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

Xiaofeng Yang\textsuperscript{a}, Ashley J. James \textsuperscript{a,*}, John Lowengrub\textsuperscript{b}, Xiaoming Zheng\textsuperscript{b}, Vittorio Cristini\textsuperscript{c}

\textsuperscript{a}Department of Aerospace Engineering and Mechanics, University of Minnesota, 107 Akerman Hall, 110 Union St SE, Minneapolis 55455, USA
\textsuperscript{b}School of Mathematics, University of California, Irvine
\textsuperscript{c}Biomedical Engineering, University of California, Irvine

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 track both the level set function and the volume fraction. The level set function is tracked 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 field. Different from previous works, we have developed an analytic method for finding the line constant on triangular grids, which makes interface reconstruction efficiently 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. Since the numerical velocity field is in general not exactly divergence free, we conserve the volume of each single fluid phase by performing a local volume fraction correction for interfacial cells, followed by a global redistribution of any remaining volume imbalance. 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.

Key words: VOF, Level set, Interface, Unstructured grid

Preprint submitted to Journal of Computational Physics 22 April 2005
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 other to create emulsions, polymer blending, and so on. Numerical simulations of such flows have been 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 have their own advantages and disadvantages, and have 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 & Kothe [15].

In a LS method, the interface is captured by a level set function \( \phi \), 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

\[
\phi_t + \mathbf{u} \cdot \nabla \phi = 0. \quad (1)
\]

Usually, \( \phi \) is initialized as a signed distance from each grid point to the interface, and it is desirable to maintain \( \phi \) as a signed distance function as the interface evolves since, otherwise, high gradients or even a jump in \( \phi \) can develop, for example, when interfaces merge. The high gradients in \( \phi \) could result in excessive errors when, for example, derivatives of \( \phi \) are taken to find the interface normal vectors. In addition, maintaining \( \phi \) 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 \( \phi \) 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 work. A review of the LS method with an emphasis on applications in fluid interfaces was given by Sethian & Smereka [3]. See also Osher & Fedkiw [4]. The advantage of a LS method is that it handles merging and breaking of the interface automatically, the interface never has to be explicitly reconstructed, and since $\phi$ is a smooth function, the unit normal $\mathbf{n}$ and curvature $\kappa$ of the interface can be easily calculated from $\phi$ as

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

(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 $\phi$ can remain unchanged with some order of accuracy during the redistancing, the volume enclosed by the zero level sets is not conserved as $\phi$ evolves according to Eq. (1). An open problem is to find more conservative schemes to solve (1) [18].

In a VOF method, a volume fraction function $f$ is defined, which has a value of unity in one fluid and zero in the other fluid. For convenience, we call the fluid for which $f$ is unity as fluid 1, and the other fluid as fluid 2. Analytically, the interface is sharp, and $f$ is discontinuous across the interface. Numerically, $f$ is discretized, and the discrete values of $f$ are associated with the cell centers. The value of $f$ associated with a grid cell that includes an interface (called an interfacial cell) is thus between zero and unity, and is defined as the ratio of the volume of fluid 1 to the volume of the grid cell. Inversely, the interface can be determined by the discrete $f$ field, which is called interface reconstruction. The $f$ field is subject to the advection by the flow field, that is

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

(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 solve 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 & Kothe [6]. Piecewise circle [19], piecewise parabolic [20,21], and
piecewise spline \cite{22} interface reconstructions have also been developed. Two new interface reconstruction methods based on least squares fit were presented by Scardovelli \& Zaleski \cite{23}. Based on the reconstructed interface, several schemes have been developed to advect \( f \). Most of these works are based on Eulerian advectons of \( f \) on structured grids, either with a split scheme or an unsplit scheme. These kind of schemes are described by Rider \& Kothe \cite{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 (LE) 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. \cite{24} and Ashgriz et al. \cite{25}. A new mixed split Eulerian implicit - Lagrangian explicit (EI-LE) advection scheme was presented by Scardovelli \& Zaleski \cite{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 \& Kothe \cite{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 type and extent of the kernel operation may very well depend on the interfacial flow being modeled \cite{26}. A detailed comparison of several kernels and some principles on how to choose a kernel function were given by Williams et al \cite{27}.

Recently, methods that couple two different schemes have been developed. Examples are the coupled level set and volume-of-fluid method (CLSVOF) \cite{28,29}, the hybrid particle level set method \cite{30}, and the mixed markers and volume-of-fluid method \cite{31}. A coupled method takes advantage of the strengths of each of the two methods, and are therefore superior to either method alone. In a CLSVOF method, both \( \phi \) and \( f \) are tracked according to Eqs. (1) and (3), respectively. \( \phi \) is used to calculate the unit normal and curvature of the interface, \( f \), together with the unit normal calculated from \( \phi \), is used to reconstruct a new interface. Thus, \( \phi \) can be finally reset to a signed distance function based on the reconstructed interface. Volume can be conserved very accurately by tracking \( f \). Since \( \phi \) 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 \& Puckett \cite{28} are for structured grids. A few works have been done on unstructured grids \cite{24,25,32,19}. Very recently, Sussman et
al. [33,34] have developed a CLSVOF method for 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 interface changes of the interface, the 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 by Zheng et al. [35], which is based on the earlier work of Cristini et al. [36]. For the advection of the level set function, we solve Eq. (1) using a discontinuous Galerkin finite element method. For the evolution of 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. In that the numerical velocity field is not exactly divergence free, we conserve the volume of each single fluid phase by performing a local volume fraction correction for interfacial cells and followed by a global redistribution of the rest volume imbalance. The method is also coupled to a finite element based Stokes solver. We should say that 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. Sections 2 and 3 describe the advection of $\phi$ and $f$ respectively. Section 4 describes the coupling between the advection of $\phi$ and $f$. In sections 5 and 6, we describe briefly the adaptive mesh algorithm and the Stokes flow solver, respectively. Code validation and results are in section 7. A conclusion is given in section 8.

2 The Eulerian evolution of the level set function

The evolution of the level set function is achieved by solving Eq. (1) using a discontinuous Galerkin finite element method [37,38] on a fixed Eulerian grid (in contrast to the projected Lagrangian grid used in the next section). Let $P^k(T)$ be the set of all polynomials of degree at most $k$ on the triangular element $T$, and $T_h$ a triangulation of the computational domain $\Omega$ with a
characteristic grid size $h$. The finite element space is defined as

$$W_h^k = \{ v \in L_2(\Omega) : v|_T \in P^k(T), \forall T \in \mathcal{T}_h \},$$

(4)
i.e., the space of piecewise polynomials of degree $k \geq 0$ with no continuity requirements across the element boundaries.

For incompressible flow, in which case $\nabla \cdot \mathbf{u} = 0$, we can define our numerical method as finding an approximate solution $\phi_h \in W_h^k$, such that

$$\int_T \frac{\partial \phi_h}{\partial t} v - \int_T \phi_h \mathbf{u}_h \cdot \nabla v + \sum_{e \in \partial T} \int_e \mathbf{u}_h \cdot \mathbf{n} \hat{\phi}_h v^- = 0,$$

$$\forall T \in \mathcal{T}_h, \forall v \in W_h^k,$$

(5)

where $\mathbf{u}_h$ is the discrete velocity, $v^-$ denotes the value of a test function taken from within the element $T$, $\hat{\phi}_h$ is a single-valued flux through an element boundary $e$, and $\mathbf{n}$ is the outward unit normal of $T$. Note that $\mathbf{u}_h$ is continuous across any element boundaries (we employ a continuous velocity space. See section 6.), but $\phi_h$ and the test function $v$ are discontinuous across an element boundary.

There are various definitions for the flux $\hat{\phi}_h$, which can be either a central or an upwind-biased average. We define $\hat{\phi}_h$ as the complete upwind flux, i.e.,

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

(6)

where $\phi_h^-$ is the value of $\phi_h$ on the boundary $e$ taken from within the element $K$, and $\phi_h^+$ is the value of $\phi_h$ on the boundary $e$ taken from outside the element $K$.

In general $\mathbf{u}_h$, which is obtained from a flow solver, is not exactly divergence free. So in practice, we solve the following modified version of Eq. (1)

$$\phi_t + \nabla \cdot (\mathbf{u} \phi) = \phi(\nabla \cdot \mathbf{u}),$$

(7)

such that the level set function is only convected by the velocity field even when the velocity field is not divergence free. Thus, our numerical method is to find an approximate solution $\phi_h \in W_h^k$, such that

$$\int_T \frac{\partial \phi_h}{\partial t} v - \int_T \phi_h \mathbf{u}_h \cdot \nabla v + \sum_{e \in \partial T} \int_e \mathbf{u}_h \cdot \mathbf{n} \hat{\phi}_h v^- = \int_T \nabla \cdot \mathbf{u} \phi_h v,$$

(6)
In particular, we take piecewise linear test functions for the calculations in the present paper. The time integration is performed using a second order Runge-Kutta scheme. In addition, we employ the narrow band/local level set method for efficiency, i.e., we solve Eq. (8) only in the neighborhood of the interface (see also [35]).

Since the numerical level set function $\phi_h$ obtained from the above discontinuous Galerkin method is discontinuous, we project (in the $L_2$ sense) the discontinuous $\phi_h$ onto a continuous piecewise linear space for delineating the interface. Specifically, we can define the general continuous piecewise polynomial space as

$$W_h^{sk} = \{ v \in C^0(\Omega) \cap L_2(\Omega) : v|_T \in P^k(T), \forall T \in \mathcal{T}_h \}.$$  

Let $\phi_h^*$ denotes the continuous level set function obtained by the projection. Then, the $L_2$ projection of $\phi_h \in W_h^1$ to $\phi_h^* \in W_h^{s1}$ is simply the solution of the following weak formulation

$$\int_{\Omega} \phi_h^* v = \int_{\Omega} \phi_h v, \forall v \in W_h^{s1}. \quad (10)$$

In the framework of our ACLSVOF method, this continuous level set function will be used for interface normal calculations whenever an interface reconstruction or surface tension calculation is needed, and for convinience the superscript $*$ will be omitted from now on.

After the Lagrangian-Eulerian volume fraction advection, which will be performed immediately following the Eulerian level set advection, a new interface will be reconstructed on the Eulerian grid, and the level set function will finally be reset to be exactly the signed distance function to the reconstructed interface, which will be used as the new initial value for the next time advancement. Details on coupling of the level set to the volume of fluid method will be addressed in section 4.

3 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. A schematic of the method is shown in Fig. 1. During the projection stage, we project all the fluid elements
(or equivalently the grid cells) by moving the grid vertices according to the local velocity of the fluid using a Lagrangian approach. For example, in Fig. 1, by projecting fluid element $ABC$ we get the corresponding projected fluid element $A'B'C'$. Notice that for a linear velocity field triangles remain triangles. In addition, due to the incompressibility of the fluid the volume of fluid contained in the triangular element remains a constant during the projection, though the shape of the element does deform. The vertices of all projected elements form a new grid, which we call the Lagrangian grid. The original grid is called the Eulerian grid. As the shape of each element deforms, the interface position in each interfacial cell also changes. So, after projection we reconstruct the interface on the Lagrangian grid. This is called the reconstruction stage. The interface is approximated by piecewise linear segments using 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 the 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 remap each fluid element on the Lagrangian mesh back to the original Eulerian mesh (or an adapted mesh when grid is remeshed) during the remapping stage, which gives us the new volume fraction on the Eulerian grid and completes the volume fraction advection. Details of the method are as follows.

3.1 Lagrangian volume fraction projection

During the projection stage, the volume fraction is advected by a Lagrangian integration according to the local velocity of the fluid. We rewrite and solve
the volume fraction advection Eq. (3) in an integral form as

$$\frac{d}{dt} \int_{V(t)} f \, dV = 0, \quad (11)$$

where $V(t)$ denotes an arbitrary material volume, which contains the same material all the time and whose boundary is subject to the advection by the local velocity of the fluid. If we define

$$V^n_T = \int_T f(t^n) \, dV, \quad (12)$$

as the volume of fluid 1 in an element $T$ at time $t^n$, then Eq. (11) can be discretized as

$$\tilde{V}^{n+1}_T = V^n_T, \quad \forall T \in \mathcal{T}_h, \quad (13)$$

where $\tilde{V}^{n+1}_T$ denotes the volume of fluid 1 in a material element $T$ after a time step $\Delta t$ Lagrangian integration.

For incompressible flow, the volume of any arbitrary material volume does not change with time, and Eq. (13) implies that

$$\tilde{f}^{n+1}_T = f^n_T, \quad \forall T \in \mathcal{T}_h, \quad (14)$$

where $\tilde{f}^{n+1}_T$ denotes the volume fraction of a material element $T$ after a time step $\Delta t$ Lagrangian integration, and $f^n_T$ is the volume fraction in the material element $T$ at time $t^n$. In other words, the volume fraction in any element remains a constant during the Lagrangian integration.

The boundary of each element is subject to an advection by the local velocity of the fluid. Therefore, we have that

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}, \quad (15)$$

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 $T \in \mathcal{T}_h$ is equivalent to find the Lagrangian position of each vertex for every element in $\mathcal{T}_h$. This can be done by solving Eq. (15) discretely at each grid node. This gives us a new projected grid that we call the Lagrangian grid. Using a second
order Runge-Kutta scheme, we can obtain the coordinates of each grid node of the Lagrangian grid as the following

\[
\begin{align*}
\ddot{x}_i^{n+\frac{1}{2}} &= x_i^n + \frac{\Delta t}{2} u(x_i^n, t^n), \quad \text{for} \quad i = 1, \ldots, MT, \\
\dot{x}_i^{n+1} &= x_i^n + \Delta t u(x_i^{n+1/2}, t^{n+\frac{1}{2}}), \quad \text{for} \quad i = 1, \ldots, MT,
\end{align*}
\]

(16) (17)

where \(MT\) is the total number of grid nodes.

Theoretically, the volume of fluid in each element, which is treated as a material element, can be conserved exactly during the above Lagrangian projection. However, in numerical computations, the volume of fluid can not be conserved exactly, due to the following three reasons: (i) the numerical velocity field delivered from the flow solver is in general not exactly divergence free; (ii) the numerical velocity field is approximated by piecewise linear basis functions, and a triangular material element remains a triangle during a projection only for piecewise linear velocity fields; and (iii) the numerical integration of Eq. (15) is of course not exact. In other words, because of these three reasons, the volume of a material element can not be conserved exactly during the above numerical Lagrangian projection, and thus, the volume of each single fluid phase can not be conserved exactly during the projection. As we will show in section 7, the volume loss/gain due to the last two reasons is extremely small. The magnitude of the volume loss/gain due to velocity divergence depends on the quality of the flow solver.

In fixed grid Eulerian VOF methods, the divergence effect is sometimes corrected by modifying the volume fraction, i.e., by solving the modified volume fraction advection equation

\[
f_t + \nabla \cdot (f \mathbf{u}) = f \nabla \cdot \mathbf{u},
\]

(18)

such that the volume fraction advection Eq. (3) is always satisfied, and thus the volume fraction is exactly advected by the velocity field only. However, for the Lagrangian-Eulerian method, the non-conservation of each fluid phase is due to the change of the volume of each fluid element during each projection, which can not always be corrected by simply modifying the corresponding volume fraction. For example, for a cell of volume fraction 1, if the volume of the cell after projection is smaller than its original volume, the volume of fluid in this cell will always be less than its original volume since the maximum volume fraction can only be 1. If the volume of a cell after projection is greater than its original volume, then conserving the volume of fluid 1 by decreasing the volume fraction in this cell can easily result in a so called 'flotsam'. In order to conserve the volume of fluid more accurately, or even exactly, we have developed a two-step algorithm, in which the volume fraction values
in interfacial cells are modified such that the volume of each fluid phase is conserved.

First, we correct the volume fraction of all interfacial cells such that the volume of fluid 1 in each of the interfacial cells is conserved during a projection, i.e.,

\[
(f_{T}^{n+1})^* = \begin{cases} 
\frac{\tilde{f}_{T}^{n+1} V_{T}^{n}}{V_{T}}, & \forall T \in \mathcal{T}_h, \\
\tilde{f}_{T}^{n+1}, & \text{otherwise},
\end{cases}
\]

(19)

where \( \mathcal{T}_h \) is the set of all interfacial cells. When the corrected volume fraction is greater than 1 or less than 0, we set it to 1 or 0 respectively. After the first step, we compute an imbalance \( \Delta V \) in the volume of fluid 1 as

\[
\Delta V = \sum_{T \in \mathcal{T}_h} ((f_{T}^{n+1})^* V_{T}^{n+1} - f_{T}^{n} V_{T}^{n}),
\]

and the sum of volume of fluid 1 in all interfacial cells as

\[
V_0 = \sum_{T \in \mathcal{T}_h} (f_{T}^{n+1})^* V_{T}^{n+1},
\]

Then, during the second step we distribute the imbalance to the interfacial cells as the following

\[
(f_{T}^{n+1})^{**} = \begin{cases} 
(f_{T}^{n+1})^* (1 + \frac{\Delta V}{V_0}), & \forall T \in \mathcal{T}_h, \\
(f_{T}^{n+1})^*, & \text{otherwise}.
\end{cases}
\]

(20)

Again, when the corrected volume fraction is greater than 1 or less than 0, we set it to 1 or 0 respectively. Since this cutoff can result in a new lower order imbalance in volume of fluid, in general the second step can be repeated several times until \( \Delta V \) is less than some small tolerance. Our experience shows that for most cases \( \Delta V \) is reduced to machine zero by performing the second step only once. However, in very rare cases, it may be necessary to apply the second step twice or even three times. Since the total volume of fluid contained within the computational domain is conserved, conserving the volume of fluid 1 also conserves the volume of fluid 2.

It is worth pointing out that our first step correction is based on physical considerations. However, the second step is purely mathematical, while it has been used by many researchers, for example in [25]. In fact, as we will show in section 7.3, even applying just the first step of the conservation algorithm can reduce the volume error greatly and errors do not seem to accumulate. Applying the second step allows full mass conservation.
3.2 The analytic interface reconstruction

To remap the volume fraction from the Lagrangian mesh back to the Eulerian mesh and reinitialize the level set function, we need to determine explicitly where the interface is, a procedure called interface reconstruction. 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 a characteristic unit normal of the interface in this cell, \( \mathbf{x} \) is the location of a point on the interface and \( \alpha \) 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 \( \alpha \) 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 \( \alpha \) is calculated and compared to the given volume of fluid in the cell. A final \( \alpha \) 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. Each linear interface segment is denoted by its two end points. The interface reconstruction algorithm must find the coordinates of the two end points for each interface segment. For this purpose, we first need to determine the orientation of each interface segment, i.e., find the interface normal. This is done by taking the gradient of the level set function in the framework of our ACLSVOF method. 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. Since the level set function is continuous, it is easier to attain high order accuracy in taking its derivatives than in a pure VOF method. In particular, we fit a quadratic function for \( \phi \) by a least squares method over a stencil that consists of the reference cell under consideration and the cells that share at least a vertex with the reference cell, as illustrated in Fig. 2. To simplify the calculation, we define a local coordinate system for each normal calculation such that the origin of this local coordinate system is exactly the mid-point of the linear interface segment so determined from the level set function, at which the normal is located. Notice that the three values of the level set function, defined at the three vertices of each triangular cell, provides a linear interface approximation, which can be simply obtained by linearly interpolating the zero level set points on the appropriate edges. In each local coordinate system, the quadratic function has the generic form

\[
\phi = Ax^2 + Bxy + Cy^2 + Dx + Ey + F, \tag{21}
\]
Consider an arbitrary triangular cell of a given volume fraction $f$ and a line segment with a given normal $\mathbf{n}$. The vertices of the triangle are denoted by $A$, $B$, and $C$ in an arbitrary order. A line with normal $\mathbf{n}$ is drawn through an arbitrary vertex of the triangle. For example, without loss of generality we can draw such a line through vertex $A$ as shown in Fig. 3. 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. 3(a)),
(ii) the triangle is divided by the line into two parts, one on each side, (see Fig. 3(b)), and (iii) the line coincides with one of the edges of the triangle (see Fig. 3(c)). Mathematically, the applicable category can be determined by taking 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 at 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 by 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 \( |p_B| \leq |p_C| \) (for \( |p_B| > |p_C| \), we can follow the same below derivation and obtain similar results), and define \( f^* = \frac{\Delta_{ABD}}{\Delta_{ABC}} \) (see Fig. 4). Note that in all figures \( BD \) and \( EF \) are parallel to the given line segment. It is easy to show that

\[
f^* = \frac{\Delta_{ABD}}{\Delta_{ABC}} = \frac{p_B}{p_C}, \tag{22}
\]

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). The fluid configuration is shown in Fig. 4(a) and 4(b), which correspond to \( f < f^* \) and \( f > f^* \) respectively. For \( f < f^* \) (Fig. 4(a)), we have that

\[
\frac{\overrightarrow{AE}}{\overrightarrow{AB}} = \frac{\overrightarrow{AF}}{\overrightarrow{AD}} = \sqrt{\frac{\Delta_{AEF}}{\Delta_{ABD}}} = \sqrt{\frac{f}{f^*}}, \quad \text{and} \quad \overrightarrow{AD} = \frac{p_B}{p_C} \overrightarrow{AC} = f^* \overrightarrow{AC}.
\]

Thus, the coordinates of the two end points, \( E \) and \( F \), of the interface segment in the triangle \( \Delta ABC \) are

\[
x_E = x_A + \sqrt{\frac{f}{f^*}} \overrightarrow{AB}, \quad \text{and} \quad x_F = x_A + \sqrt{f^*} \overrightarrow{AC}.
\]

For \( f > f^* \) (Fig. 4(b)), we have that

\[
\frac{\overrightarrow{CE}}{\overrightarrow{CB}} = \frac{\overrightarrow{CF}}{\overrightarrow{CD}} = \sqrt{\frac{\Delta_{CEF}}{\Delta_{CBD}}} = \sqrt{\frac{1-f}{1-f^*}}, \quad \text{and} \quad \overrightarrow{CD} = (1-f^*) \overrightarrow{AC}.
\]

Thus,

\[
x_E = x_C + \sqrt{\frac{1-f}{1-f^*}} \overrightarrow{CB}, \quad \text{and} \quad x_F = x_C + \sqrt{(1-f)(1-f^*)} \overrightarrow{CA}.
\]
Fig. 4. Fluid configuration for $p_{BPC} > 0$.

If $p_B$ and $p_C$ are negative, then vertex $C$ must be in fluid 1. The configuration of fluid is shown in Fig. 4(c) and Fig. 4(d), which correspond to $f < 1 - f^*$ and $f > 1 - f^*$ respectively. Similarly, we can derive that for $f < 1 - f^*$

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

and for $f > 1 - f^*$

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

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. 5). Under this assumption, vertex $B$ must be in fluid 1 as shown in Fig. 5(a) and Fig. 5(b), which correspond to $f < f^*$ and $f > f^*$ respectively. We can show that for $f < f^*$

$$
\mathbf{x}_E = \mathbf{x}_B + \sqrt{\frac{f}{f^*}} \mathbf{BA}, \text{ and } \mathbf{x}_F = \mathbf{x}_B + \sqrt{f f^*} \mathbf{BC},
$$
(a) $f \leq f^*$  
(b) $f > f^*$

Fig. 5. Fluid configuration for $p_B < 0$ and $p_C > 0$.

(a) $p_B < 0$  
(b) $p_B > 0$

Fig. 6. Fluid configuration for $p_B \neq 0$ and $p_C = 0$.

and for $f < f^*$

$$x_E = x_C + \sqrt{\frac{1-f}{1-f^*}} \overrightarrow{CA}, \text{ and } x_F = x_C + \sqrt{(1-f)(1-f^*)} \overrightarrow{CB}.$$

For category (iii), we assume $p_B \neq 0$ and $p_C = 0$ without loss of generality. The fluid configuration is shown in Fig. 6(a) and Fig. 6(b), which correspond to $p_B < 0$ and $p_B > 0$ respectively. Similarly, we can show that for $p_B < 0$

$$x_E = x_B + \sqrt{f \overrightarrow{BA}}, \text{ and } x_F = x_B + \sqrt{f \overrightarrow{BC}},$$

and for $p_B > 0$

$$x_E = x_B + \sqrt{1-f} \overrightarrow{BA}, \text{ and } x_F = x_B + \sqrt{1-f} \overrightarrow{BC}.$$

The resulting piecewise linear interface conserves the volume of fluid exactly. The method is analytical, efficient, and extendable 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. A 3D algorithm has been formulated, which will be published separately.
Fig. 7. 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.

3.3 Volume fraction remapping

During this stage, we find the new volume fraction $f^{n+1}$ on the original Eulerian grid (or an adapted grid when the grid is adaptive) by remapping each projected fluid element onto the original grid. For example, in Fig. 1, the two fluids in the projected cell $A'B'C'$ are separated by the interface segment $E'F'$, which is obtained through the interface reconstruction process. Fluid 1 occupies the entire volume $A'B'E'F'$. By remapping, we can find the portion of fluid 1 in $A'B'E'F'$ that maps into the original cell $ABC$. Notice that the rest of fluid 1 in $A'B'E'F'$ will be mapped into the adjacent cells of $ABC$.

The remapping is done by performing polygon-polygon clippings, in which fluid elements and grid cells are treated as polygons, and each fluid element is clipped against every grid cell in the original mesh. In practice, we only need to clip a grid cell against the fluid elements which are in the vicinity of the cell, and we do not need to clip a cell against the ‘empty’ polygonal cells, which are solely occupied by fluid 2 and have zero volume fractions. These strategies can save computation dramatically and can be implemented easily even when the grid is adaptive.

Polygon-polygon clipping is a very basic element in graphics. Various clip-
Trimming algorithms, for example the two most widely-known algorithms — the Sutherland-Hodgeman algorithm [39] and the Weiler-Atherton algorithm [40], have been designed. We use the Sutherland-Hodgeman algorithm for its simplicity, and easy extension to 3D. The basic idea of this algorithm is to “trim” the subject polygon, which is subject to clipping, by each of the edges of the clip polygon. The clip polygon must be convex for this algorithm, which is satisfied in our problem because all the grid cells and fluid elements are convex. As an example, Fig. 7 illustrates the clipping of a four-sided polygonal fluid element $p_1p_2p_3p_4$ against a triangular grid cell $q_1q_2q_3$. To illustrate the “trimming” process, let us consider the clipping of $p_1p_2p_3p_4$ against the first edge $q_1q_2$ of the clip polygon $q_1q_2q_3$ for example. For convenience, we name the infinitely-long line, which is draw through the vertices $q_1$ and $q_2$, as $\mathcal{L}$, and define that a vertex $p_j$ is on the “right” side of $\mathcal{L}$ if $p_j$ is on the same side of $\mathcal{L}$ as the other vertex of $q_1q_2q_3$ is, and “left” side otherwise. Then, the trimming process can be described as in algorithm 1. The list of the vertices produced from the trimming process defines a clipped polygon, that is $r_1r_2r_3r_4$ for this example (see Fig. 7(b)). In general, the trimming process is reentered to clip the previously clipped polygon against the subsequent edges of the clip polygon, and the overall clipping completes as soon as the polygon has been trimmed by all the edges of the clip polygon.

The clipping deposits each of the projected fluid elements into one or more grid cells on the Eulerian mesh. The new volume fraction of a cell on the original mesh is then the sum of all the deposits of fluid 1 in this cell divided by its cell volume. Since the projected fluid elements are connected consecutively without any gap or intersections 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 the remapping.

### Algorithm 1 Trimming a polygon by a line.

```plaintext
for each vertex pair $p_j, p_{j+1}$ of the subject polygon do
  if both $p_j$ and $p_{j+1}$ are on the right side of $\mathcal{L}$ 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 $\mathcal{L}$ 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 the vertex $p_{j+1}$.
  else
    Do nothing.
  end if
end for
```
As seen in sections 2 and 3, the advection of level set and volume fraction are coupled. Here we summarize the advection of the level set and the volume fraction as a coupled procedure. We begin by assuming that we know the level set and volume fraction on an Eulerian mesh at time level $n$. We first advect the level set on the Eulerian mesh using the method described in section 2. The advected level set function will then be used to calculate interface normals where ever an interface reconstruction is required. After the Eulerian level set advection, we perform the Lagrangian-Eulerian volume fraction advection as described in section 3, in which an interface reconstruction is required on the Lagrangian mesh. To calculate the interface normal, which is needed by the interface reconstruction, we interpolate the advected level set function on the Eulerian mesh onto the Lagrangian mesh, and the interface normal is then calculated using the least squares fit described in section 3.2. On the completion of the Lagrangian-Eulerian volume fraction advection, we obtain the new advected volume fraction on the Eulerian mesh, which will be used as the initial value for the next time advancement. Up to this step, we have both the advected level set and volume fraction on the Eulerian mesh. However, the advected level set obtained so far is in general no longer a signed distance function to the material interface, a desired property of the level set function. Thus, we reconstruct the interface on the Eulerian mesh based on the current level set and volume fraction fields, and the level set function is finally reset to be exactly the signed distance function to the reconstructed interface, which completes the advection of the level set. To save computation, we can take the narrow band/local level set strategy of a pure level set method, that is, we only need to reset the level set functions near the interface. The obtained level set will be then used as an initial value for the next time advancement. Note that when the grid is adapted, which is only performed at the end of each time step, we need to remap the volume fraction and interpolate the level set onto the adapted grid. When the Stokes flow solver, which will be addressed in section 6, is coupled to the interface capturing method, the Stokes equations will be solved whenever the current velocity field is needed. Because the Stokes equations are time independent, they are solved as a quasi-static problem at each time instant.

5 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 by Zheng et al. [35], which is an adaptation to flat domains of the adaptive surface triangulated mesh of Cristini et al. [36]. The basic ideas of the method
are described as follows. We refer readers to [35,36] for further details.

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

\[ E = \frac{1}{2} \sum_{\text{edges}} \gamma^2, \]  

(23)

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 (23) implies that \( l \approx l_0 \), i.e., the optimal mesh is achieved by the minimization of mesh energy.

By defining the mesh energy (23), the mesh is thus analogous to a system of springs. For a fixed mesh topology, the minimization of the mesh energy (23) is thus equivalent to the relaxation of a dynamical system of massless springs with tensions \( \gamma \) 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,

\[ \dot{\mathbf{x}} = \sum_{j=1}^{N_c} \hat{e}_j \gamma_j, \]  

(24)

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

The dynamics of Eq. (24) involves a spectrum of relaxation times. Fortunately, the relaxation of slow modes is circumvented by topological mesh restructuring, which includes node addition, subtraction, and reconnection. These three operations are elementary ingredients for adaptively generating unstructured grids. Node addition or subtraction is needed when grids need to be refined or coarsened in some local region. Node reconnection is to reconnect some of the nodes such that the elements formed by these nodes become more regular, or not extremely stretched. For more specific details, see [35,36]. Topological restructuring operations provide access to mesh configurations with lower energy than accessible by mesh relaxation, and accelerate the equilibration of slow long-wave modes. 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.

6 The Stokes flow solver

To test the surface tension calculation, we couple our ACLSVOF interface capturing method to a finite element based Stokes flow solver. The governing equations for the flow solver are the Stokes equations

\begin{align}
-\nabla \cdot (\mu D(\mathbf{u})) + \nabla p &= -\sigma \kappa \delta_\Sigma \mathbf{n} = \nabla \cdot F_s, \\
\nabla \cdot \mathbf{u} &= 0,
\end{align}

with Dirichlet (for prescribed boundary flows, e.g., shear flow and no-slip flows) and/or Robin-type (free-slip) boundary conditions (see [35]), where \( D(\mathbf{u}) = \nabla \mathbf{u} + (\nabla \mathbf{u})^T \) is the rate of strain tensor, \( F_s = \sigma \delta_\Sigma (I - \mathbf{n} \otimes \mathbf{n}) \) is the capillary pressure tensor, \( \mu \) is the dynamic viscosity of the local fluid, \( p \) is the pressure, \( \sigma \) is the surface tension coefficient, and \( \delta_\Sigma = \delta(\phi) |\nabla \phi| \) is the surface delta function. Notice that the surface tension force has been incorporated in the momentum equation as a singular body force. The normals are calculated from the level set function using the least squares paraboloid fit described in section 3.2.

The Stokes equations are solved using the Mini finite element method [41], i.e., we use the finite element spaces

\begin{align}
V_h &= (M_h)^2 \oplus (B^3)^2, \\
Q_h = \{ \psi \in C^0(\Omega) \cap L^2(\Omega) : \psi|_T \in P^1(T), \quad \forall T \in \mathcal{T}_h, \quad \int_\Omega \psi = 0 \}
\end{align}

for velocity and pressure respectively, where

\begin{align}
M_h &= \{ v \in C^0(\Omega) \cap H^1_0(\Omega) : v|_T \in P^1(T), \quad \forall T \in \mathcal{T}_h \} \\
B^3 &= \{ v : v|_T \in P^3(T) \cap H^1_0(T), \quad \forall T \in \mathcal{T}_h \}.
\end{align}

An illustration of the Mini element spaces is shown in Fig. 8. Multiplying Eq. (25) by \( \mathbf{v} \in V_h \) and Eq. (26) by \( \psi \in Q_h \), we get the weak formulation

\[ \int_\Omega \mu D(\mathbf{u}) \cdot \nabla \mathbf{v} - \int_\Omega p \nabla \cdot \mathbf{v} = - \int_\Omega F_s \cdot \nabla \mathbf{v}, \quad \forall \mathbf{v} \in V_h, \]
Fig. 8. The Mini finite element spaces: • value of velocity; + value of pressure.

\[ \int_\Omega \psi \nabla \cdot \mathbf{u} = 0, \quad \forall \psi \in Q_h. \] (32)

See [35] for a derivation. The resulting linear system from the discretization is solved using an inexact Uzawa method [42]. See also [35].

7 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 by the Stokes flow solver described in section 6.

7.1 Interface reconstruction test

A circle of radius 0.25 centered at the center of the computational domain is reconstructed using our ACLSVOF interface reconstruction method. 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. For the convergence study, an \( L_1 \) error norm is defined as the summation of the difference in volume distribution between the analytic circle and the reconstructed circle over all interfacial cells. Table 1 shows the \( L_1 \) error norms for three sets of uniform unstructured triangular grids and the order of convergence, indicating that our interface reconstruction method is almost third order.

7.2 Volume tracking test

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 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 test problems: simple translation, rotation of a circular fluid body, solid body rotation of Zalesak slotted disk [43], and the single vortex problem introduced by Rider and Kothe [15], which provides a rigorous test of a volume tracking method in cases where significant interface change occurs. These test problems have been widely used recently [6,24,30]. For the convergence study, we define an $L_1$ error norm as

$$E^{L_1} = \int_{\Omega} |f^{\text{computed}}(\mathbf{x}, t) - f^{\text{exact}}(\mathbf{x}, t)| d\mathbf{x}.$$  \hspace{2cm} (33)

we also define the time averaged mass error

$$E^M = \frac{1}{T_f} \int_{t=0}^{T_f} \frac{|M(t) - M(0)|}{T_f} dt,$$  \hspace{2cm} (34)

where

$$M(t) = \int_{\Omega} f(\mathbf{x}, t) d\mathbf{x},$$  \hspace{2cm} (35)

and $T_f$ is the total integration time. Even though the velocity field is given analytically, fluid mass is not exactly conserved because the second order Runge-Kutta integration can result in errors in Lagrangian grids and the velocity field
is approximated using piecewise linear basis functions. In order to show the mass error inherent in the volume tracking algorithm, we do not use the mass conservation method proposed in section 3 here. As shown in Tables 2-7, the mass errors are extremely small relative to the $L_1$ error norms as a consequence of the second order discretization of our coupled volume tracking method and the fact that the velocity field is analytically divergence free. For all test problems in this section, except the solid body rotation of Zalesak disk, we use a fixed CFL number of 0.7, and the computational domain is a $(0, 1) \times (0, 1)$ square. Convergence studies are performed for both fixed uniform grids and adaptive clustered grids. The fixed uniform grids consist of almost regular triangles, whose edge lengths are roughly 0.032, 0.016, and 0.008, respectively for the three sets of uniform grids. For all adaptive grids, grids are clustered near the interface to render high resolutions near the interface. The edge lengths of the smallest grid cells, denoted by $h_{\text{min}}$, 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.

7.2.1 Simple translation

An initially circular fluid body of radius 0.25 centered at the center of the computational domain is translated by a uniform solenoidal velocity field along the $45^\circ$ 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, allowing the error to be quantified with Eq. (33).

Fig. 9 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 toward 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. 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
Fig. 9. Reconstructed interfaces (left) and the corresponding $L_1$ error contours (right) of an initially circular fluid body after a complete diagonal translation. $h_{\text{min}} \approx 0.032, 0.016$, and $0.008$, respectively from top to the bottom. The contours are of 20 levels from $-3.1 \times 10^{-6}$ to $2.7 \times 10^{-6}$ for (b), from $-4.0 \times 10^{-7}$ to $2.9 \times 10^{-7}$ for (d), and from $-9.2 \times 10^{-8}$ to $7.2 \times 10^{-8}$ for (f).

function exactly.
Table 2

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

<table>
<thead>
<tr>
<th>$h_{\text{min}}$</th>
<th>Triangles</th>
<th>$L_1$ error</th>
<th>order</th>
<th>Mass error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.032</td>
<td>2048</td>
<td>7.10E-5</td>
<td>-</td>
<td>0.0</td>
</tr>
<tr>
<td>0.016</td>
<td>8192</td>
<td>1.67E-5</td>
<td>2.09</td>
<td>0.0</td>
</tr>
<tr>
<td>0.008</td>
<td>32768</td>
<td>3.61E-6</td>
<td>2.21</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Adaptive grids:

<table>
<thead>
<tr>
<th>$h_{\text{min}}$</th>
<th>Triangles</th>
<th>$L_1$ error</th>
<th>order</th>
<th>Mass error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.032</td>
<td>764</td>
<td>9.14E-5</td>
<td>-</td>
<td>0.0</td>
</tr>
<tr>
<td>0.016</td>
<td>1722</td>
<td>1.80E-5</td>
<td>2.34</td>
<td>0.0</td>
</tr>
<tr>
<td>0.008</td>
<td>3590</td>
<td>5.07E-6</td>
<td>1.83</td>
<td>0.0</td>
</tr>
</tbody>
</table>

7.2.2 Solid body rotation

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 at 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 interface change, allowing the error to be quantified with Eq. (33).

Fig. 10 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, and because they are much lower than the $L_1$ error norms, the mass errors do not nullify our convergence study. The results verify that our volume tracking method conserves mass very accurately for this test even without using the mass conservation method proposed in section 3.

7.2.3 Zalesak slotted disk rotation

Zalesak slotted disk [43] (refer to Fig. 11) 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 [44] of this problem, in
Fig. 10. Reconstructed interfaces (left) and the corresponding $L_1$ error contours (right) of an initially circular fluid body after a complete revolution. $h_{\text{min}} \approx 0.032$, $0.016$, and $0.008$, respectively from top to the bottom. The contours are of 20 levels from $-1.9 \times 10^{-5}$ to $1.9 \times 10^{-5}$ for (b), from $-2.6 \times 10^{-6}$ to $2.1 \times 10^{-6}$ for (d), and from $-3.0 \times 10^{-7}$ to $4.0 \times 10^{-7}$ for (f).

which the specifications of the geometry are slightly modified. 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 such that the magnitude of the linear velocity at the mid-points of the domain edges is 1. Thus, the disk
Table 3
$L_1$ error norms, convergence rates and time averaged mass errors for the solid body rotation test

<table>
<thead>
<tr>
<th>$h_{min}$</th>
<th>Triangles</th>
<th>$L_1$ error</th>
<th>order</th>
<th>Mass error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.032</td>
<td>2048</td>
<td>5.61E-4</td>
<td>-</td>
<td>2.59E-7</td>
</tr>
<tr>
<td>0.016</td>
<td>8192</td>
<td>1.28E-4</td>
<td>2.13</td>
<td>3.75E-8</td>
</tr>
<tr>
<td>0.008</td>
<td>32768</td>
<td>3.48E-5</td>
<td>1.88</td>
<td>4.57E-9</td>
</tr>
</tbody>
</table>

Adaptive grids:

<table>
<thead>
<tr>
<th>$h_{min}$</th>
<th>Triangles</th>
<th>$L_1$ error</th>
<th>order</th>
<th>Mass error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.032</td>
<td>523</td>
<td>4.09E-4</td>
<td>-</td>
<td>1.20E-6</td>
</tr>
<tr>
<td>0.016</td>
<td>1107</td>
<td>1.01E-4</td>
<td>2.02</td>
<td>2.45E-7</td>
</tr>
<tr>
<td>0.008</td>
<td>2246</td>
<td>2.54E-5</td>
<td>1.99</td>
<td>2.16E-8</td>
</tr>
</tbody>
</table>

Fig. 11. Zalesak slotted disk after one complete revolution. Left – the reconstructed disk (solid line) and the exact solution (dashed line); Right – the reconstructed disk with the computational grids.

should return to its initial position after $4\pi$ time units without any interface change.

In most previous works, uniform structured grids, with 200 cells along each coordinate direction, were used. To compare our results to these previous works, we refine our adaptive grids near the interface such that the edge length of the smallest cells are roughly 0.02, and as in [44], the time step is chosen such that a complete revolution is fulfilled with 2524 time steps, corresponding to a Courant number of about 0.25. The errors for this test are specially computed
Table 4  
$E^Z$ errors for Zalesak slotted disk rotation test. All previous results are taken from Table 3 in [45]. We remind readers that our result was obtained using adaptive triangular grids, while all previous results were obtained on uniform rectangular grids.

<table>
<thead>
<tr>
<th>Advection/reconstruction methods</th>
<th>$E^Z$ error</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLIC</td>
<td>8.38E-2</td>
</tr>
<tr>
<td>Hirt-Nichols</td>
<td>9.62E-2</td>
</tr>
<tr>
<td>FCT-VOF</td>
<td>3.29E-2</td>
</tr>
<tr>
<td>present ACLSVOF</td>
<td>1.25E-2</td>
</tr>
<tr>
<td>Youngs</td>
<td>1.09E-2</td>
</tr>
<tr>
<td>Harvie &amp; Fletcher (Stream/Youngs)</td>
<td>1.07E-2</td>
</tr>
<tr>
<td>Harvie &amp; Fletcher (Stream/Puchett)</td>
<td>1.00E-2</td>
</tr>
<tr>
<td>López et al. (EMFP-AIR)</td>
<td>8.74E-3</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (linear least-square fit)</td>
<td>9.42E-3</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (quadratic fit)</td>
<td>5.47E-3</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (quadratic fit + continuity)</td>
<td>4.16E-3</td>
</tr>
</tbody>
</table>


\[
E^Z = \frac{\sum_T |f^T_{\text{computed}} - f^T_{\text{exact}}|}{\sum_T f^T_{\text{exact}}}.
\]  

Fig. 11 shows the reconstructed slotted disk after a complete revolution. 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 [44,45,23]. However, our result looks more symmetric than other similar results, and is comparable with the result obtained using quadratic interface reconstructions in [23]. The computational grids shown on the right in Fig. 11 illustrate that the grids are refined near the interface.

The $E^Z$ error calculated using Eq. (36), and a rough comparison to those presented by other researchers are tabulated in Table 4. We remind readers that this comparison is not strict because our result was obtained using adaptive triangular grids, while all previous results were obtained on uniform rectangular grids. The time average mass error for this test is $8.64 E - 9$, which is extremely small.
7.2.4 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 velocity is given by the stream function

\[ \Psi = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y), \]

(37)

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 [46], 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 & Kothe in [15]. The flow field contains a single vortex centered at the center of 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. 12 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. 12 shows that the thin filament breaks into small pieces beginning at its tail after long time integration. 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. This unphysical pinch-off can be delayed by refining the grid near the interface, though it seems impractical to have continuous resolution of the filament, which becomes infinitely thin eventually. 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 [47] is applied to the velocity field, i.e., the velocity defined by (37) 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. (33). Convergence studies are performed on uniform grids and adaptive grids for three different values of \(T_f\). Fig. 13 shows the reconstructed interfaces and the corresponding \(L_1\) error contours on the adaptive grid of \(h_{\text{min}} = 0.016\)
Fig. 12. Evolution of an initially circular fluid body placed in the single vortex flow field.
Fig. 13. 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 the adaptive grid of $h_{\text{min}} = 0.016$ 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 $-5.9 \times 10^{-7}$ to $6.9 \times 10^{-7}$ for (b), from $-2.0 \times 10^{-5}$ to $1.3 \times 10^{-5}$ for (d), and from $-9.2 \times 10^{-5}$ to $8.8 \times 10^{-5}$ for (f).

for the three values of $T_f$. Quantitative error measurements and convergence rates are shown in Tables 5 - 7. The results show that our ACLSVOF interface tracking method is second order, even when $T_f$ is large, for which large complex topological change has occurred (see Fig. 14), indicating that our method is remarkably resilient for interface tracking. Again the mass errors are very small.
Fig. 14. Reconstructed interface of an initially circular fluid body placed in a
time-reversed, single vortex flow on the adaptive grid of $h_{\text{min}} = 0.016$ at $t = T_f/2$,
for $T_f = 8.0$.

Table 5
$L_1$ error norms, convergence rates and time averaged mass errors for the time-
reversed, single vortex test with $T_f = 0.5$

<table>
<thead>
<tr>
<th>Uniform grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>0.032</td>
<td>2048</td>
<td>1.26E-4</td>
<td>-</td>
<td>7.64E-6</td>
</tr>
<tr>
<td>0.016</td>
<td>8192</td>
<td>2.41E-5</td>
<td>2.39</td>
<td>3.17E-6</td>
</tr>
<tr>
<td>0.008</td>
<td>32768</td>
<td>4.30E-6</td>
<td>2.49</td>
<td>3.09E-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptive grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>0.032</td>
<td>523</td>
<td>1.97E-4</td>
<td>-</td>
<td>1.86E-5</td>
</tr>
<tr>
<td>0.016</td>
<td>1107</td>
<td>2.58E-5</td>
<td>2.93</td>
<td>3.01E-6</td>
</tr>
<tr>
<td>0.008</td>
<td>2246</td>
<td>5.24E-6</td>
<td>2.30</td>
<td>2.76E-7</td>
</tr>
</tbody>
</table>

relative to the $L_1$ error norms, and thus do not nullify our convergence study.

In Table 8, we compare our results with those presented by other researchers.
Again, we remind readers that this comparison is not strict because our results
are obtained using adaptive unstructured triangular grids, while the other
results were obtained on structured rectangular grids. However, we tried to
make the minimum edge lengths of the triangles to be roughly equal to the grid
sizes of the corresponding rectangular grids. The comparison shows that our
adaptive method is very competitive with the other state-of-the-art methods.
Table 6
$L_1$ error norms, convergence rates and time averaged mass errors for the time-reversed, single vortex test with $T_f = 2.0$

<table>
<thead>
<tr>
<th>Uniform grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>0.032</td>
<td>2048</td>
<td>1.41E-3</td>
<td>-</td>
<td>2.53E-5</td>
</tr>
<tr>
<td>0.016</td>
<td>8192</td>
<td>3.38E-4</td>
<td>2.06</td>
<td>5.25E-6</td>
</tr>
<tr>
<td>0.008</td>
<td>32768</td>
<td>6.84E-5</td>
<td>2.30</td>
<td>4.86E-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptive grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>0.032</td>
<td>523</td>
<td>1.62E-3</td>
<td>-</td>
<td>3.40E-5</td>
</tr>
<tr>
<td>0.016</td>
<td>1107</td>
<td>3.36E-4</td>
<td>2.27</td>
<td>6.45E-6</td>
</tr>
<tr>
<td>0.008</td>
<td>2246</td>
<td>7.95E-5</td>
<td>2.08</td>
<td>3.79E-7</td>
</tr>
</tbody>
</table>

Table 7
$L_1$ error norms, convergence rates and time averaged mass errors for the time-reversed, single vortex test with $T_f = 8.0$

<table>
<thead>
<tr>
<th>Uniform grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>0.032</td>
<td>2048</td>
<td>3.70E-2</td>
<td>-</td>
<td>7.01E-5</td>
</tr>
<tr>
<td>0.016</td>
<td>8192</td>
<td>3.75E-3</td>
<td>3.30</td>
<td>7.66E-6</td>
</tr>
<tr>
<td>0.008</td>
<td>32768</td>
<td>5.61E-4</td>
<td>2.74</td>
<td>1.45E-6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptive grids:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{\text{min}}$</td>
<td>Triangles</td>
<td>$L_1$ error</td>
<td>order</td>
<td>Mass error</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>0.032</td>
<td>523</td>
<td>3.54E-2</td>
<td>-</td>
<td>7.57E-5</td>
</tr>
<tr>
<td>0.016</td>
<td>1107</td>
<td>3.56E-3</td>
<td>3.31</td>
<td>1.41E-5</td>
</tr>
<tr>
<td>0.008</td>
<td>2246</td>
<td>5.09E-4</td>
<td>2.81</td>
<td>1.30E-6</td>
</tr>
</tbody>
</table>

7.3 Simulation of drop deformation in an extensional flow

In this section, we simulate a physical problem by coupling our adaptive CLSVOF interface tracking method with a Stokes flow solver, that is, the interface will be tracked by our adaptive CLSVOF methods 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 [48]. A schematic of
Table 8
$L_1$ error norms, and convergence rates obtained by different methods for the time-reversed, single vortex test with $T_f = 2.0$. All previous results are taken from Table 5 in [45]. We remind readers that our results are obtained using adaptive triangular grids, while all previous results were obtained on uniform rectangular grids.

<table>
<thead>
<tr>
<th>Grid size</th>
<th>$L_1$ error</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garrioch &amp; Baliga (Pilliod-Puckett advection/circle fit)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>2.23E-3</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>4.93E-4</td>
<td>2.18</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>1.12E-4</td>
<td>2.14</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (linear least-square fit)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>1.75E-3</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>4.66E-4</td>
<td>1.91</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>1.02E-4</td>
<td>2.19</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (quadratic fit)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>1.88E-3</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>4.42E-4</td>
<td>2.08</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>9.36E-5</td>
<td>2.24</td>
</tr>
<tr>
<td>Scardovelli &amp; Zaleski (quadratic fit + continuity)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>1.09E-3</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>2.80E-4</td>
<td>1.96</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>5.72E-5</td>
<td>2.29</td>
</tr>
<tr>
<td>Aulisa et al. (Hybrid markers/VOF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>1.00E-3</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>2.69E-4</td>
<td>1.89</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>5.47E-5</td>
<td>2.30</td>
</tr>
<tr>
<td>López et al. (EMFPA-SIR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$32 \times 32$</td>
<td>8.62E-4</td>
<td>-</td>
</tr>
<tr>
<td>$64 \times 64$</td>
<td>2.37E-4</td>
<td>1.86</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>5.62E-5</td>
<td>2.08</td>
</tr>
</tbody>
</table>

Present ACLSVVOF method:

$h_{\text{min}} = 0.032$  1.62E-3  -
$h_{\text{min}} = 0.016$  3.36E-4  2.27
$h_{\text{min}} = 0.008$  7.95E-5  2.08

35
the problem is shown in Fig. 15.

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, separation of liquid phases by tip streaming, and so on. It has attained wide investigation in the past and is still under intensive research [48-51]. A milestone, since the pioneering work by Taylor [48], is the experimental work by Bentley & Leal [49]. 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 [52].

We limit our simulation to two-dimensional low-speed Newtonian viscous flow with clean interfaces (i.e., no surfactant). Two dimensionless parameters determine the behavior of such a flow: the capillary number $Ca$ and the viscosity ratio $\lambda$, where $Ca = \mu Ga/\sigma$, $\mu$ is the viscosity of the suspending fluid, $G$ is the applied strain rate, $a$ is the radius of the undeformed drop, and $\lambda$ is the ratio of the drop to the suspending fluid viscosity. According to Bentley & Leal [49], 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 & Leal [49]. 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. [50]. Drop deformation under different $Ca$ will be 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. The first-step correction of the mass conservation algorithm proposed in section 3 is applied.

Fig. 16 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. [50]. It is shown that our numerical results agree with the experimental results very well in a wide range of \(Ca\). However, discrepancies occur for \(Ca\) near the critical value. In particular, our numerically predicted critical \(Ca\) is higher than the experimental critical \(Ca\) and so is the critical drop deformation. Also, for \(Ca\) near the critical value, our numerical predicted drop deformation is lower than the experimental value. There are several reasons for this error. First, our simulation is in two dimensions, i.e., the initial drop is a cylinder, while experimentally the initial drop is a real three dimensional sphere. Second, the flow strength in our simulation is a constant from the beginning of each computation, while in the experiment the flow strength increases gradually from a minimum practical value, i.e., the drop evolution under a certain \(Ca\) in fact begins with a steady state solution corresponding to a slightly smaller \(Ca\). It has been shown that “drop burst depends not only on the instantaneous flow conditions, but also on the entire time history of the velocity gradient experienced by the drop” [49]. This influence is probably large on determining the critical \(Ca\). Third, since drop deformation is very sensitive to the flow condition near the critical \(Ca\) (see Fig. 16), there is difficulty in accurately determining the critical deformation both numerically
Fig. 17. Drop breakup 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.

Fig. 18. Adaptive grids for drop breakup in an extensional flow ($\lambda = 0.1$, $Ca = 0.23$, and $h_{\text{min}} \approx 0.04$).

and experimentally [49].

When $Ca$ is greater than the critical value, the drop evolves into thin filament with rounded ends and concave sides. Fig. 17 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 & Leal for the intermediate-viscosity-ratio drops [49] and the experimental result by Hu et al. [50]. Note that in the experiment by Hu et al. [50] the external flow was stopped after the drop evolved into a thin filament, which resulted in immediate drop breakup. Fig. 18 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.

To verify the effectiveness of the mass conservation algorithm we proposed in section 3, we examined the relative mass error of the drop fluid, defined as $(M(t) - M(0))/M(0)$, for all test cases with and without applying the mass conservation algorithm. For example, Fig. 19 shows the relative mass errors for a test case of $\lambda = 0.1$ and $Ca = 0.15$, an intermediate value in our study. The result shows that even applying just the first step of the correction can improve the volume conservation greatly and errors do not seem to accumulate. In general, the number of interfacial cells is comparable to or even greater than
Fig. 19. Relative mass errors for drop deformation in an extensional flow, \( \lambda = 0.1, Ca = 0.15 \).

the number of interior cells, and the velocity is least accurate near the interface. Therefore, only applying the first step can greatly improve mass conservation. Again, the mass error is primarily due to the non-zero divergence velocity field, and the first-step correction is based on physical considerations. With full correction, we conserve mass completely.

8 Conclusions and future work

We have developed an adaptive coupled level set/volume-of-fluid volume tracking method for triangular unstructured grids. The method takes advantage of the strengths of both level set and volume-of-fluid methods, which makes the calculation of interface normal, curvature, and thus surface tension easier and more accurate, and mass is conserved very accurately. For interface reconstruction, we have developed an analytic interface reconstruction algorithm, which is not only efficient but also conserves mass exactly. Numerical validation has shown that our ACLSVOF volume tracking method is at least second order accurate. We have also coupled the volume tracking method to a finite element based Stokes flow solver. In that the numerical velocity is not exactly divergence free, we have developed a two-step mass conservation algorithm, which enables us to conserve mass better, or even exactly. Physical simulations of drop deformation in an extensional flow show that our method can capture the surface tension force accurately, and is capable of analyzing a physical problem. In addition, owing to the adaptive grid algorithm, we can 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 have also been developing a piecewise parabolic interface reconstruction method, which will improve the approximation, and thus the evolution of the interface. Based on the two dimensional codes, we will develop axisymmetric and three dimensional codes. For the 3D case, for example, the Stokes or Navier-Stokes flow solver and the discontinuous Galerkin level set evolution are already mature. Concerning the Lagrangian-Eulerian volume fraction advection, the extension of our 2D analytic interface reconstruction algorithm to 3D and the 3D polyhedron clipping will present some challenges. Fortunately, we have already derived a solution for the 3D analytic interface reconstruction, and the 3D polyhedron clipping has been developed by previous researchers. Finally, we will incorporate the effects of surfactants. Surfactants play a crucial role in drop deformation and breakup, especially in producing very tiny droplets by tipstreaming. Implementations of these ingredients are underway.

References


43