iy_{i+1} - x_{i+1}y_i)\right|. \tag{16}$$

The total volume of fluid 1 that maps into an Eulerian cell is the sum of the intersecting areas that the cell has with all the Lagrangian regions that are occupied by fluid 1. The new volume fraction in an Eulerian cell is then the total volume of fluid 1 that maps into this cell divided by the volume of the cell.

In practice, we only need to clip a grid cell against the fluid elements which are in the vicinity of the cell. Because the interface only moves a finite distance of order h in each time step, only volume fractions in cells near the interface change and need to be determined by remapping. Volume fractions in cells far away from the interface are either 1 or 0 and they can be determined simply by inspecting the level set function without remapping. These strategies can save computation time dramatically and are implemented easily for both fixed and adaptive grids. Also note that since the projected fluid elements are connected consecutively without any gap or overlaps between them, the new volume fraction on the Eulerian mesh must be between 0 and 1, and the total volume of fluid is conserved exactly during remapping.

Polygon clipping is a very fundamental element in graphics. Various clipping algorithms, for example the Sutherland–Hodgeman algorithm [42] and the Weiler–Atherton algorithm [43], have been designed. We use the Sutherland–Hodgeman algorithm for its simplicity. The basic idea of this algorithm is described in Appendix B for completeness.

2.3. The adaptive triangular mesh algorithm

To let grid refinement follow the interface motion, we adapt/rezone the grid at the end of each time step. We use the adaptive remeshing algorithm developed in [35,36], which is an adaptation to flat domains of the adaptive surface triangulated mesh of Cristini et al. [37]. The basic ideas of the method are described as follows. We refer readers to [35–37] for further details.

The key part of the algorithm is based on the minimization of the mesh configurational energy, which is defined as

$$E = \frac{1}{2}\sum_{\text{edges}} \gamma^2, \tag{17}$$

where the edge tensions are $\gamma = l - l_0$, l is the local edge length, and l_0 is an optimal edge length. The optimal edge length is determined from the relevant local length scale of a specific problem, for example, the distance of a local point to the nearest interface, the radius of curvature, and the distance between two interfaces. The mesh energy is an integral measure of how local edge lengths deviate from an optimal value. Minimization of the mesh energy (17) implies that $l \approx l_0$, i.e., the optimal mesh is achieved by the minimization of mesh energy.

By defining the mesh energy (17), the mesh is analogous to a system of springs. For a fixed mesh topology, the minimization of the mesh energy (17) is thus equivalent to the relaxation of a dynamical system of massless springs with tensions γ and damping coefficients of unity. Accordingly, the position $\mathbf{x}$ of a node evolves with a velocity $\dot{\mathbf{x}}$ proportional to the vector sum of spring forces acting on the node. Therefore,

$$\dot{\mathbf{x}} = \sum_{j=1}^{N_c} \hat{\mathbf{e}}_j \gamma_j, \tag{18}$$

where N_c is the coordination number of the node, and $\hat{\mathbf{e}}_j$ is the unit vector parallel to edge j.

Node displacements reduce the mesh energy and lead to equilibrium of tensions at all nodes. However, due to topological constraints the tensions themselves are in general nonzero. To further reduce the mesh energy, it is necessary to perform topological mesh restructuring operations, which include node addition, subtraction, and reconnection. Node addition/subtraction is needed when grids need to be refined/coarsened in some local region. Node reconnection is used to reconnect some of the nodes such that the elements formed by these nodes become more regular, or not extremely stretched. For specific details, see [35–37]. Topological restructuring operations provide access to mesh configurations with lower energy than accessible by mesh relaxation. The relaxation and topological restructuring are performed iteratively until $|\dot{\mathbf{x}}|$ is less than a prescribed value and the criteria for node addition, subtraction and reconnection are no longer satisfied.

The resulting mesh maintains the resolution of the relevant local length scale everywhere with prescribed accuracy. The minimization depends only on the instantaneous configuration of the interface, and is insensitive to the deformation history.

2.4. The Stokes flow solver

To simulate a real problem, for example drop deformation in an extensional flow, we coupled our interface capturing method with a finite element based Stokes flow solver. The governing equations for the flow solver are

$$-\nabla \cdot (\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^t)) + \nabla p = \nabla \cdot F_s, \tag{19}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{20}$$

with Dirichlet (for prescribed boundary flows, e.g., shear and no-slip flows) and/or Neumann-type (free-slip) boundary conditions (see [36]), where μ is the dynamic viscosity of the local fluid, p is the pressure, $F_s = \sigma\delta_\Sigma(I - \mathbf{n} \otimes \mathbf{n})$ is the capillary pressure tensor, σ is the surface tension coefficient, $\delta_\Sigma = \delta(\phi)|\nabla\phi|$ is the surface delta function, and I is the identity matrix. The local dynamic viscosity μ is equal to μ_1 in fluid 1, μ_2 in fluid 2. Notice that since we use the continuum surface stress formulation, the curvature of the interface does not explicitly appear in Eq. (19).

The Stokes equations are solved using the Mini finite element method [44]. The finite element spaces are

$$\begin{aligned} V_h = {} & \{\mathbf{v} \in (H^1(\Omega))^2 : \mathbf{v}|_T \in (P^1(T))^2, \forall T \in \mathscr{T}_h, \mathbf{v}|_{\partial\Omega_D} = 0, \mathbf{v}\cdot\bar{\mathbf{n}}|_{\partial\Omega_N} = 0\} \\ & \oplus \{\mathbf{v} : \mathbf{v}|_T \in (P^3(T))^2 \cap (H^1_0(T))^2, \forall T \in \mathscr{T}_h\}, \end{aligned} \tag{21}$$

$$Q_h = \left\{ q \in C^0(\Omega) \cap L^2(\Omega) : q|_T \in P^1(T), \forall T \in \mathscr{T}_h, \int_\Omega q = 0 \right\} \tag{22}$$

for velocity and pressure, respectively, where $\partial\Omega_D$ denotes Dirichlet boundary, $\partial\Omega_N$ denotes Neumann-type boundary, and $\bar{\mathbf{n}}$ is the normal to $\partial\Omega_N$.

Multiplying Eq. (19) by $\mathbf{v}_h \in V_h$ and Eq. (20) by $q_h \in Q_h$, we get the weak formulation for the unknowns $(\mathbf{u}_h, p_h) \in V_h \times Q_h$:

$$\int_\Omega \mu(\nabla \mathbf{u_h} + (\nabla \mathbf{u_h})^t) : \nabla \mathbf{v_h} - \int_\Omega p_h \nabla \cdot \mathbf{v_h} = -\int_\Omega F_s : \nabla \mathbf{v_h}, \tag{23}$$

$$\int_\Omega q_h \nabla \cdot \mathbf{u_h} = 0 \tag{24}$$

for all test functions $(\mathbf{v}_h, q_h) \in V_h \times Q_h$. See [36] for a derivation.

To evaluate the surface stress in the above weak formulation, the normal vector $\mathbf{n}$ at a cell vertex is calculated from the continuous level set function using a least squares fit similar to that described in Section 2.2.2. The only differences are that the origin of the local coordinate system is located at the cell vertex, and the fitting is over a stencil that consists of the vertex under consideration and all its nearest neighbors. In addition, as is in [36], in practice the Dirac delta function $\delta(\phi)$ is replaced by a smoothed version $\delta_\epsilon(\phi)$, where $\delta_\epsilon(\phi) = \mathrm{d}H_\epsilon/\mathrm{d}\phi$ and H_ϵ is a smoothed Heaviside function, defined as

$$H_\epsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\epsilon, \\ \frac{1}{2}\left(1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right) & \text{if } |\phi| \leqslant \epsilon, \\ 1 & \text{if } \phi > \epsilon, \end{cases} \tag{25}$$

where the parameter ϵ is taken to be 2–4 times the smallest mesh size. The local dynamic viscosity μ is also smoothed using the smoothed Heaviside function. Thus,

$$\mu = \mu_1 + (\mu_2 - \mu_1)H_\epsilon(\phi). \tag{26}$$

The resulting linear system from the discretization is solved using an inexact Uzawa method [45]. Matrix inversions are performed using SSOR preconditioned conjugate gradient method. See also [36].

2.5. Summary

The whole procedure is summarized as follows:

(1) At the beginning of a computation, the initial values of ϕ, and f are specified according to the initial interface position. Then, the flow field, i.e. the velocity and pressure, are initialized by solving the Stokes equations. Notice that since the Stokes equations are time independent, they can be solved as a quasi-steady problem at any time instance.

(2) Knowing the initial values of ϕ, f and $\mathbf{u}$, or the values of ϕ, f and $\mathbf{u}$ at time level n, we perform (i) the first step of the two-stage RKDG Eulerian level set evolution to obtain the intermediate Eulerian level set function, $\phi^{n+1/2}$, on the Eulerian grid; and (ii) the first step of the two-stage Runge–Kutta projection of the grid, in which grid nodes are projected according to Eq. (12). The projection results in an intermediate Lagrangian mesh. The position of a node in the intermediate Lagrangian mesh is given by $\tilde{\mathbf{x}}_i^{n+1/2}$.

(3) The intermediate Eulerian velocity $\mathbf{u}(\mathbf{x}_i^n, t^{n+\frac{1}{2}})$ is obtained by solving the Stokes equations on the Eulerian mesh, for which the interface normal vector in the surface stress term is calculated from the intermediate level set function $\phi^{n+1/2}$. The intermediate Lagrangian velocity $\mathbf{u}(\tilde{\mathbf{x}}_i^{n+1/2}, t^{n+\frac{1}{2}})$ is obtained from the intermediate Eulerian velocity $\mathbf{u}(\mathbf{x}_i^n, t^{n+\frac{1}{2}})$ through interpolation.

(4) Then, the second step of the RKDG level set evolution is performed to solve for ϕ^{n+1}, the Eulerian level set function at time level $n + 1$, and the second step of the Runge–Kutta grid projection is performed to obtain the final Lagrangian mesh corresponding to time level $n + 1$, in which grid nodes are projected according to Eq. (13).

(5) The volume fractions remain the same during the projection according to Eq. (10). The level set function on the final Lagrangian grid is obtained from ϕ^{n+1} through interpolation. Based on both the level set function and volume fraction, we reconstruct an interface on the final Lagrangian grid, as described in Section 2.2.2.

(6) To let grid refinement follow the interface motion, grid is adapted/rezoned based on the current interface position, which results in a new Eulerian grid.
(7) The values of f on the new Eulerian grid are obtained through remapping, as described in Section 2.2.3. The level set function on the new Eulerian grid is directly obtained as the signed distance function to the previously reconstructed interface on the Lagrangian grid. The distance function is calculated as the minimum distance of a point to the linear segments that constitute the interface. The flow velocity on the new Eulerian grid is obtained by solving the Stokes equations.
(8) Steps 2–7 are repeated until a specified end time is reached.

3. Results

In this section, we examine the integrity and accuracy of our ACLSVOF interface reconstruction and volume tracking method, and present numerical simulations of drop deformation in an extensional flow using our adaptive ACLSVOF method, for which the flow velocity is solved using the Stokes flow solver described in Section 2.4.

3.1. Interface reconstruction test

3.1.1. Reconstruction of a linear interface

For a linear interface defined by $\mathbf{n} \cdot \mathbf{x} = \alpha$, the analytic level set function is given by $\phi = \mathbf{n} \cdot \mathbf{x} - \alpha$, which is a linear function of the coordinates of a point. Because the least squares quadratic fitting approximates a linear function exactly, our normal calculation method, described in Section 2.2.2, computes the normal vector of a linear interface exactly. In addition, because the linear interface segments are finally reconstructed analytically using the analytic reconstruction method, described in Appendix A, our interface reconstruction method reconstructs a linear interface exactly. This implies that our interface reconstruction method is second order accurate [46].

3.1.2. Reconstruction of a circular interface

A circle of radius 0.368 centered at $(0.525, 0.464)$ in a $(0,1) \times (0,1)$ computational domain is reconstructed using the CLSVOF interface reconstruction method described in Section 2.2.2. The discrete volume fraction data are initialized exactly as the volume ratio of each cell occupied by the interior fluid. The level set at each node point is initialized exactly as the signed minimum distance of the node to the circle.

Fig. 3 shows the reconstructed interface on a 2×10^2 uniform triangular grid, which is obtained from a 10^2 uniform rectangular grid by dividing each rectangular cell into two triangular cells along one of its diagonals. The difference between the reconstructed interface and the analytic interface is nearly invisible.

To quantify the reconstruction error, an L_1 error norm is defined. Since a reconstructed interface segment does not exactly coincide with the analytic interface, there is a volume mismatch between the analytic interface and the reconstructed interface, as illustrated in Fig. 4. The L_1 error norm is defined as the summation of the volume mismatches in all interfacial cells, and is computed analytically. Table 1 shows the reconstruction errors and convergence rates on five uniform triangular grids. The results verify that our interface reconstruction method is second order accurate.

3.2. Volume tracking tests

Uniform translation and solid body rotation tests have been used widely for assessing the integrity and capability of an interface tracking method [6]. As stated in [6], "an acceptable volume tracking method must translate and rotate fluid bodies without significant distortion or degradation of fluid interfaces. Mass should also be converged rigorously in these cases". However, simple translation and rotation tests provide only a minimal assessment of the capability of a volume tracking method since the interface does not deform. Interfacial flows, for example the Rayleigh–Taylor, Richtmeyer–Meshkov, and Kelvin–Helmholtz instabilities, often involve strong vortical flows near the interface, and thus complex changes of the interface. "A complete assessment

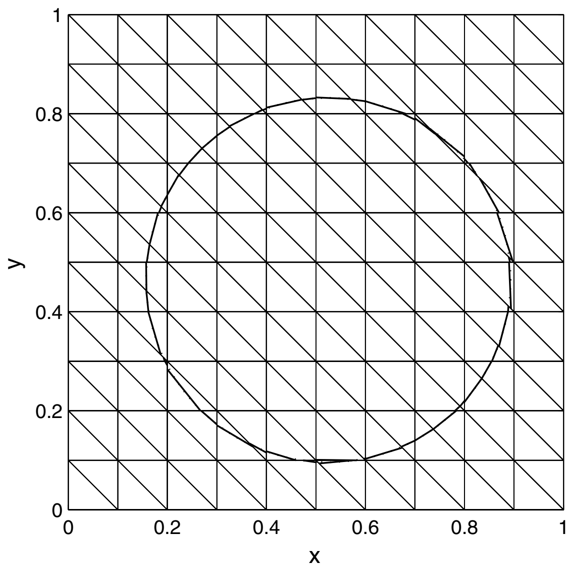


Fig. 3. Reconstructed interface on a 2×10^2 uniform triangular grid. The difference between the reconstructed interface and the analytic interface, which is plotted using a dotted line, is nearly invisible.

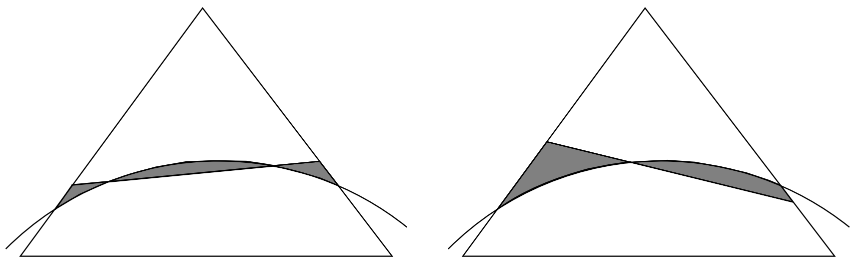


Fig. 4. Volume mismatch (shaded area) between the analytic circle and the reconstructed linear interface in a triangular cell.

Table 1
L_1 error norms, and convergence rates for circle reconstruction

Triangles	L_1 error	Order
2×10^2	1.47E−3	–
2×20^2	3.17E−4	2.21
2×40^2	8.67E−5	1.87
2×80^2	1.94E−5	2.16
2×160^2	5.26E−6	1.88

of interface tracking methods should therefore impose strong vorticity at the interface" [15]. We refer readers to the articles by Rider and Kothe [6,15] for an elegant discussion of choosing appropriate test problems.

In this section, we present results for four selected test problems: simple translation, and rotation of a circular fluid body, solid body rotation of Zalesak's slotted disk [47], and the single vortex problem introduced by Rider and Kothe [15], where the single vortex problem provides a rigorous test of a volume tracking method in cases where significant interface stretching occurs. These test problems have been widely used recently [6,23,24,30,48].

For the solid body rotation of Zalesak's slotted disk, the geometry and grid were chosen to allow comparison to data in the literature and are described below in Section 3.2.3. For all other tests, we perform convergence studies, for which an L_1 error norm is defined as

$$E^{L_1} = \int_\Omega |f^{\text{computed}}(\mathbf{x},t) - f^{\text{exact}}(\mathbf{x},t)|\,\mathrm{d}\mathbf{x}. \tag{27}$$

Convergence studies are performed on both fixed uniform grids and adaptive clustered grids. The fixed uniform grids are obtained from the corresponding uniform structured rectangular grids, with edge lengths of 0.032, 0.016, and 0.008, respectively, by dividing each rectangular cell into two triangular cells. For all adaptive grids, grids are clustered near the interface to render high resolution near the interface. The edge lengths of the smallest grid cells, denoted by $h_{\min}$, are roughly 0.032, 0.016, and 0.008, respectively for the three sets of adaptive grids. We expect the errors on the adaptive grids to be about the same size as on the corresponding uniform grids, because only the grid in the vicinity of the interface should affect the accuracy of the evolution of the interface position, but many fewer grid cells are used on the adaptive grids than on corresponding uniform grids. The *CFL* number for these tests is 0.7. The computational domain is a $(0,1)\times(0,1)$ square.

To quantify the accuracy of mass conservation of the method, we define a time averaged mass error as

$$E^M = \int_{t=0}^{T_f} \frac{|M(t) - M(0)|}{T_f}\,\mathrm{d}t, \tag{28}$$

where

$$M(t) = \int_\Omega f(\mathbf{x},t)\,\mathrm{d}\mathbf{x} \tag{29}$$

and T_f is the total integration time. As the velocity fields are given analytically for these tests, volume changes in these tests are primarily due to reasons (ii), (iii) and (iv), as elaborated in Section 2.2.1.

3.2.1. Simple translation of a circular fluid body

An initially circular fluid body of radius 0.25 centered in the computational domain is translated by a uniform velocity field along the 45° domain diagonal. The velocity remains constant with unit components, but changes sign at times 0.25 and 0.75, respectively. Therefore, the fluid body should return to its initial position after 1 time unit without any interface deformation.

Fig. 5 shows the final reconstructed interfaces and the corresponding L_1 error contours on the three sets of adaptive grids. The reconstructed interface and error contours are isotropic with respect to flow, exhibiting no bias towards the flow direction. The total number of triangles is subject to change as the grid adapts to the evolving interface. At $t = 0$, the total numbers of triangles in the computational domain are 764, 1722, and 3590, respectively, which are much less than the number of triangles of the corresponding uniform grids, especially as the minimum grid sizes become smaller (see Table 2).

Table 2 shows that our volume tracking method translates a fluid body with second order accuracy on both uniform grids and adaptive clustered grids. In addition, as we have expected, the L_1 errors on the adaptive grids are very close to those on the corresponding uniform grids because they have similar minimum grid sizes, while using significantly fewer grid cells. It is also shown that the mass errors are zero on each of the grids, verifying that our method conserves mass exactly for simple translation problems because the second order Runge–Kutta method integrates a constant function exactly.

3.2.2. Solid body rotation of a circular fluid body

An initially circular fluid body of radius 0.15 centered at the point $(0.5, 0.75)$ is revolved around the center of the unit domain with a constant unit angular velocity for one complete revolution. Therefore, the fluid body should return to its initial position after 2π time units without any change in the interface shape.

Fig. 6 shows the final reconstructed interface and the corresponding L_1 error contours on the three sets of adaptive grids. Table 3 shows second order convergence in the L_1 error norms for both uniform grids and adaptive grids. Again, the L_1 errors on the adaptive grids are very close to those on the corresponding uniform grids having similar minimum grid sizes, while the number of triangles are greatly reduced by using adaptive grids. The time averaged mass errors are not zero because the projection of a fluid element subject to rotation can not be integrated exactly. However, the mass errors are very small compared to the total mass of the drop,

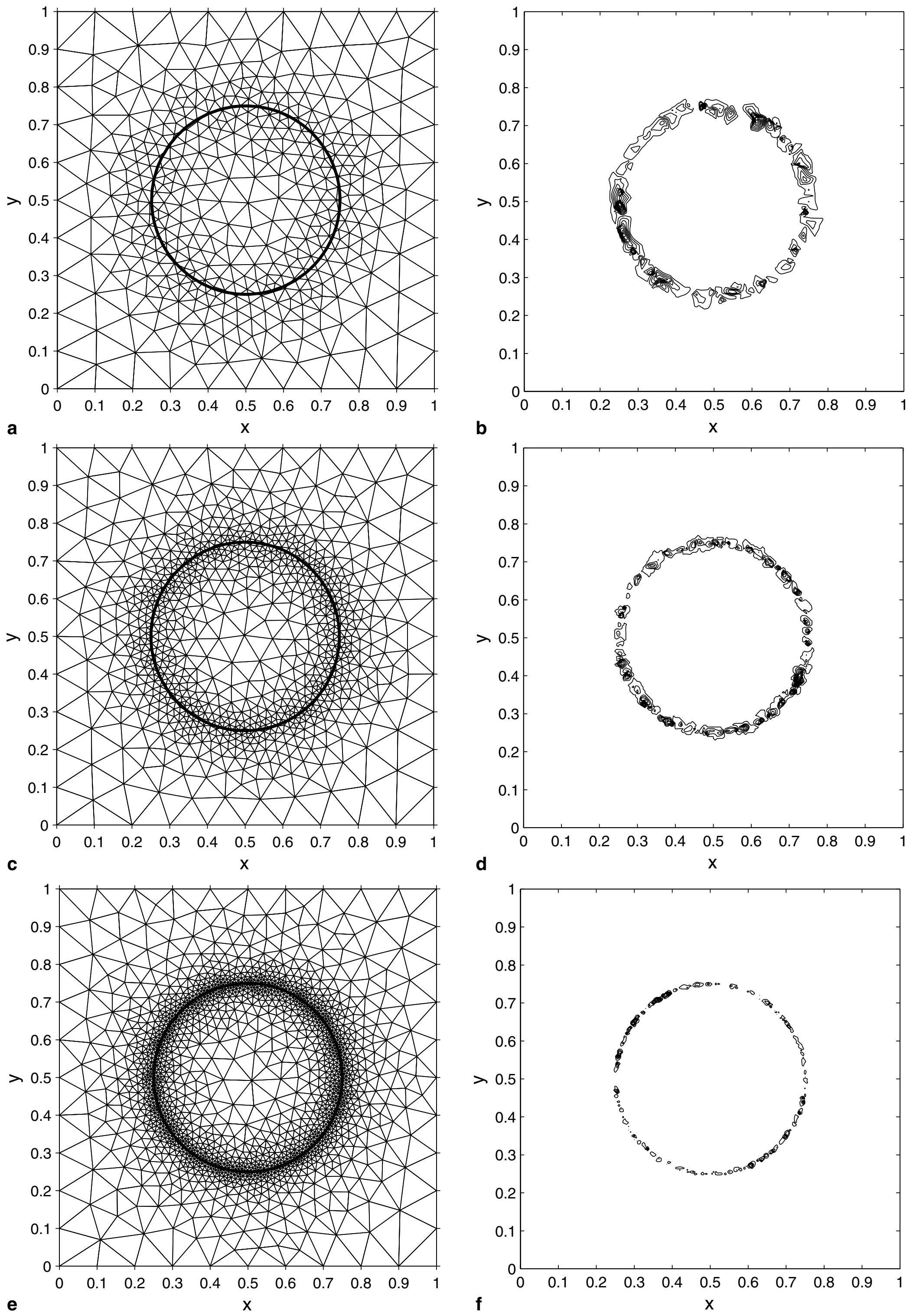


Fig. 5. Reconstructed interfaces (left) and the corresponding L_1 error contours (right) of an initially circular fluid body after a complete diagonal translation. $h_{\min} \approx 0.032$, 0.016, and 0.008, respectively from top to the bottom. The contours are of 20 levels from -3.1×10^{-6} to 2.7×10^{-6} for (b), from -4.0×10^{-7} to 2.9×10^{-7} for (d), and from -9.2×10^{-8} to 7.2×10^{-8} for (f).

Table 2
L_1 error norms, convergence rates and time averaged mass errors for the simple translation test

$h_{\min}$	Triangles	L_1 error	Order	Mass error
Fixed uniform grids				
0.032	2048	7.10E−5	–	0.0
0.016	8192	1.67E−5	2.09	0.0
0.008	32768	3.61E−6	2.21	0.0
Adaptive grids				
0.032	764	9.14E−5	–	0.0
0.016	1722	1.80E−5	2.34	0.0
0.008	3590	5.07E−6	1.83	0.0

and because they are much lower than the L_1 error norms, the mass errors do not nullify our convergence study. In addition, it is seen that the mass errors are significantly reduced by refining the grid.

3.2.3. Zalesak slotted disk rotation

Zalesak slotted disk [47] (refer to Fig. 7) is revolved about the center of the computational domain with a constant angular velocity. To compare our results with other's, we consider Rudman's version [49] of this problem, in which the specifications of the geometry are slightly modified from [47]. In particular, the computational domain is a $(0,4)\times(0,4)$ square. The circle is centered at $(2,2.75)$, with a diameter of 1. The width of the slot is 0.12. The maximum width of the upper bridge, that connects the left and right portions of the circle, is 0.4. The constant angular velocity is 0.5. Thus, the disk should return to its initial position after 4π time units without any interface change.

In most previous works, uniform rectangular grids, with 200 cells along each coordinate direction, were used. To compare our results to these previous works, we present test results on both a fixed uniform triangular grid and adaptive triangular grids. The uniform triangular grid is obtained from the uniform rectangular grid, with 200 cells along each coordinate direction, by dividing each rectangular cell into two triangles along one of its diagonals. For adaptive grids, we refine the grid near the interface such that the edge length of the smallest cells is roughly 0.02. In addition, as in [49] the time step is chosen such that a complete revolution is completed in 2524 time steps, corresponding to a Courant number of about 0.25, and the errors for this test are computed as

$$E^Z = \frac{\sum_T |f_T^{\text{computed}} - f_T^{\text{exact}}|}{\sum_T f_T^{\text{exact}}}. \tag{30}$$

Fig. 7(a) shows the reconstructed slotted disk after one complete revolution on an adaptive grid. The difference between the reconstructed disk and the exact solution is only visible in the small regions near the sharp corners, where the sharp corners are smoothed due to the linear reconstruction of the interface. This is consistent with those results presented by other researchers, for example in [49,48,23]. However, our result looks more symmetric than most other similar results, and is comparable with the results obtained using quadratic interface reconstruction methods in [23] (refer to Fig. 8 in [23] for a comparison). This is more evident when looking at the results after 10 complete revolutions, shown in Fig. 7(b) (refer to Fig. 9 in [23] for a comparison). It is seen that even after 10 complete revolutions most of the reconstructed interface still coincides with the exact solution. There is only mild overshooting at the corners. The adaptive grid for this computation is shown in Fig. 7(c), which only uses about 5500 grid cells, compared to $2\times200\times200$ grid cells on the uniform grid. Fig. 7(d) shows the reconstructed slotted disk after 10 complete revolutions on a refined adaptive grid, the minimum grid size of which is $\sqrt{2}$ times smaller than that of the previous grid. We see that refining the grid does improve the accuracy.

The E^Z errors after one complete revolution are tabulated in Table 4, and are compared with the errors presented by other researchers. It is seen that the error on the fixed uniform triangular grid is smaller than most of the other errors, and is only bigger than the errors of Scardovelli and Zaleski obtained using a quadratic fit. The error on adaptive grid is slightly larger than the error on the uniform triangular grid. This difference is mainly because the control of the smallest grid size of the adaptive grid is only approximate, i.e.,

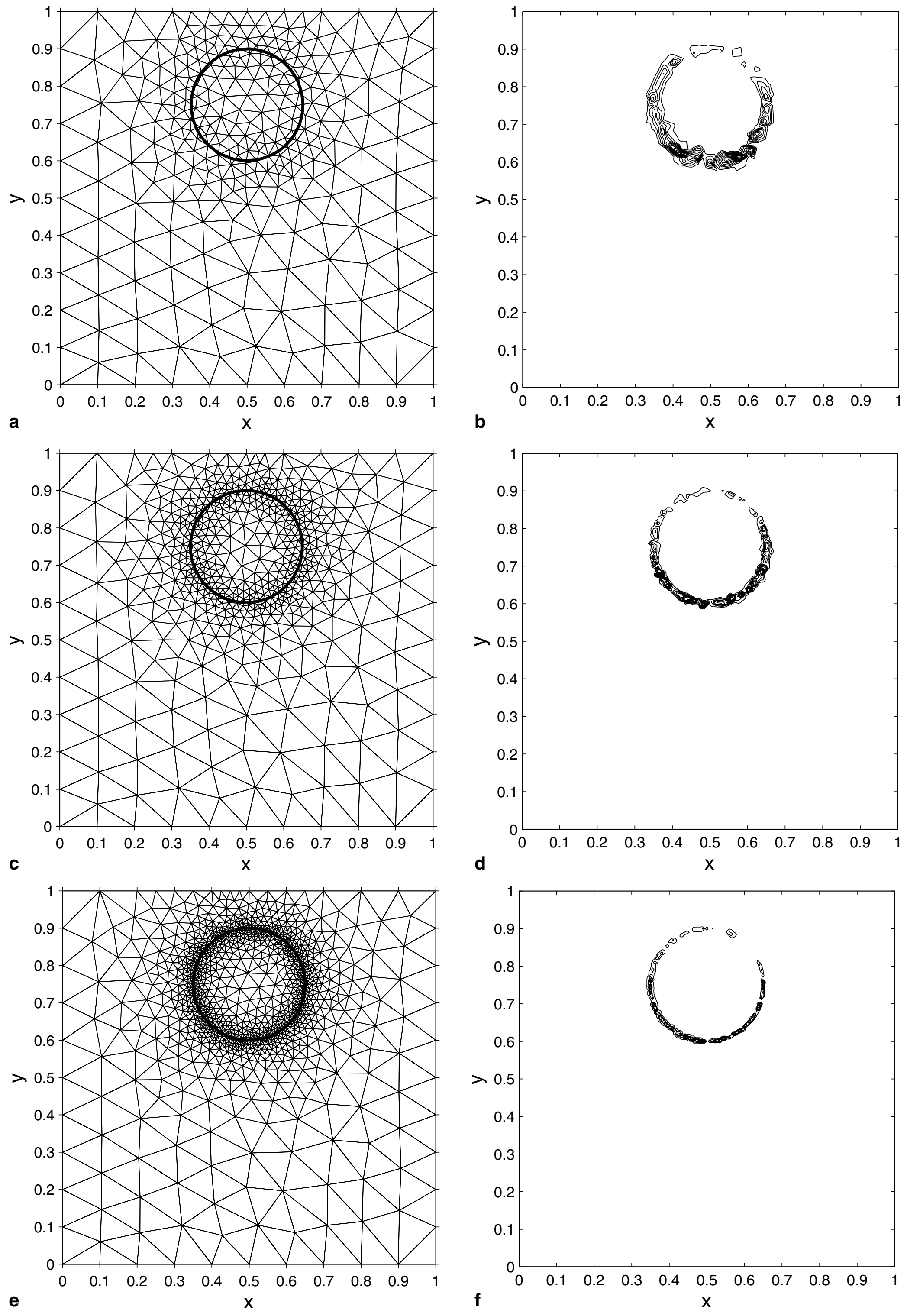


Fig. 6. Reconstructed interfaces (left) and the corresponding L_1 error contours (right) of an initially circular fluid body after a complete revolution. $h_{\min} \approx 0.032$, 0.016, and 0.008, respectively from top to the bottom. The contours are of 20 levels from -1.9×10^{-5} to 1.9×10^{-5} for (b), from -2.6×10^{-6} to 2.1×10^{-6} for (d), and from -3.0×10^{-7} to 4.0×10^{-7} for (f).

Table 3
L_1 error norms, convergence rates and time averaged mass errors for the solid bod

$h_{\min}$	Triangles	L_1 error
Fixed uniform grids		
0.032	2048	5.61E−4
0.016	8192	1.28E−4
0.008	32768	3.48E−5
Adaptive grids		
0.032	523	4.09E−4
0.016	1107	1.01E−4
0.008	2246	2.54E−5

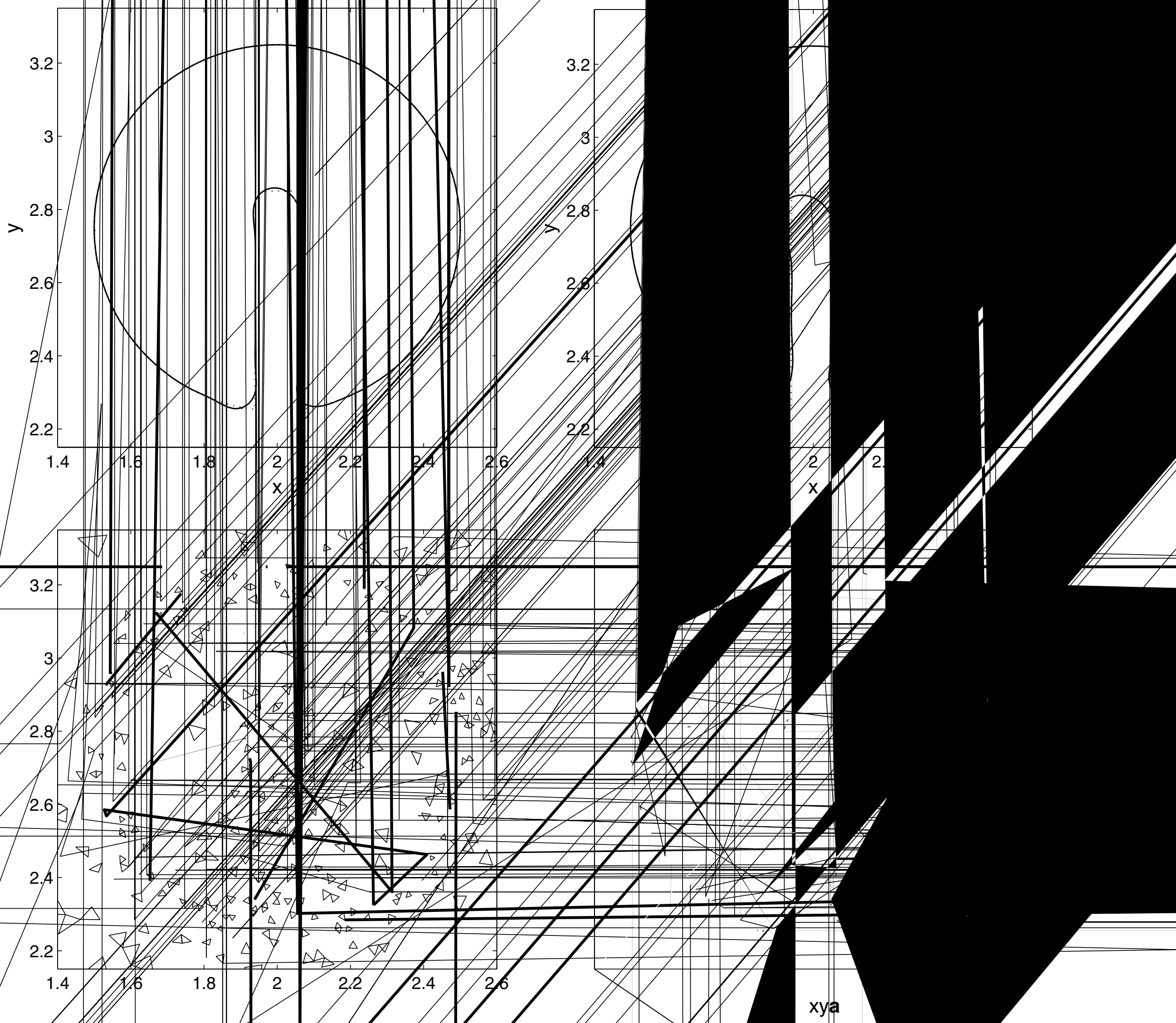


Fig. 7. Zalesak slotted disk test.

≈ 0.014.

Table 4
E^Z errors for Zalesak slotted disk test after one complete revolution

Advection/reconstruction methods	E^Z error
SLIC	8.38E−2
Hirt–Nichols	9.62E−2
FCT–VOF	3.29E−2
Youngs	1.09E−2
Harvie and Fletcher (Stream/Youngs)	1.07E−2
Harvie and Fletcher (Stream/Puckett)	1.00E−2
López et al. (EMFPA-SIR)	8.74E−3
present ACLSVOF (uniform triangular grid)	7.19E−3
present ACLSVOF (adaptive triangular grid)	1.25E−2
Scardovelli and Zaleski (linear least-square fit)	9.42E−3
Scardovelli and Zaleski (quadratic fit)	5.47E−3
Scardovelli and Zaleski (quadratic fit + continuity)	4.16E−3

All previous results are taken from Table 3 in [48].

the smallest grid cells vary slightly in size spatially and temporally. The time averaged mass errors for this test are on the order of 10^{-9}, which is extremely small.

3.2.4. *Single vortex flow*

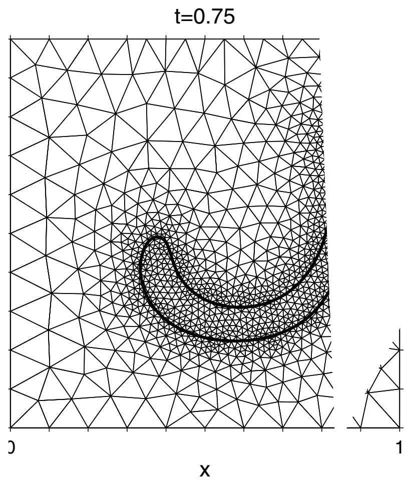
An initially circular fluid body of radius 0.15 centered at the point (0.5,0.75) is evolved by a single vortex flow field. The flow field is given by the stream function

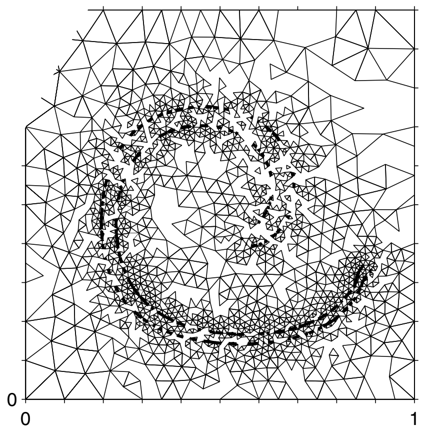
$$\Psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y), \tag{31}$$

where the velocity vector is defined by $(-\partial\Psi/\partial y, \partial\Psi/\partial x)$. This flow field was first introduced by Bell et al. in [50], and first used as the flow field in the "single vortex" problem for assessing the integrity and capability of an interface tracking method by Rider and Kothe in [15]. The flow field contains a single vortex centered in the domain. The vortex spins fluid elements, stretching them into a thin filament that spirals toward the vortex center. Thus each fluid element can undergo very large deformation.

Fig. 8 shows the evolution of the circular fluid body in the single vortex flow field. The solution is computed using an adaptive mesh of moderate resolution, for which the smallest grid cell has an edge length of 0.016. It is seen that the fluid body is spun into a thinning filament, which compares very well with those results obtained using other state-of-the-art methods, for example [6,24]. Theoretically, the filament spirals toward the vortex center, and becomes thinner and thinner continuously. However, Fig. 8 shows that the thin filament breaks into small pieces beginning at its tail after long time evolution. This is because the grid resolution has become low relative to the thinning filament when the thickness of the filament is equal to or less than the grid size. Because the interface is represented by a single line segment in each interfacial grid cell, any two interfaces merge automatically whenever they come into the same cell, resulting in an unphysical 'pinch-off'. Once the pinch-off occurs, the linear segment approximation of an interface immediately flattens the high curvature region, effectively applying numerical surface tension. Because the filament asymptotes to infinitesimal thickness, any grid will eventually provide inadequate resolution. However, this unphysical pinch-off can be delayed by refining the grid near the interface. Because we have implemented adaptive unstructured grids in our algorithm, we can refine the grid near the interface efficiently.

For convergence studies, the Leveque cosine term [51] is applied to the velocity field, i.e., the flow field defined by (31) is multiplied by $\cos(\pi t/T_f)$. By doing this, the flow field reverses in time such that any fluid body returns to its initial position at time T_f, allowing the error to be quantified with Eq. (27). Convergence studies are performed for three different values of T_f on both fixed uniform grids and adaptive grids. Fig. 9 shows the reconstructed interfaces and the corresponding L_1 error contours on the adaptive grid of $h_{\min} \approx 0.016$ for the three values of T_f. Quantitative error measurements and convergence rates are shown in Table 5. The convergence rates show that our ACLSVOF method is second order accurate, even when T_f is large, for which large complex topological change has occurred (see Fig. 10), indicating that our method is remarkably resilient for interface capturing. Once again, we see that the errors on the adaptive grids are similar to those on fixed





uniform ' :ause the initial fluid body is the s .e solid body ro
the gri' rt of the computations are the sa: .he solid body ro
are ' ıtive to the L_1 error norms, and nullify our conver

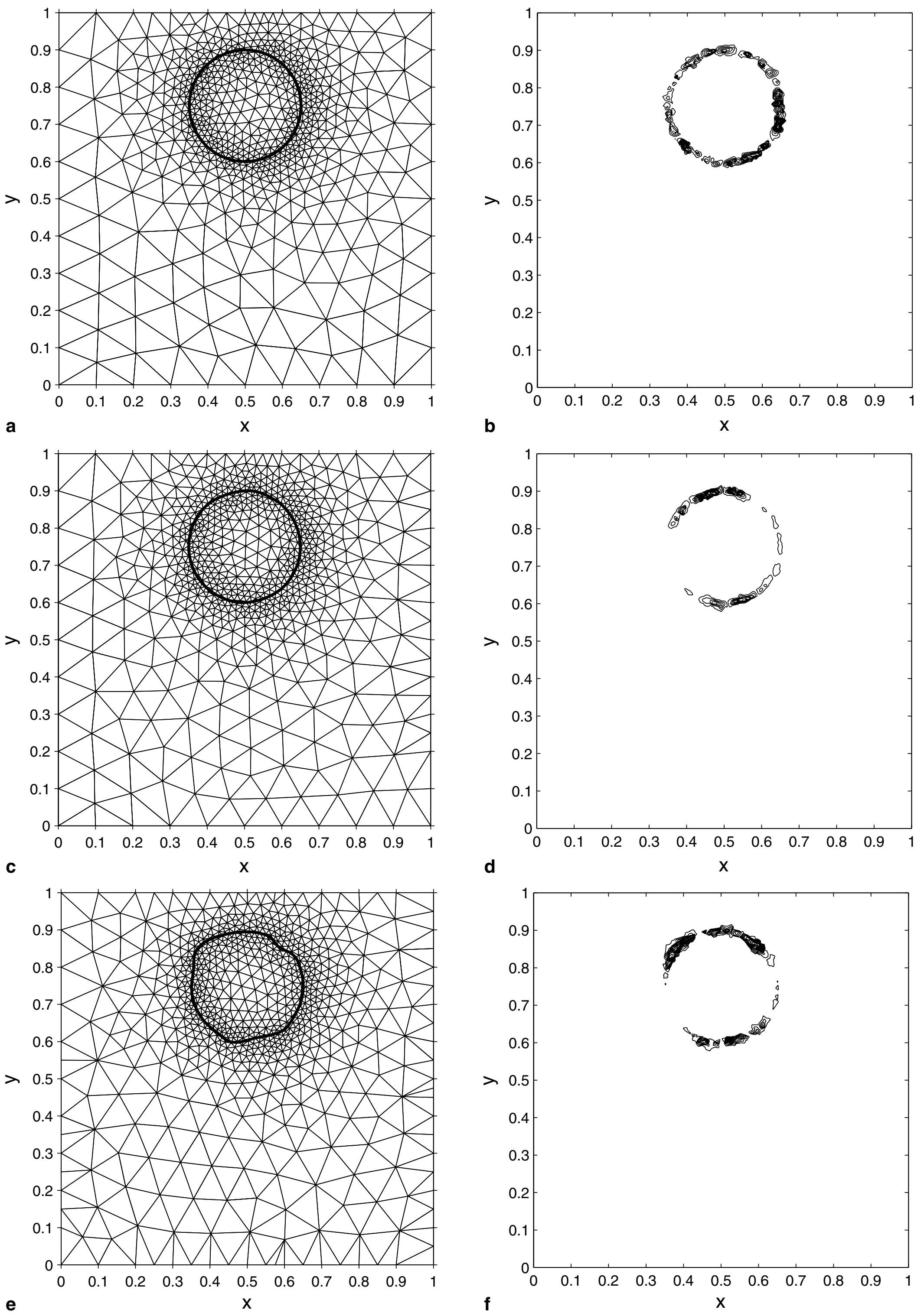


Fig. 9. Reconstructed interfaces (left) and the corresponding L_1 error contours (right) for an initially circular fluid body placed in a time-reversed, single vortex flow on an adaptive grid of $h_{\min} \approx 0.016$ at $t = T_f$, where $T_f = 0.5$, 2.0 and 8.0, respectively from top to the bottom. The contours are of 20 levels from -5.9×10^{-7} to 6.9×10^{-7} for (b), from -2.0×10^{-5} to 1.3×10^{-5} for (d), and from -9.2×10^{-5} to 8.8×10^{-5} for (f).

Table 6
L_1 error norms, and convergence rates obtained using different methods for the time-reversed, single vortex test with $T_f = 2.0$

Computational methods	Grid size	L_1 error	Order
Garrioch and Baliga (Pilliod–Puckett advection circle fit)	32×32	2.23E−3	–
	64×64	4.93E−4	2.18
	128×128	1.12E−4	2.14
Scardovelli and Zaleski (linear fit)	32×32	1.75E−3	–
	64×64	4.66E−4	1.91
	128×128	1.02E−4	2.19
Scardovelli and Zaleski (quadratic fit)	32×32	1.88E−3	–
	64×64	4.42E−4	2.08
	128×128	9.36E−5	2.24
Scardovelli and Zaleski (quadratic fit + continuity)	32×32	1.09E−3	–
	64×64	2.80E−4	1.96
	128×128	5.72E−5	2.29
Aulisa et al. (markers/VOF)	32×32	1.00E−3	–
	64×64	2.69E−4	1.89
	128×128	5.47E−5	2.30
López et al. (EMFPA-SIR)	32×32	8.62E−4	–
	64×64	2.37E−4	1.86
	128×128	5.62E−5	2.08
Present (fixed uniform grids)	$h_{\min} = 0.032$	1.41E−3	–
	$h_{\min} = 0.016$	3.38E−4	2.06
	$h_{\min} = 0.008$	6.84E−5	2.30
Present (adaptive grids)	$h_{\min} \approx 0.032$	1.62E−3	–
	$h_{\min} \approx 0.016$	3.36E−4	2.27
	$h_{\min} \approx 0.008$	7.95E−5	2.08

All previous results are taken from Table 5 in [48].

Table 7
L_1 error norms obtained using different methods for the time-reversed, single vortex test with $T_f = 8.0$

Advection/reconstruction methods	L_1 error
Rider and Kothe/Puckett	1.44E−3
Harvie and Fletcher/Youngs	2.16E−3
Harvie and Fletcher/Puckett	1.18E−3
EMFPA/Youngs	2.13E−3
EMFPA/Puckett	1.17E−3
EMFPA-SIR	7.57E−4
Aulisa et al., markers/VOF	4.78E−4
Aulisa et al., markers/local area conservation	4.06E−5
Present (fixed uniform grids)	5.61E−4
Present (adaptive grids)	5.09E−4

All previous results were taken from Table 7 in [48], except that the result of Aulisa et al. (markers/local area conservation) was taken from Table 1 in [52]. All previous results were obtained on a 128×128 rectangular grid. Results of the present paper were obtained on a fixed uniform grid and an adaptive grid (both with $h_{\min} \approx 0.008$), respectively.

and the velocity field will be computed by the Stokes flow solver. More specifically, we consider drop deformation in an extensional flow, which is modeled by experimentalists in four-roll mill devices, first introduced by G.I. Taylor [53]. A schematic of the problem is shown in Fig. 11.

The control of drop deformation and breakup is of fundamental importance in numerous industrial applications such as the dispersion of immiscible fluids into each other to create emulsions, the separation of liquid phases by tip streaming, and so on. It has attained wide investigation in the past and is still under intensive research [53–56]. A milestone, since the pioneering work by Taylor [53], is the experimental work by Bentley

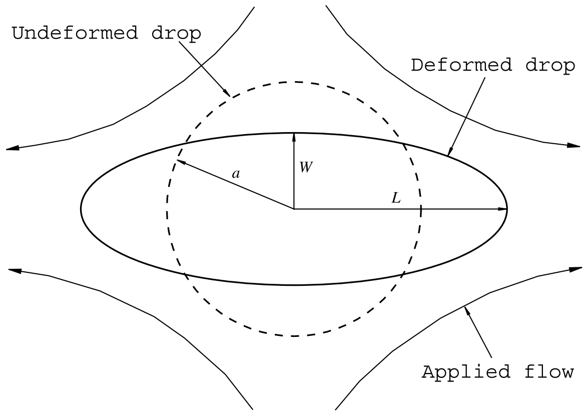


Fig. 11. Schematic of drop deformation in an extensional flow.

and Leal [54]. They designed a computer-controlled four-roll mill apparatus, which enabled them automatic, flexible and accurate control of the experiment with the capability to investigate drop deformation and breakup in a wide range of flow conditions, in particular, the transient flow in between pure extensional flow and pure shear flow, which had not been investigated experimentally up to then. Their work not only provided the first experimental data for drop deformation and breakup in this transient flow, but also improved the data for drop deformation in pure extensional flow and pure shear flow. The validity of small deformation theory and slender drop theory was also clarified. For a review of the research prior to 1994, we refer readers to the review paper by Stone [57].

We limit our simulation to two-dimensional low-speed Newtonian viscous flow with clean interfaces. Two dimensionless parameters determine the behavior of such a flow: the capillary number *Ca* and the viscosity ratio λ, where $Ca = \mu Ga/\sigma$, μ is the viscosity of the suspending fluid, *G* is the applied strain rate, *a* is the radius of the undeformed drop, and λ is the ratio of the drop to the suspending fluid viscosity. According to Bentley and Leal [54], the flow behavior can be qualitatively classified into three categories: low-viscosity-ratio drops ($\lambda < 0.02$), intermediate-viscosity-ratio drops ($0.02 < \lambda < 2.0$), and high-viscosity-ratio drops ($\lambda > 3.0$). For a detailed description of the characteristics of each flow category, we refer readers to Bentley and Leal [54]. For the purpose of code validation in the present paper, we choose a fixed viscosity ratio $\lambda = 0.1$, for which recent experimental data are available in Hu et al. [55]. Drop deformation under different *Ca* are investigated and compared to the experiment. As is usual, drop deformation is defined for a steady drop as $D_f = (L - W)/(L + W)$, where *L* and *W* are the half length and half width of the drop, respectively. For the numerical computation, we take a $(-10, 10) \times (-10, 10)$ computational domain, the size of which has been chosen to be large enough to avoid the effect of imposing the exact extensional flow on the boundary. The mesh is refined near interfaces so that the solution is grid independent. A *CFL* number of 0.7 is used for all computations.

Fig. 12 shows drop deformation as a function of capillary number for $\lambda = 0.1$, where the final data point in each set corresponds to the maximum value of *Ca* for which a steady drop shape is possible. The experimental result is from Hu et al. [55]. It is shown that our numerical results agree with the experimental results very well in a wide range of *Ca*, except near the critical point. As *Ca* is further increased, the drop symmetrically evolves into a thin filament with rounded ends and concave sides. Fig. 13 shows the evolution of a drop for $\lambda = 0.1$ and $Ca = 0.23$, which agrees qualitatively very well with the experimental drop shape described by Bentley and Leal for the intermediate-viscosity-ratio drops [54] and the experimental result by Hu et al. [55]. Fig. 14 shows the adaptive grids at $t = 0$ and $t = 8.4$. Observe that the grid is highly refined near the interface and tracks the evolution of the interface throughout the domain.

We monitored the time averaged mass errors for all simulations. Table 8 shows the time averaged mass errors obtained on different grids and using various interface thicknesses for a test case with $\lambda = 0.1$ and $Ca = 0.1$, an intermediate value in our study. The *CFL* number for this test is 0.3. The total integration

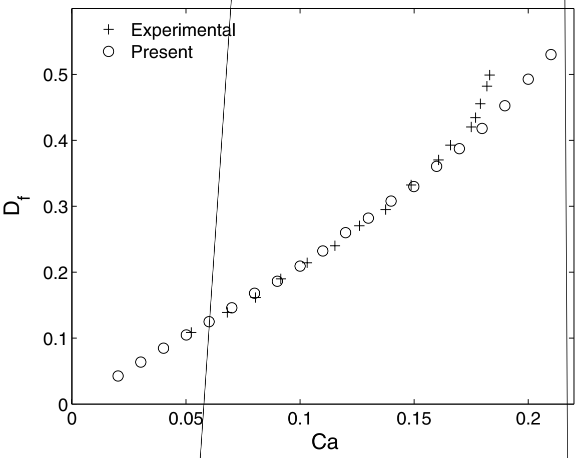


Fig. 12. Dependence of drop deformation on capillary number in pure extensional flow for $\lambda = 0.1$. The experimental data are from Hu et al. [55].

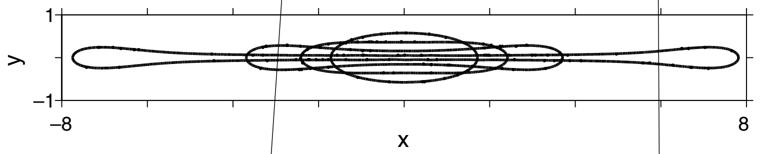
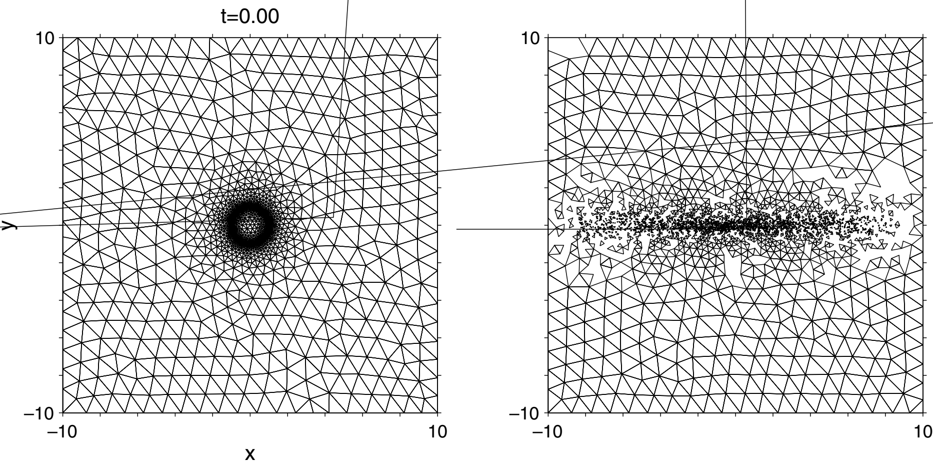


Fig. 13. Drop deformation in an extensional flow ($\lambda = 0.1$ and $Ca = 0.23$). The interfaces correspond to $t = 1.0$, 5.0, 7.0, and 8.4, respectively.



time is one unit. We analyze how the mass error decreases as the grid is refined and as the interface thickness is increased. Mass errors for test cases of other Ca numbers have similar behavior. From Table 8, we see that:

(1) For the same grid, using a relatively large ϵ (recall that ϵ is the half width of the smoothed interface) reduces the mass error significantly. This is because a larger interface thickness results in more smoothing of the discontinuity in fluid properties across the interface and of the singular surface tension force.

Table 8
Time averaged mass errors for drop deformation in an extensional flow ($Ca = 0.1$)

	$h_{\min} \approx 0.1$	$h_{\min} \approx 0.05$	$h_{\min} \approx 0.025$
$\epsilon = 2h_{\min}$	2.83E−3	1.43E−3	7.47E−4
$\epsilon = 3h_{\min}$	6.55E−4	3.18E−4	2.46E−5
$\epsilon = 4h_{\min}$	2.05E−4	6.03E−5	5.44E−6
$\epsilon = 5h_{\min}$	1.84E−4	9.52E−5	1.66E−5

The numbers in the table are the mass errors. From left to right the minimum grid sizes are reduced. From top to bottom the interface thickness increases.

Smoothing reduces the numerical velocity divergence near the interface, which, as a consequence, reduces the mass error. Also, we see that for the same grid the mass errors corresponding to $\epsilon = 5h_{\min}$ are very close to or even larger than those corresponding to $\epsilon = 4h_{\min}$, indicating that $\epsilon = 4h_{\min}$ is large enough to smooth the discontinuities.

(2) For a constant $\epsilon/h_{\min}$ refining the grid reduces the mass error. Especially, when $\epsilon/h_{\min}$ is larger, say $\epsilon = 4h_{\min}$, refining the grid reduces the mass error more effectively. When $\epsilon/h_{\min}$ is small, say $\epsilon = 2h_{\min}$, the discontinuous fluid properties and the singular surface tension force are not well smoothed, which makes the numerical velocity divergence near the interface relatively high, and thus refining the grid is not as effective as when $\epsilon = 4h_{\min}$.

(3) If we let the interface thickness, instead of the ratio $\epsilon/h_{\min}$, be a constant, refining the grid reduces the mass error substantially. For instance, in Table 8 the case with $h_{\min} \approx 0.1$ and $\epsilon = 2h_{\min}$ has the same interface thickness as the case with $h_{\min} \approx 0.05$ and $\epsilon = 4h_{\min}$. We see that the latter case has a much lower mass error.

(4) The maximum mass error in Table 8 is 2.83E−3, which is only 0.09% of the mass of the drop. In practice, the effect of this small amount of mass change on the overall fluid configuration is negligible.

We conclude that the mass errors in our simulation are small, and more importantly, with a moderate interface thickness refining the grid can reduce the mass error substantially.

4. Conclusions and future work

We have developed an adaptive coupled level set/volume-of-fluid method for capturing an interface on triangular unstructured grids. The method takes advantage of the strengths of both the level set and volume-of-fluid methods, which makes the calculation of the interface normal vector, and thus the surface tension easier and more accurate. Mass is also conserved very accurately. The analytic interface reconstruction algorithm we have developed has proven to be efficient, and conserve mass exactly. Numerical validation showed that our ACLSVOF volume tracking method is second order accurate, and is comparable to other state-of-the-art methods. The method is also coupled to a finite element based Stokes flow solver. Physical simulations of drop deformation in an extensional flow showed that our method can capture the surface tension force accurately, and is capable of analyzing a physical problem. In addition, owing to the adaptive algorithm, we resolve complex interface changes and interfaces of high curvature accurately and efficiently.

In the future, we will couple our ACLSVOF volume tracking method to a Navier–Stokes solver to incorporate the effect of inertia. We will also extend the method to the axisymmetric and 3D cases. For the 3D case, the Stokes and the Navier–Stokes flow solvers and the discontinuous Galerkin level set evolution have already been developed [35,36]. Concerning the Lagrangian–Eulerian volume fraction advection, the 3D analytic interface reconstruction and the 3D polyhedron clipping will present some challenges. Fortunately, a solution for the 3D analytic interface reconstruction has been formulated [38], and the Sutherland–Hodgeman algorithm was also targeted to 3D polyhedron clipping [42]. Finally, we will also incorporate the effects of surfactants [58,59]. Surfactants play a crucial role in drop deformation and breakup, especially in producing very tiny droplets by tipstreaming. Implementations of these ingredients are underway.

Acknowledgements

The first and second authors wish to thank the Minnesota Supercomputer Institute and the 3M Corporation for partial support. The remaining authors acknowledge the support of the National Science Foundation (Division of Mathematical Sciences). The authors wish to thank the reviewers for their useful suggestions.

Appendix A. Analytic interface reconstruction on triangular grids

Given the volume fraction f and the interface normal vector $\mathbf{n}$ in a triangular cell, we want to find the unique linear segment whose normal vector is $\mathbf{n}$ and which truncates the cell with the given volume fraction. Let A, B, and C be the three vertices of the cell. We draw a line with normal $\mathbf{n}$ through an arbitrary vertex of the cell. For example, without loss of generality we can draw such a line through vertex A as shown in Fig. A.1. Note that in all figures the triangles have been rotated so that the line segment appears horizontal. The relationship between the line and the triangle can be divided into three categories: (i) the whole triangle is on one side of the line (see Fig. A.1(a)), (ii) the triangle is divided by the line into two parts, one on each side, (see Fig. A.1(b)), and (iii) the line coincides with one of the edges of the triangle (see Fig. A.1(c)). Mathematically, the applicable category can be determined from two inner products: $p_B = \overrightarrow{AB} \cdot \mathbf{n}$ and $p_C = \overrightarrow{AC} \cdot \mathbf{n}$, where the over-right arrows are used to denote a vector pointing from the first point to the second point. If p_B and p_C are of the same sign, then B and C are on the same side of the line, and category (i) applies. If they are of different signs, then category (ii) applies. If one of the inner products is zero, then category (iii) applies. We assume that the triangle does not degenerate to a straight line, a requirement of grid generation, so p_B and p_C can not be zero at the same time. Notice that we have the option to transform category (ii) to category (i) simply by exchanging the indices of the three vertices.

For category (i), without loss of generality, we assume that $|p_B| \leqslant |p_C|$ (for $|p_B| > |p_C|$, we only need to exchange the indices B and C in the results below), and define $f^* = V_{\Delta ABD}/V_{\Delta ABC}$ (see Fig. A.2). Note that in all figures BD and EF are parallel to the given line segment. It is easy to show that

$$f^* = \frac{V_{\Delta ABD}}{V_{\Delta ABC}} = \frac{p_B}{p_C}. \tag{A.1}$$

If p_B and p_C are positive, then vertex A must be in fluid 1 (we assume that the normal vector points away from fluid 1 and to fluid 2). This fluid configuration is shown in Fig. A.2(a) and (b), which correspond to $f < f^*$ and $f > f^*$, respectively. For $f < f^*$ (Fig. A.2(a)), we have that

$$\frac{|\overrightarrow{AE}|}{|\overrightarrow{AB}|} = \frac{|\overrightarrow{AF}|}{|\overrightarrow{AD}|} = \sqrt{\frac{V_{\Delta AEF}}{V_{\Delta ABD}}} = \sqrt{\frac{f}{f^*}} \quad \text{and} \quad \overrightarrow{AD} = \frac{p_B}{p_C}\overrightarrow{AC} = f^*\overrightarrow{AC}.$$

Thus, the coordinates of the two end points, E and F, of the interface segment in the triangle ΔABC are

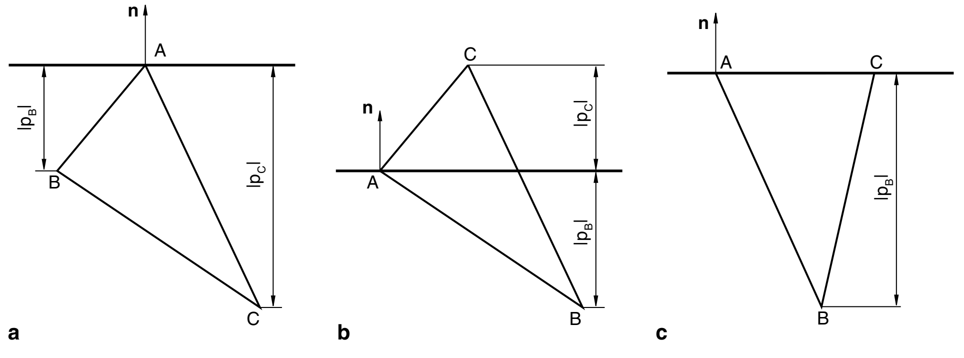


Fig. A.1. Drawing a line segment through a vertex of a triangular cell: (a) $p_B p_C > 0$, (b) $p_B p_C < 0$ and (c) $p_B p_C = 0$.

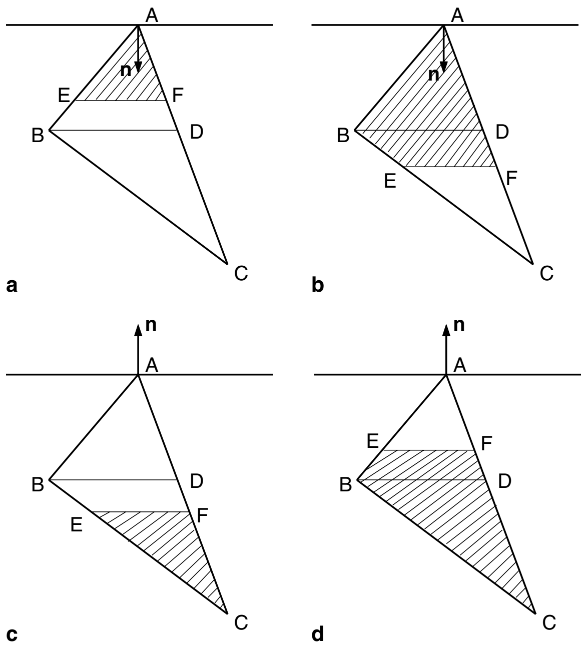


Fig. A.2. Fluid configuration for $p_B p_C > 0$: (a) $f < f^*$, (b) $f > f^*$, (c) $f < 1 - f^*$ and (d) $f > 1 - f^*$.

$$\mathbf{x}_E = \mathbf{x}_A + \sqrt{\frac{f}{f^*}}\overrightarrow{AB} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_A + \sqrt{ff^*}\,\overrightarrow{AC}.$$

For $f > f^*$ (Fig. A.2(b)), we have that

$$\frac{|\overrightarrow{CE}|}{|\overrightarrow{CB}|} = \frac{|\overrightarrow{CF}|}{|\overrightarrow{CD}|} = \sqrt{\frac{V_{\Delta CEF}}{V_{\Delta CBD}}} = \sqrt{\frac{1-f}{1-f^*}} \quad \text{and} \quad \overrightarrow{CD} = (1-f^*)\,\overrightarrow{CA}.$$

Thus,

$$\mathbf{x}_E = \mathbf{x}_C + \sqrt{\frac{1-f}{1-f^*}}\overrightarrow{CB} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_C + \sqrt{(1-f)(1-f^*)}\,\overrightarrow{CA}.$$

If p_B and p_C are negative, then vertex C must be in fluid 1. This configuration is shown in Fig. A.2(c) and (d), which correspond to $f < 1 - f^*$ and $f > 1 - f^*$, respectively. Similarly, we can derive that for $f < 1 - f^*$

$$\mathbf{x}_E = \mathbf{x}_C + \sqrt{\frac{f}{1-f^*}}\overrightarrow{CB} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_C + \sqrt{f(1-f^*)}\,\overrightarrow{CA}$$

and for $f > 1 - f^*$

$$\mathbf{x}_E = \mathbf{x}_A + \sqrt{\frac{1-f}{f^*}}\overrightarrow{AB} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_A + \sqrt{(1-f)f^*}\,\overrightarrow{AC}.$$

For category (ii), $p_B p_C < 0$, we assume $p_B < 0$ and $p_C > 0$ without loss of generality, and define $f^* = V_{\Delta ABD}/V_{\Delta ABC} = p_B/(p_B - p_C)$ (see Fig. A.3). Under this assumption, vertex B must be in fluid 1, as shown in Fig. A.3(a) and (b), which correspond to $f < f^*$ and $f > f^*$, respectively. We can show that for $f < f^*$

$$\mathbf{x}_E = \mathbf{x}_B + \sqrt{\frac{f}{f^*}}\overrightarrow{BA} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_B + \sqrt{ff^*}\,\overrightarrow{BC}$$

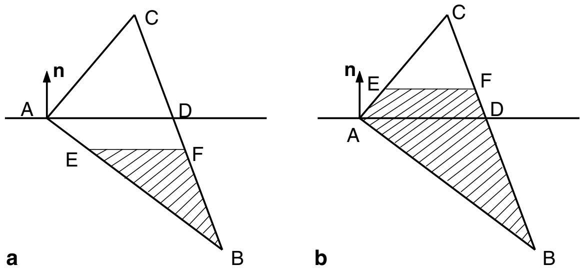


Fig. A.3. Fluid configuration for $p_B < 0$ and $p_C > 0$: (a) $f \leqslant f^*$ and (b) $f > f^*$.

and for $f > f^*$

$$\mathbf{x}_E = \mathbf{x}_C + \sqrt{\frac{1-f}{1-f^*}}\overrightarrow{CA} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_C + \sqrt{(1-f)(1-f^*)}\,\overrightarrow{CB}.$$

For category (iii), we assume $p_B \neq 0$ and $p_C = 0$ without loss of generality. The fluid configuration is shown in Fig. A.4(a) and (b), which correspond to $p_B < 0$ and $p_B > 0$, respectively. We can show that for $p_B < 0$

$$\mathbf{x}_E = \mathbf{x}_B + \sqrt{f}\,\overrightarrow{BA} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_B + \sqrt{f}\,\overrightarrow{BC}$$

and for $p_B > 0$

$$\mathbf{x}_E = \mathbf{x}_B + \sqrt{1-f}\,\overrightarrow{BA} \quad \text{and} \quad \mathbf{x}_F = \mathbf{x}_B + \sqrt{1-f}\,\overrightarrow{BC}.$$

The resulting piecewise linear interface conserves the volume of fluid exactly. The method is analytical, efficient, and extendible to 3D. For each reconstruction, the method only requires several basic algebraic operations, one square root, and several if-else operations, compared to a number of operations that is unknown a priori in iterative methods. Further details and a 3D algorithm are formulated in [38].

Appendix B. Sutherland–Hodgeman polygon clipping

In this appendix, we describe briefly the Sutherland–Hodgeman polygon clipping algorithm [42] used in Section 2.2.3. The basic idea of this algorithm is to "trim" the subject polygon, which is to be clipped against the clip polygon, by each extended edge of the clip polygon. In our case, the subject polygon is a Lagrangian fluid region that is occupied by fluid 1, and the clip polygon is a triangular Eulerian grid cell. The clip polygon must be convex for this algorithm, which is satisfied in our case because all the grid cells are convex. As an example, Fig. B.1 illustrates the clipping of a four-sided polygonal fluid element $p_1p_2p_3p_4$ against a triangular grid cell $q_1q_2q_3$, where $p_1p_2p_3p_4$ is trimmed by the three extended edges of $q_1q_2q_3$ in order, as shown in Figs. B.1(b), (c), and (d), respectively.

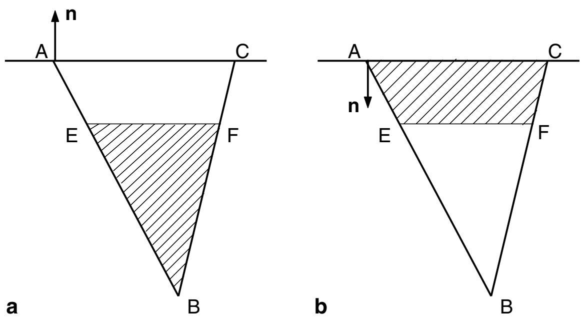


Fig. A.4. Fluid configuration for $p_B \neq 0$ and $p_C = 0$: (a) $p_B < 0$ and (b) $p_B > 0$.

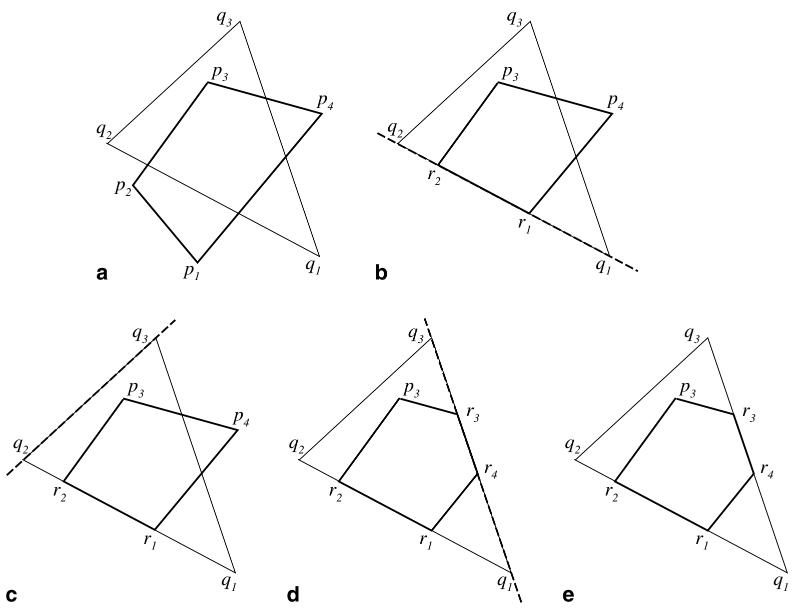


Fig. B.1. Clipping of a four-sided polygonal fluid element against a triangular grid cell. (a) The polygons before clipping. (b) Clipping of $p_1p_2p_3p_4$ against edge q_1q_2. (c) Clipping of $r_1r_2p_3p_4$ against edge q_2q_3. (d) Clipping of $r_1r_2p_3p_4$ against edge q_3q_1. (e) The polygons after clipping.

To illustrate the "trimming" process, let us consider the clipping of $p_1p_2p_3p_4$ against the first edge q_1q_2 for example, which will result in a clipped polygon $r_1r_2p_3p_4$ (see Fig. B.1(b)). For convenience, we name the infinitely-long line, obtained by extending edge q_1q_2, $\mathscr{L}$, and define that a vertex p_j is on the "right" side of $\mathscr{L}$ if p_j is on the same side of $\mathscr{L}$ as the other vertex q_3 is, and is on the "left" side otherwise. Then, the trimming process can be described as in Algorithm 1.

Algorithm 1. Trimming a polygon by a line.

```
for each vertex pair $p_j$, $p_{j+1}$ of the subject polygon
    if both $p_j$ and $p_{j+1}$ are on the right side of $\mathscr{L}$ then
        Add $p_{j+1}$ to the list of the vertices of the clipped polygon.
    else if $p_j$ is on the right and $p_{j+1}$is on the left then
        Add the intersection of the line segment $p_jp_{j+1}$ and $\mathscr{L}$ to the list of the vertices of the clipped polygon.
    else if $p_j$ is on the left and $p_{j+1}$is on the right then
        Add both the intersection and vertex $p_{j+1}$ to the list of the vertices of the clipped polygon.
    else
        Do nothing.
    end if
end for
```

The list of the vertices produced from the trimming process defines the clipped polygon, that is $r_1r_2p_3p_4$ for this example. In general, the trimming process is reentered to clip the previously clipped polygon against the subsequent edges of the grid cell, and the overall clipping completes as soon as the polygon has been trimmed by all the edges of the grid cell.

References

[1] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189.

[2] S.J. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[3] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372.
[4] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[5] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[6] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[7] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[8] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1992) 25–37.
[9] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2001) 708–759.
[10] V.E. Badalassi, H.D. Ceniceros, S. Banerjee, Computation of multiphase systems with phase field models, J. Comput. Phys. 190 (2003) 371–397.
[11] J.S. Kim, K. Kang, J. Lowengrub, Conservative multigrid methods for Cahn–Hilliard fluids, J. Comput. Phys. 193 (2004) 511–543.
[12] D. Jacqmin, Calculation of two-phase Navier–Stokes flows using phase-field modeling, J. Comput. Phys. 155 (1999) 96–127.
[13] D.M. Anderson, G.B. McFadden, A.A. Wheeler, Diffuse-interface methods in fluid mechanics, Annu. Rev. Fluid Mech. 30 (1998) 139–165.
[14] B.J. Keestra, P.C.J.V. Puyvelde, P.D. Anderson, H.E.H. Meijer, Diffuse interface modeling of the morphology and rheology of immiscible polymer blends, Phys. Fluids 15 (9) (2003) 2567–2575.
[15] W.J. Rider, D.B. Kothe, Stretching and tearing interface tracking methods, Tech. Rep. AIAA 95-1717, AIAA, the 12th AIAA CFD Conference, San Diego, 1995.
[16] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.
[17] M. Sussman, E. Fatemi, P. Smereka, S. Osher, An improved level set method for incompressible two-phase flows, Comput. Fluids 27 (1998) 663–680.
[18] M. Sussman, E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (4) (1999) 1165–1191.
[19] S.J. Mosso, B.K. Swartz, D.B. Kothe, R.C. Ferrell, A parallel, volume-tracking algorithm for unstructured meshes, in: Parallel Computational Fluid Dynamics'96, Italy, 1996.
[20] G.R. Price, G.T. Reader, R.D. Rowe, J.D. Bugg, A piecewise parabolic interface calculation for volume tracking, in: Proceedings of the Sixth Annual Conference of the Computational Fluid Dynamics Society of Canada, vol. VIII, 1998, pp. 71–77.
[21] Y. Renardy, M. Renardy, PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method, J. Comput. Phys. 183 (2002) 400–421.
[22] I. Ginzburg, G. Wittum, Two-phase flows on interface refined grids modeled with VOF, staggered finite volumes, and spline interpolants, J. Comput. Phys. 166 (2001) 302–335.
[23] R. Scardovelli, S. Zaleski, Interface reconstruction with least-square fit and split Eulerian–Lagrangian advection, Int. J. Numer. Methods Fluids 41 (2003) 251–274.
[24] K. Shahbazi, M. Paraschivoiu, J. Mostaghimi, Second order accurate volume tracking based on remapping for triangular meshes, J. Comput. Phys. 188 (2003) 100–122.
[25] N. Ashgriz, T. Barbat, G. Wang, A computational Lagrangian–Eulerian advection remap for free surface flows, Int. J. Numer. Methods Fluids 44 (2004) 1–32.
[26] D.B. Kothe, W.J. Rider, S.J. Mosso, J.S. Brock, Volume tracking of interfaces having surface tension in two and three dimensions, Tech. Rep. AIAA 96-0859, AIAA, the 34th Aerospace sciences meeting and exhibit, 1996.
[27] M.W. Williams, D.B. Kothe, E.G. Puckett, Convergence and accuracy of kernel-based continuum surface tension models, in: Proceedings of the Thirteenth US National Congress of Applied Mechanics, Gainesville, FL, 1998.
[28] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[29] M. Sussman, A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles, J. Comput. Phys. 187 (2003) 110–136.
[30] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set methods for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[31] E. Aulisa, S. Manservisi, R. Scardovelli, A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows, J. Comput. Phys. 188 (2003) 611–639.
[32] D. Greaves, A quadtree adaptive method for simulating fluid flows with moving interfaces, J. Comput. Phys. 194 (2004) 35–56.
[33] M. Sussman, A parallelized, adaptive algorithm for multiphase flows in general geometries, J. Comput. Struct. 83 (2005) 435–444.
[34] M. Sussman, M.Y. Hussaini, K. Smith, R.-Z. Wei, A second order adaptive sharp interface method for incompressible multiphase flow, in: Proceedings of the Third International Conference on Computational Fluid Dynamics, Toronto, Canada, 2004.
[35] A. Anderson, X. Zheng, V. Cristini, Adaptive unstructured volume remeshing-I: the method, J. Comput. Phys. 208 (2005) 616–625.
[36] X. Zheng, J. Lowengrub, A. Anderson, V. Cristini, Adaptive unstructured volume remeshing-II: application to two- and three-dimensional level-set simulations of multiphase flow, J. Comput. Phys. 208 (2005) 626–650.

[37] V. Cristini, J. Blawzdziewicz, M. Loewenberg, An adaptive mesh algorithm for evolving surfaces: simulations of drop breakup and coalescence of interfaces in two-phase and free-boundary flows, J. Comput. Phys. 168 (2001) 445–463.
[38] X. Yang, A.J. James, Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids, J. Comput. Phys., in press, doi:10.1016/j.jcp.2005.09.002.
[39] B. Cockburn, S. Hou, C.-W. Shu, The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, Math. Comput. 54 (1990) 545–581.
[40] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (2001) 173–261.
[41] A.V. Gelder, Efficient computation of polygon area and polyhedron volume, in: A.W. Paeth (Ed.), Graphics Gems V, Academic Press, 1995, pp. 35–41.
[42] I.E. Sutherland, G.W. Hodgeman, Reentrant polygon clipping, Communications of the ACM 17 (1) (1974) 32–42.
[43] K. Weiler, P. Atherton, Hidden surface removal using polygon area sorting, in: SIGGRAPH, 1977, pp. 214–222.
[44] D.N. Arnold, F. Brezzi, M. Fortin, A stable finite element for the Stokes equations, Calcolo 21 (1984) 337–344.
[45] H.C. Elman, G.H. Golub, Inexact and preconditioned Uzawa algorithms for saddle point problems, SIAM J. Numer. Anal. 31 (6) (1994) 1645–1661.
[46] J.E. Pilliod, E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2) (2004) 465–502.
[47] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.
[48] J. López, J. Hernández, P. Gómez, F. Faura, A volume of fluid method based on multidimensional advection and spline interface reconstruction, J. Comput. Phys. 195 (2004) 718–742.
[49] M. Rudman, Volume-tracking methods for interfacial flow calculations, Int. J. Numer. Methods Fluids 24 (1997) 671–691.
[50] J.B. Bell, P. Colella, H.M. Glaz, A second order projection method of the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[51] R.J. Leveque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (1996) 627–665.
[52] E. Aulisa, S. Manservisi, R. Scardovelli, A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking, J. Comput. Phys. 197 (2004) 555–584.
[53] G.I. Taylor, The formation of emulsions in definable fields of flow, Proc. R. Soc. Lond. Ser. A 146 (1934) 501–523.
[54] B.J. Bentley, L.G. Leal, An experimental investigation of drop deformation and breakup in steady, two-dimensional linear flow, J. Fluid Mech. 167 (1986) 241–283.
[55] Y.T. Hu, D.J. Pine, L.G. Leal, Drop deformation, breakup, and coalescence with compatibilizer, Phys. Fluids 12 (3) (2000) 484–489.
[56] M. Siegel, Cusp formation for time-evolving bubbles in two-dimensional Stokes flow, J. Fluid Mech. 412 (2000) 227–257.
[57] H.A. Stone, Dynamics of drop deformation and breakup in viscous fluids, Annu. Rev. Fluid Mech. 26 (1994) 65–102.
[58] A.J. James, J. Lowengrub, A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant, J. Comput. Phys. 201 (2004) 685–722.
[59] X. Yang, A.J. James, An arbitrary Lagrangian–Eulerian method for interfacial flows with insoluble surfactants, in preparation.