Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 224 (2007) 150–167
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction

Rajkeshar Singh [a], Wei Shyy [b,*]

[a] Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA
[b] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

Received 1 September 2006; received in revised form 18 November 2006; accepted 22 December 2006
Available online 24 January 2007

Abstract

Multiphase flows associated with interfacial dynamics, steep jumps in fluid properties and moving boundaries between different phases pose substantial computational challenges in terms of both modeling as well as computational efficiency. The present work extends a marker-based immersed boundary, or front tracking, technique to model the three-dimensional interfacial dynamics. It tracks the moving boundary using triangulated surface grids and solves the flow governing equations on a stationary Cartesian grid. A locally adaptive grid is employed to help meet the resolution requirements based on the interface location and solution features. The interface resolution is controlled via a conservative restructuring technique satisfying mass continuity. An improved level contour reconstruction algorithm for topology change, preserving the interface connectivity information, is presented highlighting various algorithmic difficulties and implemented remedies. The outlines of a finite-volume, staggered grid Navier–Stokes solution using the projection method are discussed. The impact of conservative interface restructuring and reconstruction has been assessed against mass-conservation and spurious velocity errors. The overall capabilities of the developed algorithms have been demonstrated for large density ratios, O(1000), interfacial flows using various rising bubbles and drop collision/coalescence computations involving coalescence and break-up dynamics.

Keywords: Immersed boundary; Adaptive Cartesian grid; Interface tracking; Staggered grid

1. Introduction

Multiphase flow computations offer a wide variety of challenges due to presence of multiple length and time scales concerning both accuracy as well as computational efficiency. Such flows are characterized by the presence of an interface that evolves with the flow and exhibits complex shapes and topological changes. Besides the challenges in numerical modeling of the interfacial effects such as the treatment of discontinuities in the fluid and flow properties, accurate interface tracking itself is a difficult and challenging issue. In Refs. [1,2],

* Corresponding author. Tel.: +1 734 936 0102; fax: +1 734 936 0106.
E-mail addresses: rks@ufl.edu (R. Singh), weishyy@umich.edu (W. Shyy).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.12.026

background information on the interfacial dynamics and moving boundary computations is reviewed. Overall, there are three main categories of interface tracking methods: Lagrangian; Eulerian; mixed Eulerian–Lagrangian. The Lagrangian methods adopt a body fitted grid approach with the interface as a boundary and are, in general, the most difficult choice for three-dimensional computations. As opposed to that, Eulerian methods such as volume of fluid and level set methods are among the most convenient techniques. Details of these Eulerian methods and recent advances addressing issues related to mass-conservation and geometry-computation can be found in [3–6]. The marker-based front tracking approach [7–17] falls in the Eulerian–Lagrangian category. It tracks the interface using a set of markers moving passively on a stationary background, where all the flow computations are performed.

A traditional front-tracking method tracks three-dimensional interfaces using triangulated surfaces. The most serious drawback of this approach is often attributed to the algorithmic complexity, especially in maintaining the interface connectivity information when topology changes occur. Some of the recent algorithmic developments in literature have been geared towards simplifying the approach by eliminating the need for explicit connectivity information. The connectivity-free point-set method of Torres and Brackbill [12] and connectivity-free tracking of triangular elements by Shin and Juric [13] are some of the key efforts in this direction. Of these, the method of Shin and Juric [13], using connectivity-free tracking and level-contour-based interface reconstruction, is an algorithmically convenient and attractive option. However, owing to the lack of connectivity information, the method in [13] has to reconstruct the interface for not just topology change but also the interface resolution control. In this regard, the connectivity information allows greater flexibility in dealing with flows with multiple interfaces in close proximity. An example is bubbly flows, such as those simulated by Esmaeeli and Tryggvason [14], where O(100) bubbles move in close proximity without merging with each other. In such flows, a connectivity free tracking may introduce undesirable mergers while performing the level-contour-based reconstruction to control the interface resolution.

The present work uses the Eulerian–Lagrangian approach: interface is tracked using triangulated surface with explicit connectivity information; flow-equations are solved on a background Cartesian grid. The interfacial dynamics is modeled using the immersed boundary method [16]. The level-contour-based reconstruction is used only to perform topology changes. Interface tracking with triangulated surfaces and level-contour-based reconstruction has also been reported in [15]. In the present paper, the reconstruction algorithm and the various associated difficulties are reviewed, and remedies for maintaining valid interface connectivity data are proposed, implemented, and tested. It is demonstrated that the approach developed is robust and offers satisfactory accuracy [8]. Furthermore, the effects of interface reconstruction on mass-conservation and spurious velocity currents are also addressed. The call for reconstruction has been automated using a simple probe-based technique to detect the instances of topology change. In addition, the mass-conserving interface smoothing approach of Sousa et al. [17] has been extended to handle marker addition and deletion; the outcome is improved control of the interface resolution. The interface tracking algorithms can be used and adapted for a variety of interfacial flows.

The implementation of marker-based methods is more involved compared to Eulerian methods; however, it is capable of resolving features of substantial geometric complexities satisfactorily. Another issue addressed in the present study is balancing the cost and accuracy of the computation. Since resolution requirements can quickly make three-dimensional computations cost-prohibitive, an adaptive Cartesian grid method, utilizing the interface location and solution features, has been developed. Regarding the flow solver, a staggered grid, finite volume formulation is used for the incompressible flow computation using a projection method. In order to simplify the staggered grid algorithm, constraints on the velocity and pressure field adopted by Losasso et al. [19] are incorporated.

In short, the objective of the present effort is development of an integrated three-dimensional computational capability for interfacial flows involving topology changes. The following aspects represent the specific improvements and contributions of our work relative to those reported in the literature:

- Interface tracking using triangulated surface grids and development of a mass-conserving technique to control the interface resolution;
- Detection of topology change via a simple probe-based technique;

- Subsequent interface reconstruction via level-contours techniques, along with detailed account of algorithmic-complexity and mass-conservation;
- Development of a dynamically adaptive Cartesian grid technique with staggered grid computation for two-phase flows with disparate length scale variations.

It should be noted that compared to Eulerian methods, the present approach, utilizing markers and enforcing mass-conservation, does add computational cost. However, in our view, it strives to reach a compromise between accuracy and computational cost. Alternatively, the sharp interface method [20,21], while tending to offer higher order of accuracy, is substantially more time consuming, as well as more demanding in data-structure. While finite difference approaches [22] can be substantially more convenient than finite volume approaches [21], the conservation laws can not be readily guaranteed in the interfacial region. Furthermore, the adaptive grid technique substantially reduces the CPU time while maintaining desirable accuracy.

2. Immersed boundary method

The immersed boundary method [16] uses a single-fluid formulation for the entire domain. The key components of the method include smoothed treatment of the surface tension force (Eq. (1)) and fluid property jumps. As shown in Eq. (2), the fluid properties are smoothed using an indicator function I that varies smoothly from zero (inside interface) to one (outside interface) within 4–5 cells across the interface. The smoothed density is defined as a linear function of the indicator function while, based on the tangential stress continuity condition, the viscosity is computed using linear variation of the inverse of kinematic viscosity [23]. The subscripts 1 and 2 denote the properties of fluid outside and inside the interface, respectively

$$\nabla \cdot u = 0$$

$$\rho\left(\frac{\partial u}{\partial t} + \nabla \cdot (uu)\right) = -\nabla p + \nabla \cdot \mu(\nabla u + \nabla^T u) + \rho g + \int_S \sigma \kappa n \delta \, \mathrm{d}s \tag{1}$$

$$\rho = \rho_2 + (\rho_1 - \rho_2) I$$

$$\frac{\rho}{\mu} = \frac{\rho_2}{\mu_2} + \left(\frac{\rho_1}{\mu_1} - \frac{\rho_2}{\mu_2}\right) I \tag{2}$$

This indicator function is computed using the Poisson equation (Eq. (3)), where the integration on the right hand side is performed over the interface and the terms n and $\mathrm{d}A$ are interface normal vector and area element. The one-dimensional form of Dirac-delta function ($\delta_{(x-x\_\mathrm{interface})}$) is computed using Eq. (4) [24] and it is plotted in Fig. 1. The term D represents the discretized form of the three-dimensional delta function.

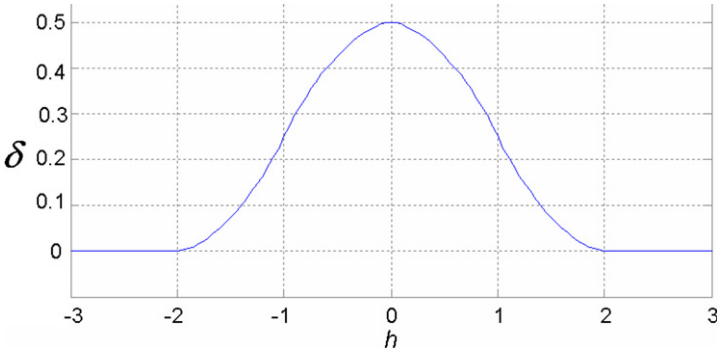

Fig. 1. One dimensional delta function with two-cell support.

$$\nabla^2 I(x) = \nabla \cdot \left(\int_{\text{interface}} n\delta_{(x-x_{\text{interface}})}\, \mathrm{d}A \right) \tag{3}$$

$$D(r) = \frac{\delta(r_x/\Delta x)\delta(r_y/\Delta y)\delta(r_z/\Delta z)}{\Delta x \Delta y \Delta z}$$

$$\delta(h) = \begin{cases} \frac{1}{8}\left(3 - 2|h| + \sqrt{1 + 4|h| - 4h^2}\right) & \text{if } |h| \in [0,1) \\ \frac{1}{8}\left(5 - 2|h| + \sqrt{-7 + 12|h| - 4h^2}\right) & \text{if } |h| \in (1,2) \\ 0 & \text{if } |h| \in [2,\infty) \end{cases} \tag{4}$$

3. Interface tracking

The markers (triangle nodes in Fig. 2(a)) are advected using a delta-function based interpolation of the velocity field. Markers are locally added and deleted at every time step to preserve the spacing between ($\Delta/3$, Δ), where Δ is the background cell-size. Any volume-error in the interface, introduced by the marker advection, is recovered explicitly by perturbing the markers in local normal direction. These errors are typically very small (e.g. O(10^{-5})%) but, if uncorrected, they may accumulate to become significant.

3.1. Conservative interface restructuring

A typical marker addition and deletion procedure breaks a long edge at its midpoint (Fig. 2(b)), and deletes a short edge by collapsing it to the mid-point (Fig. 3(a)) [7,17]. This procedure introduces volume errors that, although small, can accumulate for long duration computations [17]. Such errors are easily avoidable by introduction of a simple step in the edge-deletion algorithm. Although interface volume errors are corrected explicitly after each time step, the purpose of this conservative restructuring is to minimize the magnitude of this artificial correction by avoiding volume-errors due to marker addition/deletion. Also, since the interface is represented by linear elements, addition or deletion of elements can always produce local changes in the interface curvature. However, the effect of such perturbations is mitigated by the stabilizing effect of viscous forces and smoothed treatment of the surface tension forces.

Since the marker addition at the edge-midpoint does not produce any volume-error, the marker-addition procedure is same as shown in Fig. 2(b). The marker deletion procedure has a volume-correction step that is performed after collapsing the edge to its midpoint. Fig. 3(a) shows marker removal from a two-dimensional interface by collapsing the edge p_2p_4 to the mid-point p_3, resulting in a net interface-area loss. To correct this error, first a reference point p_{ref} is defined at $p_3 - n$ along the local normal vector in Fig. 3(a). A reference area $p_1p_2p_4p_5p_{\text{ref}}p_1$ is computed and point p_3, after marker deletion, is relocated (Fig. 3(b)) along the normal vector to set the new area $p_1p_3p_5p_{\text{ref}}p_1$ same as the reference area. The equivalent steps in three dimensions are summarized as:

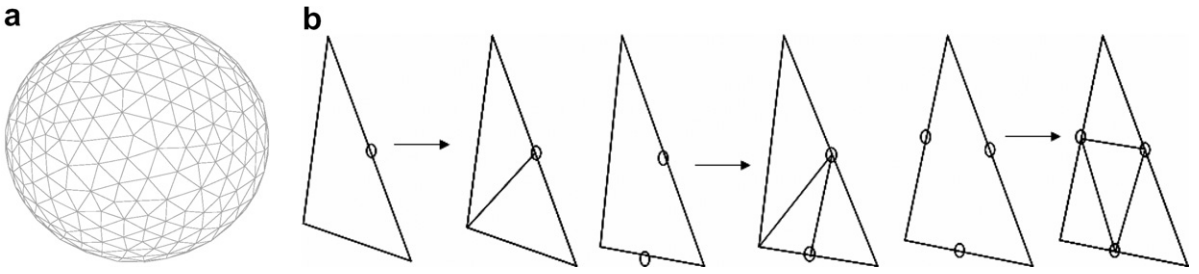

Fig. 2. Interface tracking and marker addition: (a) a triangulated interface; (b) edge splitting at the midpoint to introduce new markers.

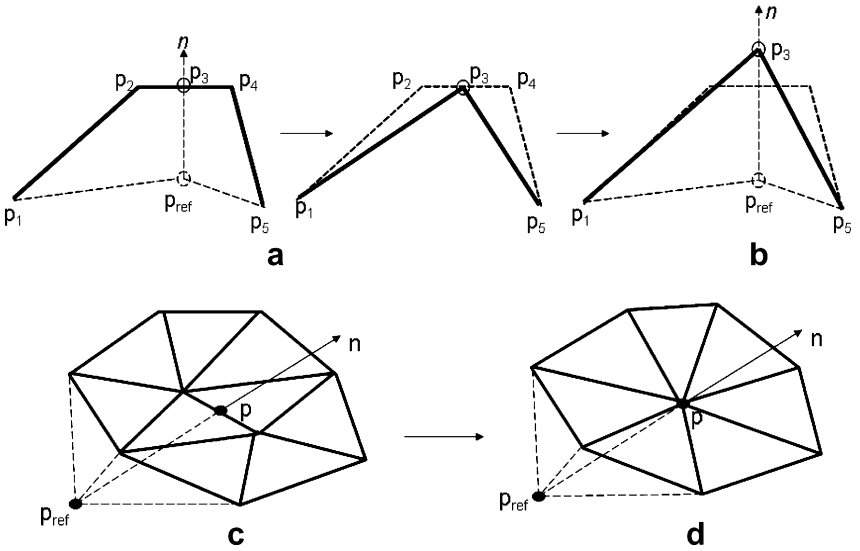

Fig. 3. Marker/edge deletion: (a) a 2D edge deleted by collapsing it to the midpoint p_3; (b) p_3 is relocated to preserve the local interface area made with reference point p_{ref}; (c) a triangle edge to be collapsed to its midpoint p; (d) point p after edge deletion is relocated along the local normal direction.

1. Select point p at the middle of the edge to be deleted and compute the local normal vector n (Fig. 3(c)).
2. Define a reference point p_{ref} at $p - n$ and a reference volume Φ_{ref} by adding the volumes of tetrahedrons made by point p_{ref} and all the triangles in Fig. 3(c).
3. Collapse the edge to point p and update interface connectivity information.
4. Compute the new volume Φ made by p_{ref} and the triangles of Fig. 3(d). Recover the reference volume by relocating p to $p_{ref} + (\Phi_{ref}/\Phi)n$.

3.2. Interface reconstruction for topology change

The level-contour-based reconstruction technique of Shin and Juric [13] is used to handle topology changes by recreating the interface at indicator = 0.5 contour (Fig. 4(b)). Defects in reconstructed interface-volume are explicitly corrected by perturbing the markers in local normal direction. In case of multiple reconstructed interfaces, the magnitude of the volume-correction applied to an interface is inversely proportional to its

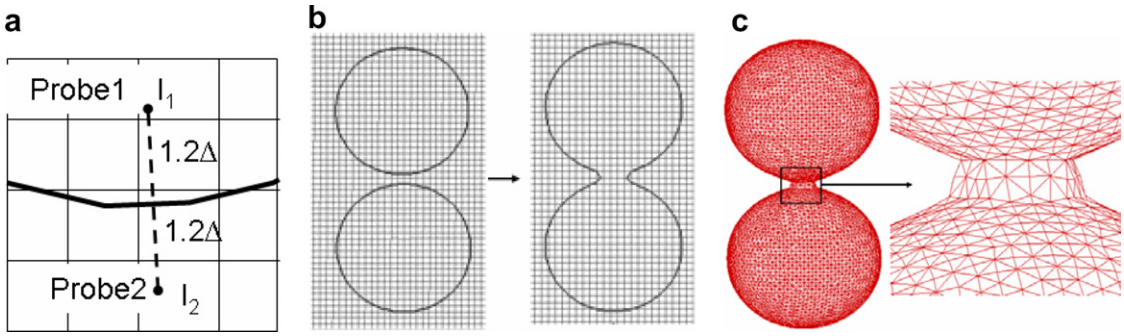

Fig. 4. Interface reconstruction: (a) reconstruction is performed if the both the indicator values (I_1 and I_2) on the two probes are either smaller or greater than half; (b) cross-section of two three-dimensional interfaces in close proximity, and the corresponding indicator = 0.5 contour; (c) reconstructed interface.

volume, so that more correction is applied to larger interfaces. Typical volume errors and the magnitudes of subsequent correction, relative to the background grid resolution, are estimated in the results section.

The instances of interface merger or breakup are detected using two probes placed in opposite directions along the local normal vector on markers, as shown in Fig. 4(a). The indicator function on these probes is bilinearly interpolated from the grid. The reconstruction is performed if both the probes from a marker have indicator function value greater or less than half, representing a situation where the immersed boundary formulation does not 'see' an interface (when approximated by indicator = 0.5) between the two probes. In case of multiple interfaces, the reconstruction is performed only for those requesting it. The probe-based method is a simple and computationally efficient criterion, performed every few time-steps (every 10 steps in the presented computations). An example of a reconstructed interface is shown in Fig. 4.

The reconstruction algorithm first creates a set of globally numbered markers at *Indicator* = 0.5 on the Cartesian cell edges. Establishing a valid triangulated surface grid-data using this cloud of markers may require tedious checks with interface-segments created in neighboring cells. A set of simple criteria is used to simplify the algorithm and avoid data-duplicity. Using the indicator function value, the algorithm flags the Cartesian cell vertices as:

$$\text{vertex\_flag} = \begin{cases} 1: & \text{indicator} > 0.5(\text{outside interface}) \\ 0: & \text{otherwise}(\text{inside/on the interface}) \end{cases} \tag{5}$$

A simple rule of associating a marker to an edge is used, stating that an edge can have a marker only if the two vertices have opposite flags. Similarly, a cell 'contains' a marker only if at least one of its edges contains a marker. Using the vertex flag, it can be seen that a cell will always contain at least three markers or none. All the cells 'containing' a marker are visited and markers associated with the cell are collected to define a convex polygon. This procedure simplifies the algorithm by eliminating the need to look into the reconstructed interface-segments in neighboring cells. As an example, consider Fig. 5(a) where an edge e_2 is constructed only while visiting cell_1, by considering the markers on its vertical edges. Although the markers/vertices of edge e_2 may physically lie on the edges of cell_2 as well, cell_2 does not 'see' any marker as none of its edges have opposite vertex-flags. Although this approach facilitates a simple cell-by-cell reconstruction approach, it allows creation of edges/elements with zero length/area. Such a situation arises when, for example, the reconstructed markers p_1, p_2 and p_3 in Fig. 5(b) have the same coordinates i.e. indicator = 0.5 contour is at the Cartesian-cell vertex. However, zero length/area elements do not pose any difficulty in establishing a valid connectivity-data and are easily deleted using the marker-deletion procedure.

Occasionally, two or more disconnected polygons may be noticed in a cell (e.g. Fig. 6(a)). Presence of more than six markers in a 3D Cartesian cell produces some obvious algorithmic difficulties in terms of how to define the interface polygons using the cloud of markers on the Cartesian edges. As shown in Fig. 6(a), eight

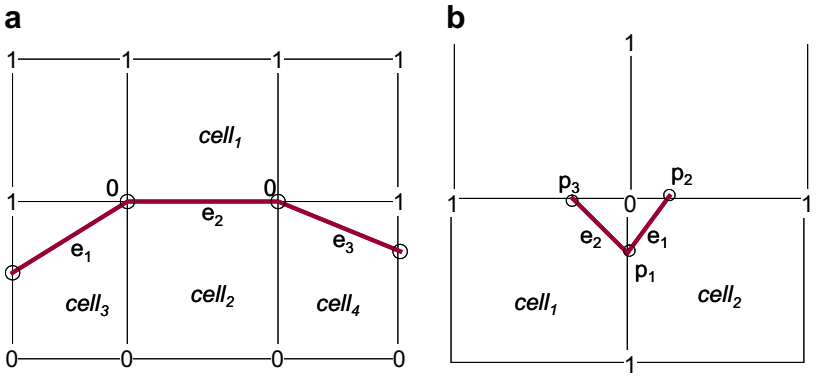

Fig. 5. Illustration of a 2D cell-by-cell reconstruction: (a) Markers on edges with opposite vertex-flags (0 and 1) are connected to form edges e_1, e_2, and e_3 in cells cell_3, cell_1 and cell_4, respectively; (b) markers p_1, p_2 and p_3 are connected in cell_1 and cell_2 irrespective of their physical coordinates.

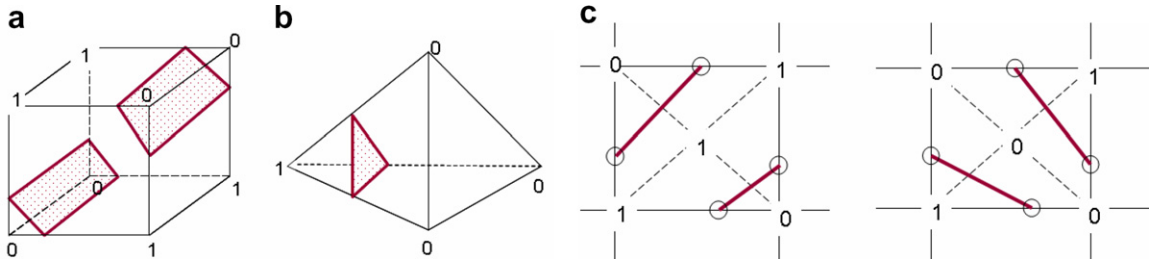

Fig. 6. Cell-by-cell interface reconstruction: (a) a cell with eight markers and one way to define two polygons; (b) a tetrahedral cell can contain at the most four markers and hence only one surface segment; (c) two ways of connecting four markers based on the vertex flags of the triangulated cell-edges.

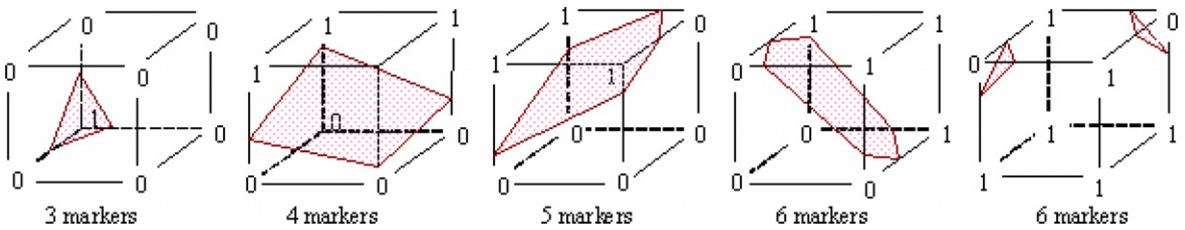

Fig. 7. Typical polygons reconstructed inside Cartesian cells shown with corresponding vertex flags.

markers can be connected in at least two ways to define two interface segments. Ambiguities with multiple polygons can be easily avoided by performing the reconstruction in tetrahedral cells that are temporarily created by breaking the Cartesian cells around the interface. A tetrahedral cell, using the vertex flag information, can have only three or four markers, and hence only one polygon (Fig. 6(b)). Equivalently, in 2D, the vertex-flags of triangulated cells can be used to decide the connectivity within a cell, as shown in Fig. 6(c). The present work creates markers only on Cartesian cell edges. The possibility of markers on tetrahedral cell-edges is used only to decide the marker-connectivity within the Cartesian cell. Since the indicator function itself is smooth in nature, polygons in Fig. 7 are the most frequently observed topologies. The non-triangular polygons are broken into triangles by inserting a marker at the center.

4. Adaptive grid for flow computation

A dynamically adaptive Cartesian grid, with the interface location and solution features as the adaptation criteria, is employed to compute the flow equations. To avoid errors due to non-uniform cells near the interface, few layers of cells ($\sim$10) across the interface are kept at maximum desired resolution. The underlying grid data is stored in an unstructured format [25] allowing fast access to the connectivity-queries issued by the flow-solver. Outline of the grid generation technique and the flow computation are described in the following sections. A comprehensive account of the grid generation and the solution-algorithms can be found in [8].

4.1. Grid generation

The computational domain is uniformly divided into a prescribed number of cells in each coordinate direction. This base grid is locally refined to prescribed levels, keeping eight to ten layers of cells across the interface at maximum resolution. Various stages of grid generation, starting from a base grid and recursively refining for desired interface resolution, is shown in Fig. 8. For demonstration, only up to four layers of cells across the interface in Fig. 8 were kept at uniform refinement-level. The refinement is isotropic in nature and cells sharing a face are not allowed to differ by more than one level. For simplification of the solution algorithm, same restriction on the cell-size is also placed on the corner cells.

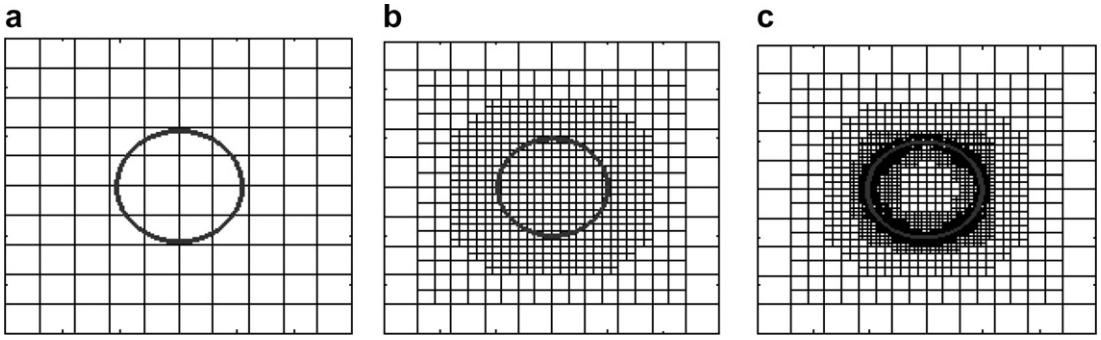

Fig. 8. Grid generation process: (a) 10×10 base grid; (b) grid after two successive levels of adaptation; (c) grid after four levels of refinement.

Since the grid within eight to ten layers across the interface is at maximum resolution, it avoids non-uniform cells in the critical areas where fluid properties undergo fast variations. As the computation progresses, a solution-based adaptation is also performed in regions away from the interface. Although more sophisticated error indicators for solution-based adaptation can be employed from literature, the present work uses a simple velocity-curl-based criterion [26] that works well for viscous flows. It computes a cell-parameter ξ and the standard deviation S_d, using Eq. (6). The terms l and N_{cell} represent the length scale (cube root of cell-volume) and the total number of Cartesian cells, respectively. The decision to refine or coarsen a cell is made by using the criteria shown in Eq. (7).

$$\begin{aligned} \zeta_{\text{cell}} &= |\nabla \otimes u| l^{3/2} \\ S_d &= \sqrt{\frac{1}{N_{\text{cell}}} \sum_{i=1,N_{\text{cell}}} \zeta_i^2} \end{aligned} \tag{6}$$

$$\begin{aligned} &\zeta_{\text{cell}} > S_d \Rightarrow \text{Refine cell} \\ &\zeta_{\text{cell}} < 0.1 S_d \Rightarrow \text{Coarsen cell} \end{aligned} \tag{7}$$

4.2. Navier–Stokes computation

A projection method is used for temporal integration of the incompressible-flow equations, where an intermediate velocity field u^* is computed and projected into a divergence-free space (Eq. (8)). The pressure Poisson equation for the velocity correction is solved using conjugate gradient method. The convection term is treated using the 2nd order Adam–Bashforth or Runge–Kutta method. The diffusion terms are discretized using Crank–Nicholson technique. The surface tension and gravitational terms are treated implicitly.

$$\begin{aligned} u^* &= u^n + \Delta t \frac{-\nabla \cdot uu - \nabla p^n + \nabla \cdot \tau + F^{n+1}_{\text{source}}}{\rho^{n+1}} \\ u^{n+1} &= u^* - \Delta t \frac{\nabla (p^{n+1} - p^n)}{\rho^{n+1}} \Rightarrow \nabla \cdot \left(\frac{\nabla \delta p}{\rho^{n+1}} \right) = \frac{\nabla \cdot u^*}{\Delta t} \end{aligned} \tag{8}$$

The staggered grid arrangement stores the face-normal velocity component on Cartesian faces and other variables (pressure, density, viscosity, etc.) are stored at the cell centers (Fig. 9(a)) [18]. The velocity-constraint of Losasso et al. [19] is used to simplify the spatial discretization on non-uniform cells. With this constraint, the fine faces (face_1 and face_2) in Fig. 9(b) see the same velocity field ($U_1 = U_2$). In order to apply the same amount of velocity correction on face_1 and face_2, the corresponding face-normal pressure gradients are also kept the same (Eq. (9) and Fig. 9(b))

$$P_{x1} = P_{x2} = \frac{(P_1 + P_2) - 2P_3}{2d} \tag{9}$$

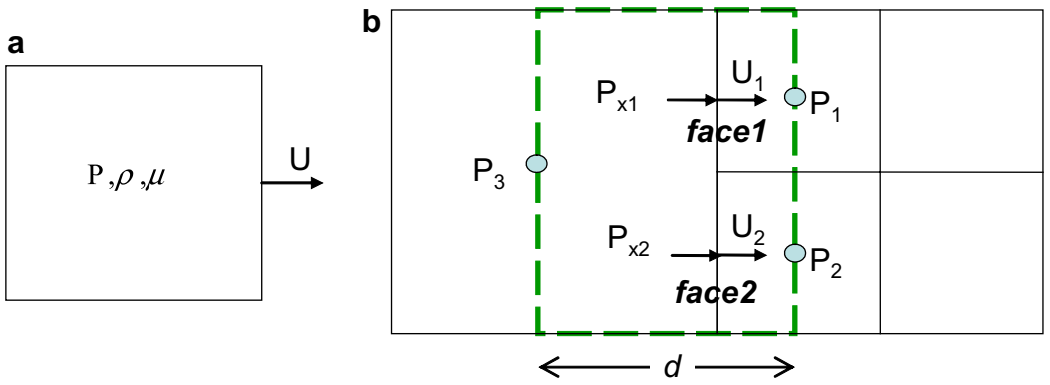

Fig. 9. Staggered grid discretization: (a) grid arrangement; (b) velocities and pressure gradients on fine faces, face$_1$ and face$_2$, of a coarse cell are constrained as $U_1 = U_2$ and $P_{x1} = P_{x2}$ *i.e.* face$_1$ and face$_2$ have the same control volume defined by the dashed line.

4.2.1. Spatial discretization of convection term

The convection term in finite volume form, integrated over the control-volume faces, can be represented as

$$\int_{CV} \nabla \cdot (\phi U)\,\mathrm{d}V = \sum \phi U \cdot n\,\mathrm{d}A = \sum \phi U_{\mathrm{cf}}\,\mathrm{d}A \tag{10}$$

The normal velocities are computed appropriately to maintain the divergence free condition within each control-volume i.e. $\sum U_{\mathrm{cf}}\,\mathrm{d}A = 0$. As an example, the control-volume face-normal velocities (U_{cf}) in Fig. 10(a) are computed, using area-weighted averaging, as:

$$\begin{aligned} U_{\mathrm{cf1}} &= 0.5(U_1 + U_2) \\ U_{\mathrm{cf2}} &= 0.5(U_1 + U_3) \\ U_{\mathrm{cf3}} &= 0.5(U_1 + U_4) \\ U_{\mathrm{cf4}} &= \frac{(U_5 \mathrm{Area}_1 + U_6 \mathrm{Area}_2)}{\mathrm{Area}_1 + \mathrm{Area}_2} \end{aligned} \tag{11}$$

The momentum flux ϕ on control-volume faces are computed using linear interpolation of the face-normal velocities. To conserve the momentum flux across non-uniform grid cells, Fig. 10(b) represents another approximation used for dealing with T-node junctions. It computes the flux through the top face of the control-volume for velocity U_1, and distributes it equally to the control-volume for velocities U_2 and U_3.

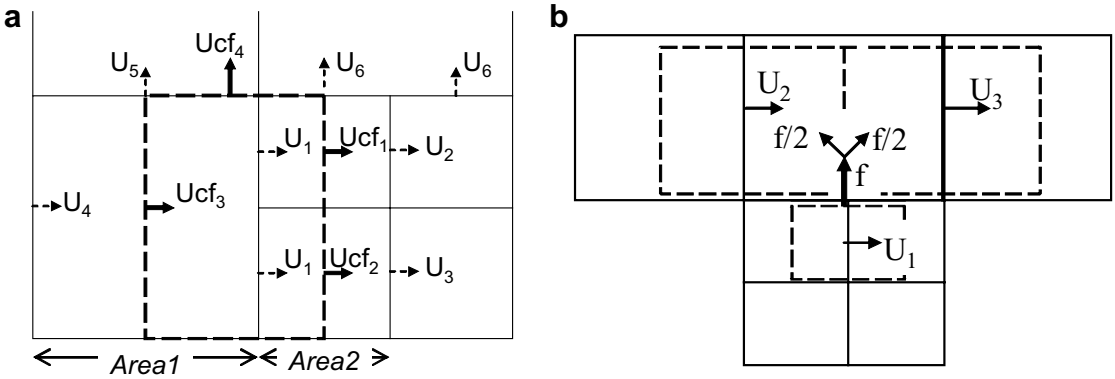

Fig. 10. Spatial discretization of convection term: (a) computation of control-volume face-normal velocities (U_{cf}); (b) flux f computed on the top-face of control volume for velocity U_1 is split equally ($f/2$) between control volume for velocities U_2 and U_3.

4.2.2. Spatial discretization of viscous term

The momentum equation with source terms due to surface tension and gravity lumped into (S_u,S_v,S_w), can be represented as

$$\begin{aligned}\rho\frac{Du}{Dt} &= -\frac{\partial p}{\partial x} + S_u + \nabla\cdot(\mu\nabla u) + \nabla\cdot\mu\frac{\partial(u,v,w)}{\partial x}\\ \rho\frac{Dv}{Dt} &= -\frac{\partial p}{\partial y} + S_v + \nabla\cdot(\mu\nabla v) + \nabla\cdot\mu\frac{\partial(u,v,w)}{\partial y}\\ \rho\frac{Dw}{Dt} &= -\frac{\partial p}{\partial z} + S_w + \nabla\cdot(\mu\nabla w) + \nabla\cdot\mu\frac{\partial(u,v,w)}{\partial z}\end{aligned} \tag{12}$$

The last and the penultimate viscous terms in the above equations come from $\nabla\cdot\mu(\nabla^{\mathrm{T}}u)$ and $\nabla\cdot\mu(\nabla u)$, respectively. The discretization of $\nabla\cdot\mu(\nabla u)$ requires computation of velocity gradients normal to the control-volume-faces, while the terms from $\nabla\cdot\mu(\nabla^{\mathrm{T}}u)$ require computation of mixed derivatives. Discretization of $\nabla\cdot\mu(\nabla u)$ term follows the strategy outlined by Fig. 10. The terms due to $\nabla\cdot\mu(\nabla^{\mathrm{T}}u)$ are added only around the interface, where the viscosity is spatially varying, producing an algorithmically convenient discretization on the locally uniform grid in the vicinity of the interface. Owing to the incompressibility condition and constant viscosity in most of the domain, the contributions due to $\nabla\cdot\mu(\nabla^{\mathrm{T}}u)$ are neglected in cells away from the interface.

5. Results and discussion

In the following sections, we will present selected case studies highlighting the current approach for handling various features associated with interfacial fluid flows. Specific cases, characterizing the conservative interface restructuring and reconstruction, are presented to address the accuracy and mass-conservation issues.

The overall capabilities of the developed algorithms and techniques are demonstrated via a select set of rising bubble and drop collision computations. Numerous dimensionless parameters can be identified for such complex flow problems. The dimensionless parameters used in the present computations are listed in Table 1. The spurious velocity computations, characterizing the effect of interface reconstruction, use the Capillary number to measure the maximum dimensionless velocity for a chosen Laplace number as the input parameter.

The rising bubble computations with single and multiple bubbles use Eötvos number and Morton number (Table 1) as the dimensionless parameters. The fluid properties outside the interface are used as the reference. The terms d, $\Delta\rho$ denote the bubble diameter and difference in fluid densities, respectively. The rise velocities are presented in terms of Reynolds number.

Collision of two equal size drops uses the Weber number and Reynolds number as the input parameters. The impact velocity (Fig. 16(a)) is used as the reference velocity scale. The properties of fluid inside the drop are used as the reference. The eccentricity of collision is defined using the impact parameter defined in Table 1.

Table 1
Key dimensionless parameters

Computed case	Dimensionless parameters
Spurious velocity computation	Capillary number ($Ca = \mu U_{\max}/\sigma$) Laplace number ($La = \rho\sigma d/\mu^2$)
Rising bubble computations	Eötvos number ($Eo = g\Delta\rho\ d^2/\sigma$) Morton number ($M = g\mu^4\Delta\rho/\rho^2\sigma^3$) Reynolds number ($Re = \rho U d/\mu$)
Binary drop collision	Impact parameter ($B = h/d$) (Fig. 16(a)) Weber number ($We = \rho U^2 d/\sigma$) Reynolds number ($Re = \rho U d/\mu$)

5.1. Interface in a time-reversed vortex field: effect of conservative restructuring

A spherical interface is placed in a time-reversed vortex field [5] to test the effect of conservative restructuring, and it's accuracy is compared with a non-conservative approach that deletes edges by collapsing them to the midpoint. Due to the time-reversed nature of the imposed flow, the spherical interface deforms severely and returns back to the original shape, albeit with some numerical errors. These errors are due to marker advection using the interpolated velocity field, and interface restructuring. The computations are performed using a sphere of diameter 0.15 placed at $(0.5, 0.75, 0.5)$ inside a unit cube centered at $(0.5, 0.5, 0.5)$ (Fig. 11(a)). The time reversed vortex field with period $T = 4$ in Eq. (13) is imposed in the computational domain. The shape history computed for a $60 \times 60 \times 60$ grid is shown in Fig. 11(b).

The time history of the interface-volume error and its grid convergence characteristics are presented in Fig. 12. For this test, no explicit interface-volume correction was applied and the phase volume error with conservative restructuring is solely due to marker advection. The conservative method was observed to perform considerably well showing a better than quadratic convergence with grid refinement.

$$\begin{aligned} u(x,y,z) &= \cos(\pi t/T)\sin^2(\pi x)(\sin(2\pi z) - \sin(2\pi y)) \\ v(x,y,z) &= \cos(\pi t/T)\sin^2(\pi y)(\sin(2\pi x) - \sin(2\pi z)) \\ w(x,y,z) &= \cos(\pi t/T)\sin^2(\pi z)(\sin(2\pi y) - \sin(2\pi x)) \end{aligned} \tag{13}$$

5.2. Effect of interface reconstruction

Since the Indicator $= 0.5$ contour is only an approximation of the actual interface, the reconstructed interface has a slightly different volume than the interface before reconstruction. This section characterizes the magnitude of such losses and the required corrections.

Since the reconstruction perturbs the interface, its effect on spurious currents for static bubble computation were also observed, suggesting that the cumulative effect of reconstruction for a given computation depends on the reconstruction frequency and its relation with the time scales of the flow.

5.2.1. Effect on mass-conservation

A spherical interface placed in a Cartesian grid was reconstructed for different background grid resolutions. The volume losses with varying number of grid-cells per interface-diameter are presented in Table 3, exhibiting the expected [13] quadratic grid-convergence rate. The volume errors (ΔV) produced by reconstruction are corrected explicitly by perturbing the markers in local normal direction. With the original surface area and interface volume denoted by A_0 and V_0, the approximate radial perturbation, required for the volume correction, can be estimated using Eq. (14). This correction term exhibits a quadratic convergence rate as shown in Table 2 (the grid cell-size Δ in Table 2 reduces by a factor of two with increasing resolution). Considering that

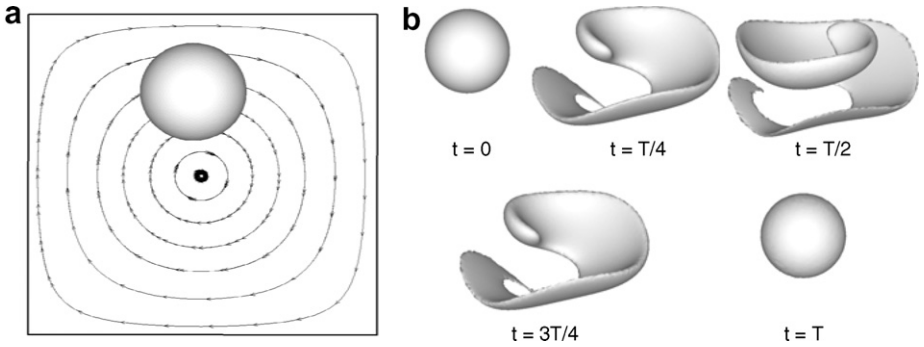

Fig. 11. Interface in a time reversed vortex field test: (a) a spherical interface placed in a time reversed vortex field; (b) computed shape history for period $T = 4$.

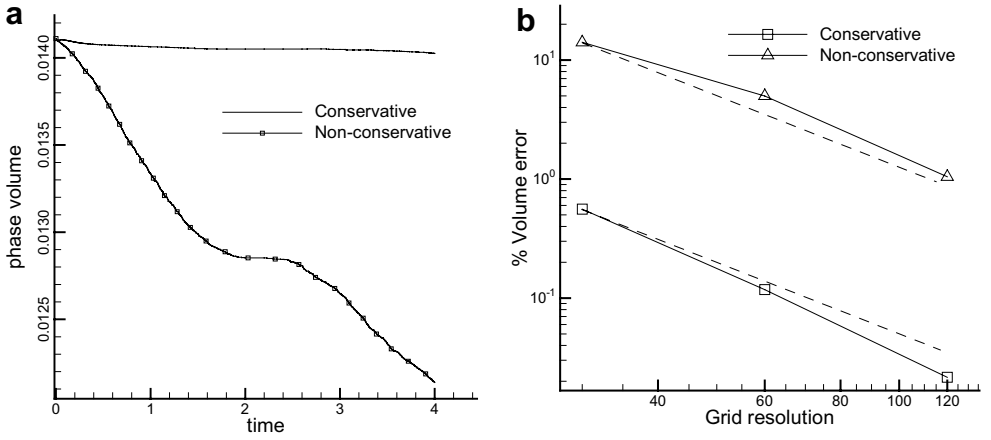

Fig. 12. Volume error for interface in a time reversed vortex field: (a) volume error on a $30 \times 30 \times 30$ grid; (b) final ($t = T$) volume error for different grid resolutions ($30 \times 30 \times 30$, $60 \times 60 \times 60$ and $120 \times 120 \times 120$) where the dotted line shows a quadratic convergence rate.

Table 2
Effect of reconstruction on mass conservation and the magnitude of radial perturbation required to correct it

Grid cell per diameter $N = d/\Delta$	% Volume defect without explicit correction $100\lvert\Delta V/V_0\rvert$	Approximate radial perturbation $\lvert\Delta r\rvert \approx (\lvert\Delta V\rvert N/6V_0)\Delta$
10	−7.60	$1.27 \times 10^{-1}\Delta$
20	−1.99	$6.63 \times 10^{-2}\Delta$
40	−0.49	$3.25 \times 10^{-2}\Delta$
80	−0.12	$1.64 \times 10^{-2}\Delta$

the fluid properties are smeared over four cells across the interface, the amount of radial disturbance caused by even moderately resolved grids ($N = 20$) is found to be reasonably small.

$$|\Delta r| \approx \frac{|\Delta V|}{A_0} = \frac{|\Delta V|}{4\pi r_0^2} = \frac{|\Delta V|}{V_0}\frac{r_0}{3} = \frac{|\Delta V|}{V_0}\frac{N}{6}\Delta \tag{14}$$

5.2.2. Effect on spurious currents

Spurious velocities or parasitic currents, clearly seen in static bubble simulations, are unphysical velocity fields created due to numerical errors caused by imbalance of interfacial stresses. The maximum spurious velocity in terms of Capillary number was found to be $O(10^{-4})$ for the tested Laplace numbers in the range from 250 to 12000. These magnitudes are consistent with the observations in literature [7,17].

The effect of interface reconstruction on spurious velocity currents was qualitatively the same as reported in [13] where temporary spikes in spurious velocities are observed after each reconstruction. Using Fig. 13, it can be observed that frequent reconstruction based on the Capillary time scale ($d\mu/\sigma$) can produce excessive disturbances that do not get sufficient time to damp-out between successive reconstructions, causing an order of magnitude larger spurious velocity error. In order to minimize such disturbances, reconstruction is performed sparingly and selectively: only for interfaces that indicate a potential topology change in the probe-based-test described earlier.

5.3. Buoyancy driven rising bubbles

The fluid density and viscosity ratios used for rising bubble computations are set to 100. The following sections present several computations for rising bubbles, including an off-axis coalescence of two rising bubbles, to establish the accuracy of the present algorithms.

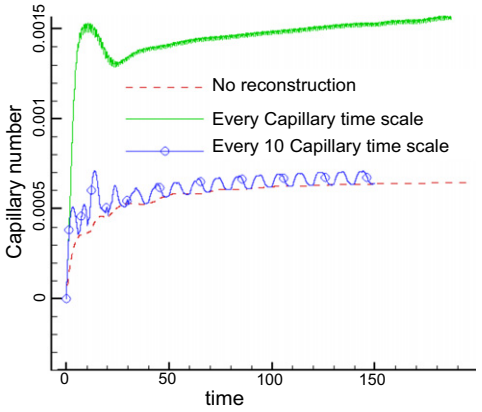

Fig. 13. Effect of reconstruction frequency on spurious velocity currents compared with the case of no reconstruction (time non-dimensionalized using $t_{\text{capillary}} = d\mu/\sigma$ scale).

5.3.1. Single rising bubble

The computations were performed in a domain of size $= (5d, 10d, 5d)$ with three levels of refinement over $12 \times 24 \times 12$ base grid. An open boundary condition was specified on the top boundary (zero velocity gradients and constant pressure) and slip condition was specified elsewhere. The computed terminal shapes and streamlines are shown in Fig. 14. The terminal rise-Reynolds number for various shape regimes are presented in Table 3, showing an excellent agreement with the estimates from the diagram of Clift et al. [27] (Fig. 14(e)) and the volume-of-fluid computations of Annaland et al. [28].

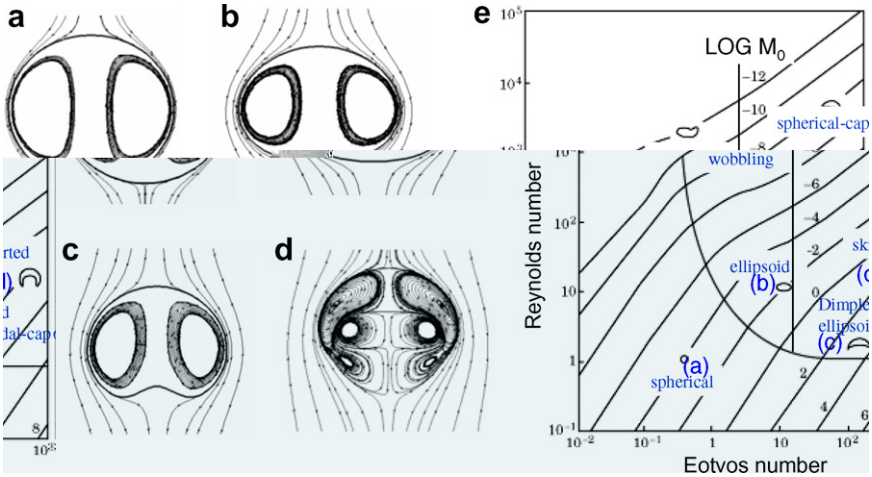

Fig. 14. Computed shapes and streamlines for cases in Table 4 falling in (a) spherical, (b) ellipsoidal, (c) dimpled ellipsoidal and (d) skirted regimes; (e) shape diagram of Clift et al. [27].

Table 3
Computed terminal rise Reynolds numbers in different terminal shape-regimes

M	Eo	Terminal shape regime (Fig. 14(e))	Rise Reynolds number		
			From [27]	From [28]	Computed
1.26×10^{-3}	0.971	Spherical	1.7	1.6	1.6
1.00×10^{-1}	9.71	Ellipsoidal	4.6	4.3	4.6
1.00×10^{-3}	97.1	Dimpled ellipsoidal	1.5	1.7	1.6
9.71×10^{-1}	97.1	Skirted	20	18.0	18.7

5.3.2. Off-axis coalescence of two bubbles

The fluid properties and computational set-up used for the computation are the same as in the volume-of-fluid computations of Annaland et al. [28] shown in Table 4. The Morton and Eötvos numbers are set to $M = 2 \times 10^{-4}$ and $Eo = 16$ falling between the skirted and spherical cap regimes in Fig. 14(e). The top bubble-center in Fig. 15(a) is at $(0, 0, 0)$, located at a distance of $2.5d$ from the bottom of the computational domain. The initial position of bottom bubble-center is at $(0.8d, -1.5d, 0.0)$. The computed bubble shape history along with a cross-section of the grid in x–y-plane is shown in Fig. 15(a). The computed instantaneous shapes are in good agreement with the computations in [28] and capture the qualitative behavior exhibited in the experiments of Brereton and Korotney [29] (Fig. 15(c)).

The coalescence process involves the bottom bubble moving into the wake of leading bubble and accelerating to cause eventual coalescence. The dimensionless x-coordinates of the top and bottom bubbles, along with the rise-Reynolds number before coalescence (Fig. 15(b)), show that the top bubble remains practically unaffected by the presence of the trailing bubble. The rise-Reynolds number of the leading bubble is roughly between 45 and 50 (47.8 at $t = 0.07$ s), consistent with the observations in literature: a value of 50 from the diagram of Clift et al. [27] for a single rising bubble; experimental and computed values of 43 and 40, respectively, reported by Annaland et al. [28].

Table 4
Computational parameters for off-axis coalescence of two rising bubbles

Individual bubble diameters (d)	0.01 m
Computational domain ($4d, 8d, 4d$)	(0.04, 0.08, 0.04) m
Grid: Three levels of refinement on $10 \times 20 \times 10$ base grid	Max. resolution $= 80 \times 160 \times 80$
Fluid properties outside interface (ρ_1, μ_1)	1000 kg/m^3, 0.1 kg/ms
Fluid inside interface (ρ_2, μ_2)	10 kg/m^3, 0.001 kg/ms
Surface tension (σ)	0.1 N/m

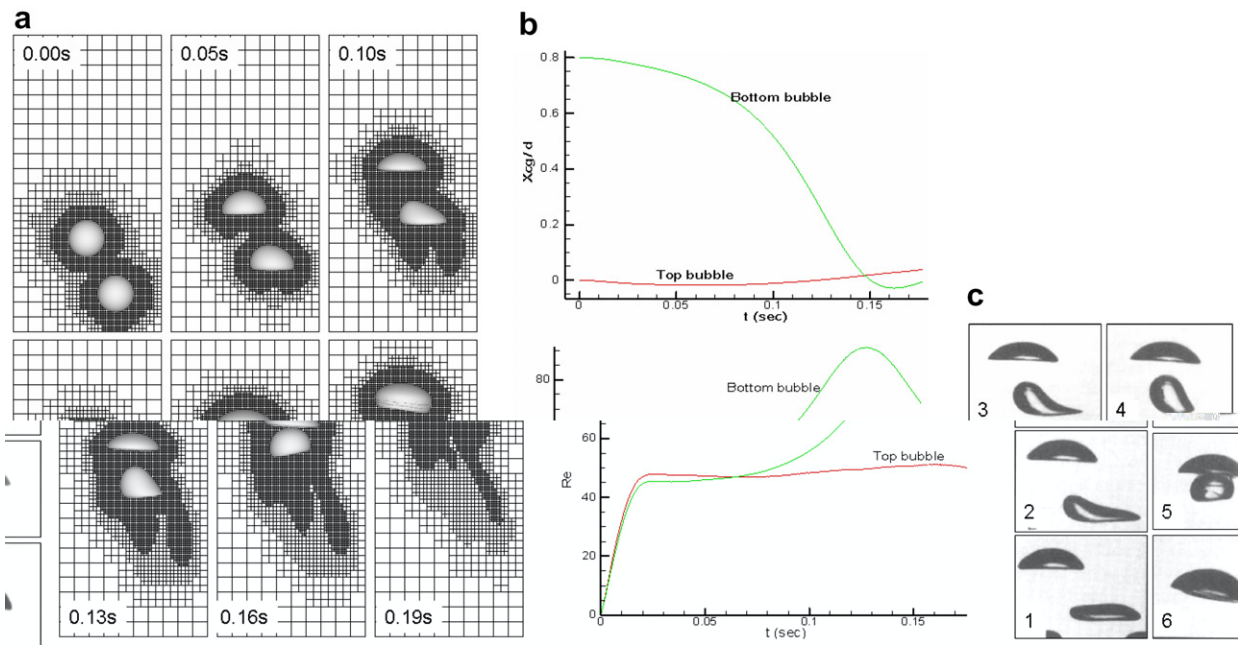

Fig. 15. Off-axis coalescence of rising bubbles: (a) computed shape history; (b) dimensional x-coordinates and rise-Reynolds number of the leading (top bubble) and trailing (bottom bubble) bubbles; (c) experimental photographs by Brereton and Korotney [29] at 0.03 s interval (taken from [28]).

5.4. Binary drop collision

A drop collision case with two equal size tetradecane ($C_{14}H_{30}$) drops in nitrogen medium is computed to further validate and highlight the capabilities of the developed algorithms and implementations. A drop collision case with Reynolds and Weber numbers of 228.0 and 37.2 is computed and compared with the experimental photographs by Qian and Law [30]. The impact parameter B defined in Fig. 16(a) is set to 0.01 producing a near head-on collision. The fluid properties shown in Table 5 produce density and viscosity ratios of 666.1 and 179.3, respectively. The Weber number 37.2 with a near head-on collision is beyond the estimated critical Weber number of 34 after which droplets separate following an initial coalescence (regime 4 in Fig. 16(b)) [30].

The computations are performed on a $7.5d \times 7.5d \times 7.5d$ domain with four levels of refinement over an $11 \times 11 \times 11$ base grid. The collisions take place in the x–y-plane and the initial horizontal separation between the drop-centers in the x–y-plane, approximated from the figures of Qian and Law [30], is set to 1.25 times the drop diameter. The initial velocity conditions are obtained by imposing the impact velocity field in all the cells that have indicator function less than 1.0. The imposed velocity field is projected in a divergence-free space and used as the initial velocity condition. All the computational boundaries are assigned outlet condition imposing zero normal velocity gradient and constant atmospheric pressure. Several snapshots of the computational domain (cross-section in the x–y-plane) are shown in Fig. 17. The time history of the computational data-size is shown in Fig. 17.

The computed drop-shapes and the velocity vectors, along with experimental photographs, are shown in Fig. 18. The velocity vectors further highlight the dynamics and mechanism of collision, where a dimpled disc is formed following the initial coalescence. The high curvature at the disc-rim creates a high-pressure zone as compared to the almost flat disc-center ($t = 0.33$ ms) causing contraction that eventually moves towards producing a cylindrical shape. Subsequently, the blobs at the cylinder-ends create a high-pressure zone that attempts to contract the cylindrical. However, the axial velocities (shown at $t = 1.26$ ms and 1.71 ms) at the cylinder-neck are strong enough to produce a momentary breakup that is quickly followed by a re-merger due to the overall center-ward motion of the droplets.

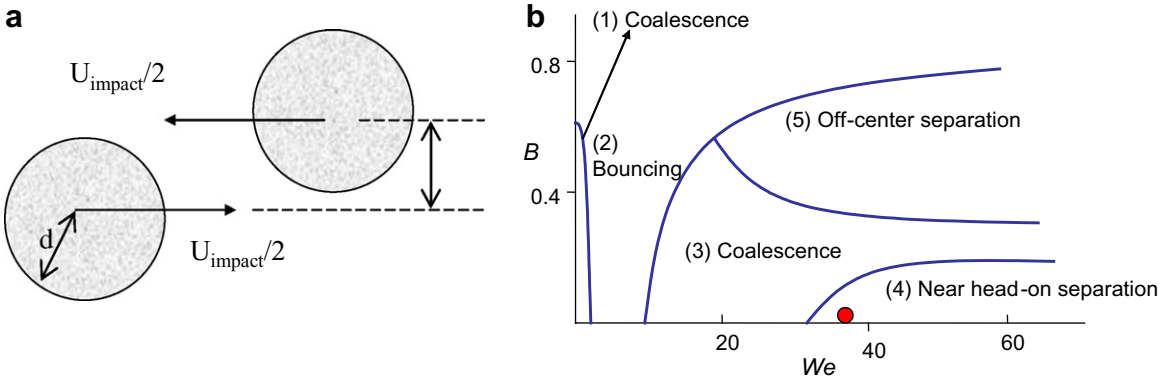

Fig. 16. Binary drop collision computation: (a) impact parameter B is defined as $B = h/d$; (b) collision-outcome diagram adapted from [30].

Table 5
Properties of tetradecane and nitrogen [31]

Medium	ρ (kg/m^3)	μ (kg/ms)	σ (N/m)
Tetradecane	758.0	2.128×10^{-3}	0.026
Nitrogen	1.138	1.187×10^{-5}	

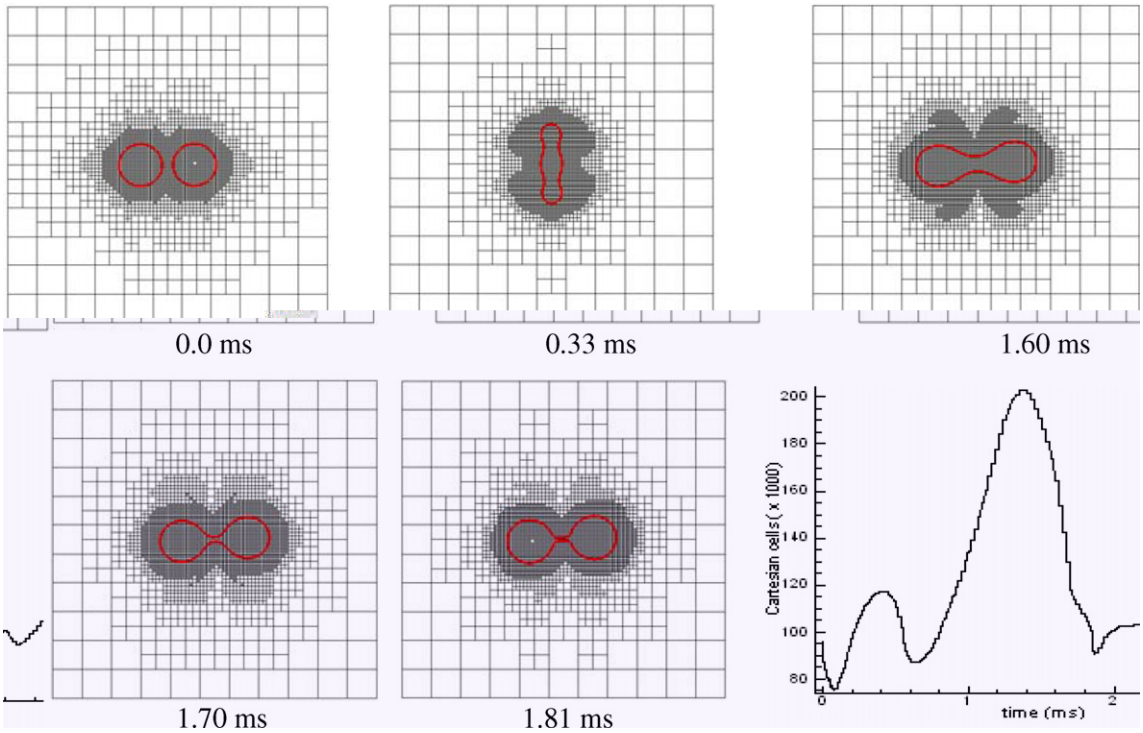

Fig. 17. Several cross-section views of the computational grid and the time history of Cartesian grid-size (number of 3D cells) for binary drop collision.

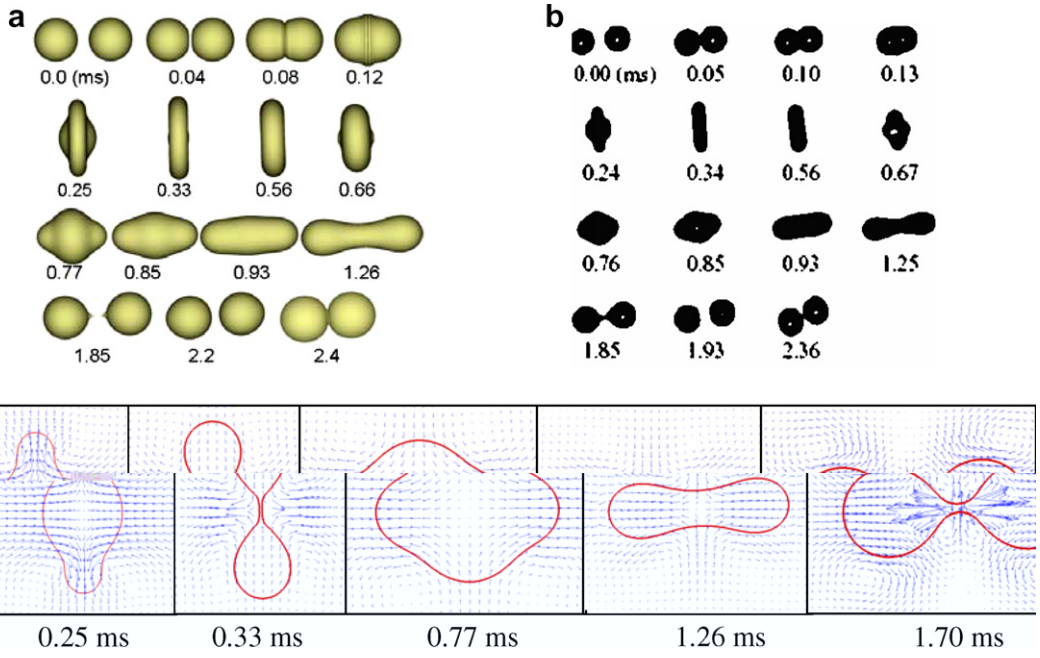

Fig. 18. Binary drop collision: (a) computed time history; (b) experimental photographs [30]; (c) interface and velocity vectors in a cross-section plane.

6. Conclusion

A three-dimensional local adaptive grid method along with the immersed boundary method was presented. A mass-conserving algorithm for marker addition and deletion was employed to control the interface resolution. The interface connectivity data offers an enhanced flexibility in dealing with multiple interfaces in close proximity, as in dense bubbly flows. The level-contour-based reconstruction technique provides a convenient approach in dealing with complex topology changes. The key steps of the reconstruction algorithm along with several issues related to maintaining valid connectivity information were presented. The call for interface reconstruction was automated by a simple probe-based criterion to check for potential topology changes. The accuracy of restructuring technique was characterized using an interface placed in a time reversed vortex field. The effect of reconstruction on mass-conservation and spurious velocities was highlighted. The background Cartesian grid was dynamically adapted based on the interface location and solution features, and a staggered grid discretization approach was used for the Navier–Stokes computation. The capabilities of the presently developed algorithms in studying interfacial-flows/bubble-dynamics were demonstrated via several rising bubble and drop collision/coalescence computations.

Acknowledgments

The present effort has been supported by the NASA Constellation University Institute Program (CUIP), Ms. Claudia Meyer program monitor. This work is dedicated to Pieter Wesseling to honor his substantial contributions to computational sciences.

References

[1] W. Shyy, Computational Modeling for Fluid Flow and Interfacial Transport, Elsevier, Amsterdam, The Netherlands, 1994 (revised printing 1997).
[2] W. Shyy, H.S. Udaykumar, M.M. Rao, R.W. Smith, Computational Fluid Dynamics with Moving Boundaries, Taylor & Francis, Washington, DC, 1996 (revised printing 1997, 1998 and 2001).
[3] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[4] D. Enright, R.P. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level-set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[5] 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) 455–584.
[6] 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.
[7] G. Tryggvason, B. Bunner, A. Esmaeeli, 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.
[8] R.K. Singh, J. Chao, M. Popescu, C.-F. Tai, R. Mei, W. Shyy, Multiphase/multi-domain computations using continuum and Lattice–Boltzmann methods, J. Aerospace Eng. 19 (2006) 288–295.
[9] R.K. Singh, Three-dimensional marker-based multiphase flow computation using adaptive Cartesian grid techniques, PhD thesis, University of Florida, 2006.
[10] M. Francois, W. Shyy, Micro-scale drop dynamics for heat transfer enhancement, Prog. Aerospace Sci. 38 (2002) 275–304.
[11] H.S. Udaykumar, H.-C. Kan, W. Shyy, R. Tran-Son-Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, J. Comput. Phys. 137 (1997) 366–405.
[12] D.J. Torres, J.U. Brackbill, The point-set method: front-tracking without connectivity, J. Comput. Phys. 165 (2000) 620–644.
[13] S. Shin, D. Juric, Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity, J. Comput. Phys. 180 (2002) 427–470.
[14] A. Esmaeeli, G. Tryggvason, Direct numerical simulations of bubbly flows, part II: moderate Reynolds number arrays, J. Fluid Mech. 385 (1999) 325–358.
[15] A. Esmaeeli, G. Tryggvason, A front tracking method for computations of boiling in complex geometries, Int. J. Multiphase Flow 30 (2004) 1037–1050.
[16] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[17] F.S. Sousa, N. Mangiavacchi, L.G. Nonato, A. Castelo, M.F. Tome, V.G. Ferreira, J.A. Cuminato, S. McKee, A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces, J. Comput. Phys. 198 (2004) 469–499.
[18] P. Wesseling, Principles of Computational Fluid Dynamics, Springer, New York, 2001.
[19] F. Losasso, F. Gibou, R.P. Fedkiw, Spatially adaptive techniques for level set methods and incompressible flow, Comput. Fluids 35 (2006) 995–1010.

[20] T. Ye, W. Shyy, C.-F. Tai, J.N. Chung, Assessment of sharp- and continuous-interface methods for drop in static equilibrium, Comput. Fluids 33 (2004) 917–926.
[21] T. Ye, W. Shyy, J.N. Chung, A fixed-grid, sharp-interface method for bubble dynamics and phase change, J. Comput. Phys. 174 (2001) 781–815.
[22] S. Marella, S. Krishnan, H. Liu, H.S. Udaykumar, Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations, J. Comput. Phys. 210 (2005) 1–31.
[23] A. Prosperetti, in: M. Rein (Ed.), Numerical Algorithms for Free-surface Flow Computations: An Overview, Drop–Surface Interactions, Springer, 2002.
[24] B.E. Griffith, C.S. Peskin, On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems, J. Comput. Phys. 208 (2005) 75–105.
[25] M.J. Aftosmis, Solution adaptive Cartesian grid method for aerodynamic flows with complex geometries, in: Computational Fluid Dynamics VKI Lectures Series, Belgium, 1997.
[26] Z.J. Wang, A quadtree-based adaptive Cartesian/quad grid flow solver for Navier–Stokes equations, Comput. Fluids 27 (1998) 529–549.
[27] R. Clift, J.R. Grace, M. Weber, Bubbles, Drops and Particles, Academic Press, New York, 1978.
[28] M.V.S. Annaland, N.G. Deen, J.A.M. Kuipers, Numerical simulation of gas bubbles behavior using a three-dimensional volume of fluid method, Chem. Eng. Sci. 60 (2005) 2999–3011.
[29] G. Brereton, D. Korotney, Coaxial and oblique coalescence of two rising bubbles, in: I. Sahin, G. Tryggvason (Eds.), Dynamics of Bubbles and Vortices near a Free Surface, ASME, New York, 1991.
[30] J. Qian, C.K. Law, Regimes of coalescence and separation in droplet collision, J. Fluid Mech. 331 (1997) 59–80.
[31] Y. Pan, K. Suga, Numerical simulation of binary liquid droplet collision, Phys. Fluids 17 (2005) 1–14.