

Three-Dimensional Adaptive Grid Computation with Conservative, Marker-Based Tracking for Interfacial Fluid Dynamics

Rajkeshar Singh^{(1)*} and Wei Shyy^{(2)†}

¹Department of Mechanical and Aerospace Engineering,
University of Florida, Gainesville, FL 32611, U.S.A.

²Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109, U.S.A.

For computational multiphase flow involving detailed interfacial dynamics, the outstanding issues include topological changes, geometric capturing, local resolution, conservative treatment, and spurious characteristics associated with property jumps across the interface. In this work, a marker-based, conservative immersed boundary with triangulated surface grid representation for 3D interfaces and local adaptive grid refinement for resolution enhancement is presented. Markers are dynamically added and deleted from the interface using a conservative restructuring to preserve the phase volume and maintain a satisfactory interface resolution. The topological changes are handled using the level-contour reconstruction technique while preserving the logical connectivity information of the interface, which provides enhanced flexibility as compared to the connectivity-free marker-based tracking methods with minimal added algorithmic-complexity. An adaptive Cartesian grid technique has been employed to refine the field computation. The overall accuracy of the algorithms has been established via studies of spurious velocity currents for static bubbles, rising bubble simulations with various terminal shapes, and head-on and off-axis coalescence of two bubbles.

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 treatment of discontinuities in the fluid and flow properties, accurate tracking and computation of geometric properties of the interface remains challenging. For reviews of the key computational issues, we refer to Shyy et al.^{1, 2}.

Numerous techniques for tracking the interface exist, each with its own strengths and weaknesses. Two most widely used classes of methods are Eulerian and mixed Eulerian-Lagrangian methods. The volume of fluid (VOF)^{3, 4} and level-set (LS)⁵ fall in the category of Eulerian methods which track the interface via a scalar function on an Eulerian (stationary) grid. The mixed Eulerian-Lagrangian method^{6, 7, 9, 10} tracks the interface using mass-less markers on an Eulerian grid. The details of some of the recent developments in Eulerian methods can be found in the works of Enright et al.¹¹ on particle level-set method; Aulisa et al.¹² on hybrid marker-VOF method; Sussman¹³ on coupled LS –VOF method. These developments have been aimed at ameliorating some of the difficulties with LS and VOF methods with regards to mass-conservation and accurate computation of geometric properties of the interface.

The most common approach for tracking three dimensional marker based interfaces is via triangulated surface grid representation^{6, 7, 17}. A marker-based tracking allows accurate representation of the interface even at sub-grid

* Graduate Student, Student Member AIAA.

† Clarence L. “Kelly” Johnson Collegiate Professor and Chair, Fellow AIAA.

level and does not suffer from numerical diffusion as the computation progresses in time. However, the most serious drawback of this approach has been attributed to the algorithmic complexity in maintaining the interface connectivity information in three-dimensional geometries, especially in the event of topological changes. Some of the recent developments have been geared towards eliminating the need for explicit connectivity information and thus simplifying the overall procedure. The connectivity free point-set method of Torres and Brackbill¹⁴ and connectivity free tracking of triangular elements by Shin and Juric¹⁵ represent some of the efforts in simplifying marker-based tracking.

The connectivity free tracking with level-contour based interface reconstruction technique of Shin and Juric¹⁵ presents a significant advance towards overall simplification of explicit marker based tracking. It uses a scalar function contour (similar to the zero-contour in the level-set method) to reconstruct the interface. The lack of explicit connectivity among interface triangles allows a cell-by-cell reconstruction approach where triangles are created in one cell at a time without having to worry about the elements created in neighboring cells. In addition to reconstruction for topological changes, markers need to be locally added and deleted from the interface to maintain appropriate resolution. In this regard, despite the simplicity offered by the method of Shin and Juric¹⁵, the lack of explicit connectivity information makes it difficult to locally add and delete markers without resorting to reconstruction of the entire interface even when no topological changes take place. For example, in bubbly flows such as those simulated by Esmaeeli and Tryggvason¹⁶ where $O(100)$ bubbles move in close proximity but are not to merge with each other, a level-contour based interface reconstruction may introduce undesirable mergers.

The presented work uses triangulated interfaces and maintains an explicit connectivity information at all times. Availability of connectivity information readily allows local marker addition and deletion. Since the basic idea of level-contour-based interface reconstruction¹⁵ is a generic approach, it can be used for reconstructing the interface while maintaining the connectivity information, which allows local marker addition and deletion wherever desired. Consequently, reconstruction of the interface is performed only when topological changes are expected and thus minimizing perturbations caused by reconstruction.

In the present approach, the interfacial dynamics is modeled using the immersed boundary method^{6, 18}. The capabilities of an adaptive Cartesian grid are employed to improve the desirable resolution without substantially increasing the overall computation cost. The numerical solutions of Navier-Stokes equations are obtained using the projection method for finite volume formulation on collocated grid^{24, 26, 27}. Further work on assessing the impact of extra dissipation due to momentum interpolation and issues with parallel computing highlighting the interplay of domain decomposition and computation cost for interfacial flows will be reported in future communications. The principal contributions of the work presented in this document are summarized below:

1. Conservative interface restructuring

The interface restructuring is performed at each time-step to locally add or delete markers by breaking long edges and deleting short ones. Sousa et al.¹⁷ developed a phase-volume preserving interface smoothing (marker relocation) technique but still observed phase-volume errors caused primarily due to the commonly used marker deletion procedure of removing edges by collapsing it to its midpoint. The current approach eliminates this error by extending the idea of phase-volume preserving interface smoothing to a conservative marker deletion algorithm.

2. Improved capability for handling topological change

The level-contour based reconstruction is used to achieve topological changes by performing cell-by-cell interface-reconstruction while maintaining the interface connectivity information. The connectivity free approach of Shin and Juric¹⁵ performs reconstruction at regular intervals to maintain marker spacing and capture topological changes. Exploiting the connectivity information, the current approach can decouple restructuring and reconstruction by performing reconstruction only for topological changes while using conservative restructuring to maintain interface resolution.

3. Adaptive grid computation

An adaptive grid based computation combined with a robust multidimensional marker based tracking is used for multidimensional immersed boundary computations. The location of the interface and the solution characteristics are used as indicators for refining or coarsening the grid. The Navier-Stokes flow equations are solved using projection method with finite volume formulation on collocated grids.

In addition to present the specific techniques for tracking interfacial dynamics, as mentioned above, the overall accuracy of the algorithms has been established via study of several test cases, including spurious velocity currents for static bubbles, rising bubble simulations with various terminal shapes, and head-on and off-axis coalescence of two bubbles.

2. Immersed Boundary Formulation

The immersed boundary method^{6, 18} uses a single fluid formulation for the entire domain by smearing the fluid properties across the interface (Fig. 1). The incompressible Navier-Stokes equations for mass and momentum conservation (equation (1)) are solved on the stationary background grid and markers are advected using the marker velocities interpolated from the background grid. The surface tension forces are computed on the interface and distributed to the background grid as a momentum source term F_s . As the interface evolves, markers are added or deleted to maintain interface resolution and complete interface reconstruction is performed when topological changes are expected based on a criteria presented later. More detailed information on the immersed boundary method could be found in the works of Tryggvason et al.⁶, Udaykumar et al.⁸, Francois and Shyy^{9, 10} and Shin and Juric¹⁵. A brief description of key components of the current 3D immersed boundary implementation is provided in following sections.

$$\nabla \cdot u = 0$$

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u u) = -\nabla p + \nabla \cdot (\nabla u + \nabla^T u) + F_s + \rho g \quad (1)$$

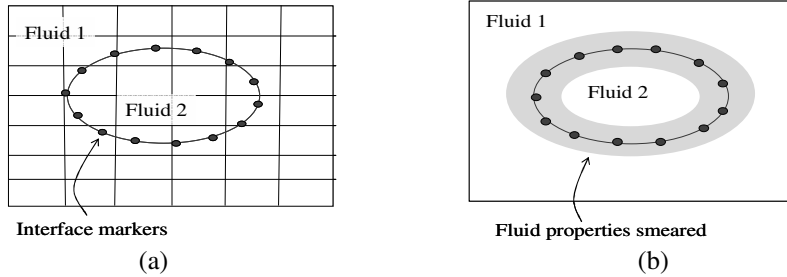


Figure 1: Illustration of immersed boundary method. (a) Interface markers are tracked on a stationary Cartesian grid; (b) the fluid property jumps are smoothed over a thin zone (4~5 grid cells) across the interface.

A. Interface Data-structure

A finite-element type data-structure is used to store the basic grid information. The coordinates of interface nodes (markers) and the three nodes forming a triangle are stored. As shown in Fig. 2, the nodes of triangles are stored in a consistent cyclic order such that the right hand thumb rule gives the normal vector pointing outside the interface. Identity of the three triangle-neighbors of each triangle and a list of triangles sharing a node is also stored.

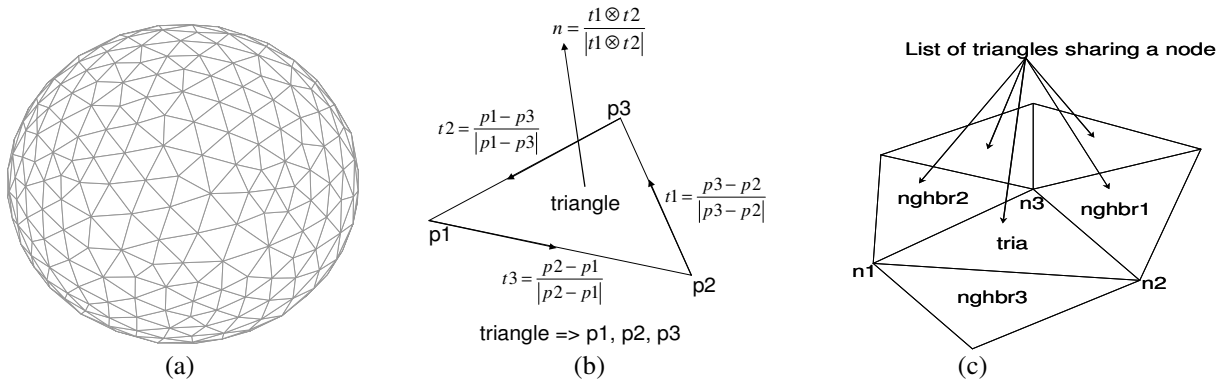


Figure 2: Main concepts employed in interface representation and data-structure. (a) A triangulated interface; (b) triangle nodes are stored in a cyclic order (e.g. (p1, p2, p3) or (p2, p3, p1)) to produce the normal vector n (using right hand thumb rule) pointing outside the interface; (c) neighbors (nghbr1, nghbr2, nghbr3) of a triangle and list of triangles connected to each node are also stored.

B. Material Property Smoothing

In the present approach, the fluid properties are smoothed using an indicator function method similar to that by Tryggvason et al.⁶. The indicator function is computed on the background grid by solving the Poisson equation (2) where the integration on the right side is performed over the interface with n and dA representing the local unit-normal vector and elemental area. The term $\delta_{(x-x_{\text{interface}})}$ is a Dirac-delta function non-zero only at $x = x_{\text{interface}}$. A three-dimensional discretized delta-function by Peskin and McQueen²⁰ (equation (3)) is used in the current work. The terms $(\Delta x, \Delta y, \Delta z)$ and (r_x, r_y, r_z) in equation (3) are the background cell dimensions and the vector from interface to the Cartesian cell-center, respectively. Since the fluid properties are constant in most part of the domain, the indicator Poisson equation (2) is solved only for cells falling in the delta-support region (non-zero source term for indicator Poisson equation (2)) and one extra layer of cells. It amounts to solving for the indicator function only within 5~6 cells across the interface. For rest of the cells, the indicator is manually set to unity outside and zero inside the interface.

$$\nabla^2 I(x) = \nabla \cdot \left(\int_{\text{interface}} n \delta_{(x-x_{\text{interface}})} dA \right) \quad (2)$$

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

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

The smoothed fluid density is evaluated using equation (4) where subscripts 1 and 2 denote the properties of fluids outside and inside the interface, respectively. The smoothed fluid densities are used to compute the smeared viscosity using equation (5) suggested by Prosperetti²¹.

$$\rho(x) = \rho_2 + (\rho_2 - \rho_1) I(x) \quad (4)$$

$$\frac{\rho(x)}{\mu(x)} = \frac{\rho_2}{\mu_2} + \left(\frac{\rho_2}{\mu_2} - \frac{\rho_1}{\mu_1} \right) I(x) \quad (5)$$

It is noted that there is a compatibility issue. Specifically, continuity of the viscous stresses normal to the interface can not be satisfied with a continuous interface method²¹. The above formulation for smeared viscosity is an attempt to satisfy the continuity of tangential shear stresses across the interface.

C. Momentum Source Computation

The currently used collocated grid arrangement requires the momentum source term (F_s in equation (1)) at Cartesian cell-centers. The surface tension forces are computed on the edges (f_{edge}) of the interface and distributed to the Cartesian cell faces using equation (6). The cell-face sources are subsequently averaged using density weighting to obtain the cell-center momentum source terms with the help of Fig. 3 and equation (7). The subscripts xe and xw denote x-component of source term distributed to the east and west faces. Similarly, subscripts yn and ys denote the y-component of source term on north and south faces. The subscripts for density terms denote the fluid density computed at respective cell-faces by linearly interpolating cell-center values. Such an approach, as compared to distributing the source terms directly to cell-centers, is found to reduce the spurious velocity currents seen in static bubble simulations by an order of magnitude²².

$$F_s = \sum_{\text{interface_edge}} f_{\text{edge}} D(r) \quad (6)$$

$$\frac{F_s}{\rho} = \frac{1}{2} \begin{pmatrix} \frac{f_{xe}}{\rho_e} + \frac{f_{xw}}{\rho_w} \\ \frac{f_{yn}}{\rho_n} + \frac{f_{ys}}{\rho_s} \end{pmatrix} \quad (7)$$

Since explicit computation of curvature for surface tension force (δf) on an elemental interface-area is sensitive to numerical errors, a line integral form as shown in equation (8)^{6, 19} is used. Here σ is surface tension and κ is local interface curvature. The line integral form obviates the need for explicit computation of curvature and provides a means to compute surface tension forces in a conservative fashion i.e. the net surface tension force on a closed interface is zero.

$$\delta f = \int_{\delta A} \sigma \kappa n dA = \int_{\delta A} \sigma (n \times \nabla) \times n dA = \int_s \sigma t \times n ds \quad (8)$$

Following al-Rawahi and Tryggvason¹⁹, the surface tension force acting at the center of a triangular element is approximated as a summation over the edges (equation (9)) where the normal and tangent vector on the triangle elements are computed as shown in Fig. 2(b). To ensure that the net force on a closed interface is zero (conservation), the cross-product of normal and tangent at the edge is obtained by averaging its value between the triangles sharing the edge¹⁹. These edge-forces are used to compute the cell-center momentum source terms as described earlier.

$$\delta f = \sum_{\text{edge}=1,2,3} f_{\text{edge}} = \sum_{\text{edge}=1,2,3} \sigma (t \otimes n)_{\text{edge}} ds \quad (9)$$

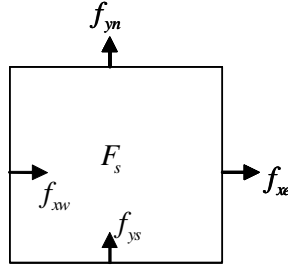


Figure 3: Cell center momentum source computation from face values for collocated grid implementation.

D. Interface Advection

Markers are advanced in time using the velocities interpolated from the background grid as shown in equation (10) where (dx, dy, dz) is Cartesian cell dimension and $D(r)$ is discrete Dirac delta function (equation (3)). The marker advection integration is carried out using second order Adam-Bashforth integration (equation (11)). The numerical errors associated with marker advection method produce some errors in the phase-volume occupied by the interface. Usually these errors are small but tend to accumulate for long duration computations. Such volume defects are explicitly corrected at each time step by perturbing the interface in local-normal direction to preserve the interface volume before and after marker advection.

$$u_{\text{marker}} = \sum_{\text{cell}} u D(r) dx dy dz \quad (10)$$

$$x^{n+1} = x^n + \Delta t \frac{3u^n - u^{n-1}}{2} \quad (11)$$

E. Marker Addition and Deletion: A Conservative Restructuring

Markers are added by introducing a new marker at the mid-point of the edges longer than a background cell size. Similarly the removal procedure collapses/deletes edges that are shorter than a third of the cell-size⁶. A typical edge deletion procedure collapses the edges at the midpoint resulting in a local phase-volume (interface volume) error^{6,17}. Usually these errors are small but they can accumulate for long duration computations and eventually become more substantial, as shown by Sousa et al.¹⁷. The marker addition does not cause any defect in the phase volumes hence the presently employed restructuring approach is the same as in Fig. 4 but with an added correction step to the edge deletion procedure to locally preserve the phase-volumes. To explain the approach in 2D, Fig. 5(a) shows a marker removal by collapsing the edge p2p4 to the mid-point p3 resulting in a net phase-area loss. To correct this error, first a reference point Pref is defined at p3-n (Fig. 5(b)) where n denotes the unit normal vector. A reference phase area p1p2p4p5Pref is computed and point p3 is relocated (Fig. 5(c)) to set the new area (p1p3p5Pref) same as the reference area.

The conservative marker deletion on 3D surfaces is similar to the 2D approach. Key steps of the algorithm for 3D interfaces are presented below with the help of Fig. 6.

ALGORITHM: Conservative interface restructuring

- Select a point P at the midpoint of the edge to be deleted. A local normal vector n is defined using the triangles connected the edge.
 - A reference point Pref is defined at P-n and reference phase-volume Φ_{ref} is computed by adding the volumes of tetrahedrons made by point Pref and all the triangles in Fig. 6(a).
 - Collapse the edge to its midpoint (Fig. 6(b)) and compute the new volume Φ made by point Pref and the triangles of Fig. 6(b).
 - Relocate the point P to Pref+(Φ_{ref}/Φ) n to preserve the local phase volume.
-

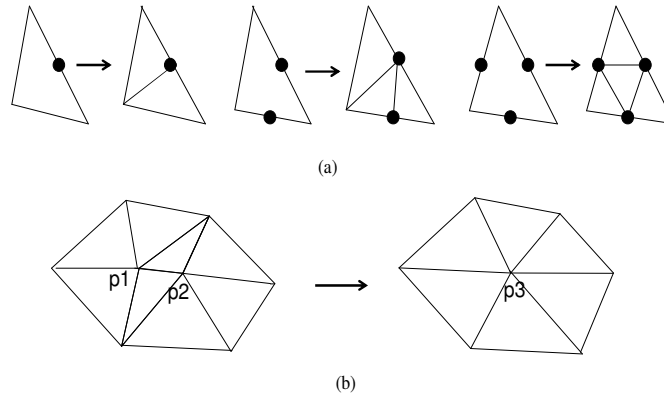


Figure 4: Schematic of grid addition and deletion to maintain consistent interfacial resolution. (a) Various numbers of long edges flagged for splitting at the midpoint by creating new markers and reconnected to create new triangles and (b) a short edge (edge p1p2) is deleted by collapsing to p3 at the midpoint.

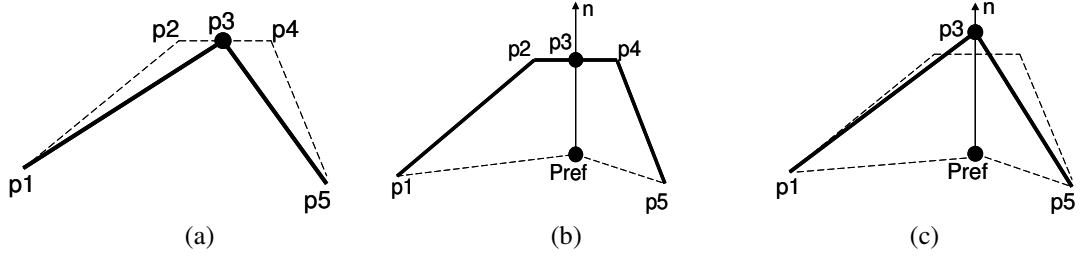


Figure 5: Illustration of the conservative interface restructuring procedures in 2D. (a) Old interface $p_1p_2p_4p_5$ and new interface $p_1p_3p_5$ with non-conservative edge deletion; (b) conservation correction computes a reference area ($p_1p_2p_4p_5Pref$) before edge deletion and (c) preserves this reference area ($p_1p_3p_5Pref$) after deletion.

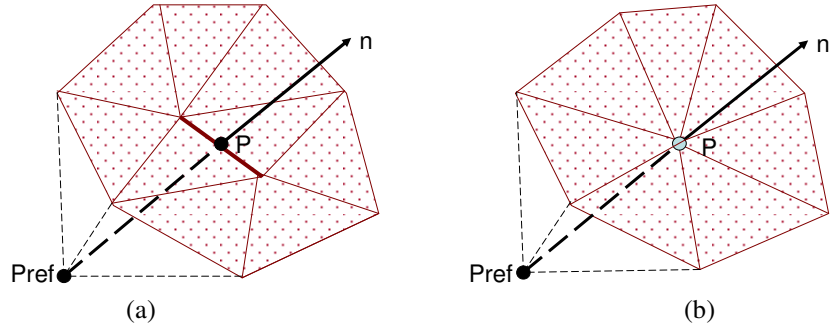


Figure 6: Illustration of conservative restructuring procedures in 3D. (a) Reference point $Pref$ and local normal vector n at the edge-midpoint P ; (b) locally re-triangulated surface after edge deletion and correction to location of P for phase-volume conservation.

1. Restructuring Test: Interface in a time reversed vortex field

A spherical interface placed in a time-reversed vortex field deforms and should, theoretically, return back to the original spherical shape. However, the errors associated with marker advection and interface restructuring (if non-conservative method is used) introduce some errors in the shape and volume of the final interface. Due to the severe deformations withstood by the interface, this particular test presents a good choice to assess the impact of the conservative restructuring algorithm. Tests were performed by placing a spherical interface of diameter 0.15 at (0.5, 0.75, 0.5) in a unit cube with its center at (0.5, 0.5, 0.5) (Aulisa et al.¹²) (Fig. 7(a)). The time reversed vortex field with period $T = 4$ in equation (12) is located at the center of the box with maximum velocity halfway from the center to the side of the box. The shape history computed for a $60 \times 60 \times 60$ grid is shown in Fig. 7(b).

The error in the final interface (at $t = T$) radius, area and volume are presented in Table 1. Figure 8 shows the time history of phase volume errors and grid convergence for conservative and non-conservative restructuring. It should be noted that no explicit phase-volume correction after marker advection is applied for this test and the phase volume error with conservative restructuring is caused solely by the errors associated with marker advection using the interpolated velocity field. As seen in Figure 8, the conservative method performs considerably well showing an order of magnitude smaller volume error and exhibits a better than quadratic convergence with grid refinement for the current problem.

$$\begin{aligned}
 u(x, y, z) &= \cos(\pi t/T) \sin^2(\pi x) (\sin(2\pi z) - \sin(2\pi y)) \\
 v(x, y, z) &= \cos(\pi t/T) \sin^2(\pi y) (\sin(2\pi x) - \sin(2\pi z)) \\
 w(x, y, z) &= \cos(\pi t/T) \sin^2(\pi z) (\sin(2\pi y) - \sin(2\pi x))
 \end{aligned} \tag{12}$$

Table 1: Error estimates comparing the performance of current conservative restructuring method with non-conservative method for an interface placed in a time reversed vortex field given by equation (12)

Grid	RMS radius error		% surface area error		% volume error	
	Conservative	Non-conservative	Conservative	Non-conservative	Conservative	Non-conservative
303	2.44×10^{-3}	1.03×10^{-2}	4.29	-7.94	-0.56	-14.10
603	9.10×10^{-5}	4.21×10^{-3}	1.53	-2.41	-0.12	-4.99
1203	7.90×10^{-5}	1.16×10^{-3}	0.42	-0.60	-0.02	-1.05

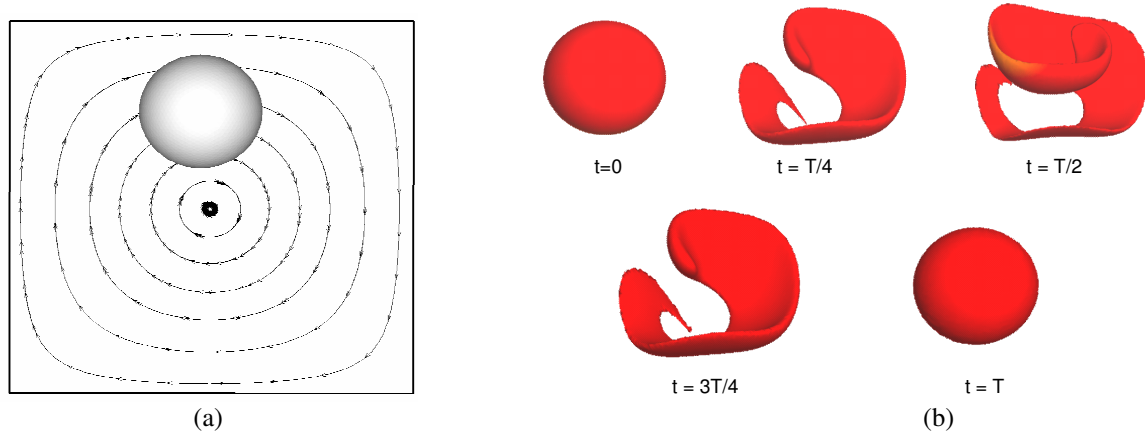


Figure 7: Interface in a time reversed vortex field test. (a) A spherical interface placed in a time reversed vortex field; (b) computed shape history for period $T = 4$.

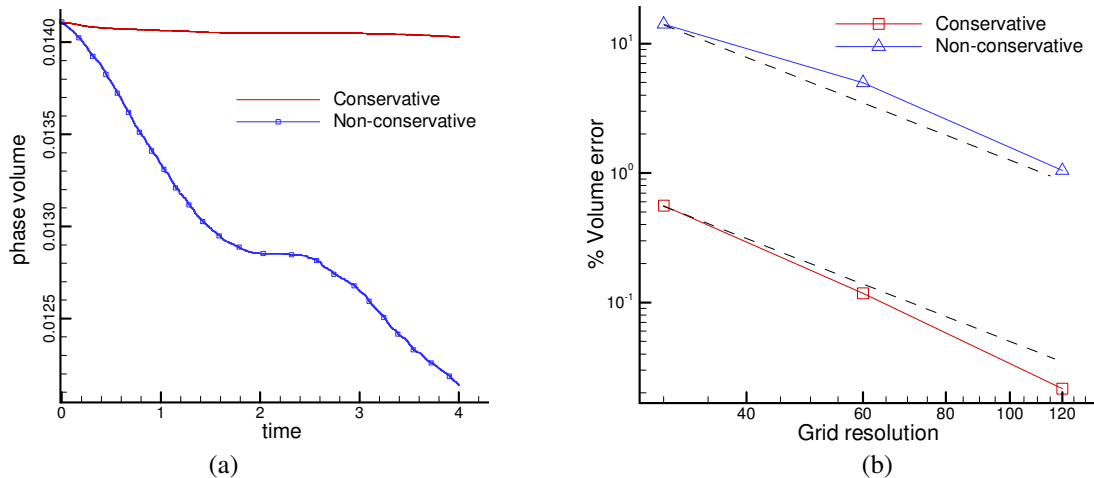


Figure 8: Comparison of phase volume error with conservative and non-conservative interface restructuring. (a) Time history of phase volume for grid $30 \times 30 \times 30$; (b) final ($t = T$) phase volume error for different grid resolutions ($30 \times 30 \times 30$, $60 \times 60 \times 60$ and $120 \times 120 \times 120$) where the dotted line is reference second order convergence.

3. Interface Reconstruction for Topology Change

The interface reconstruction algorithm is based on the indicator contour based reconstruction method of Shin and Juric¹⁵ originally developed for their connectivity free tracking. As shown in Fig. 9, the 0.5 level contour of the indicator function computed in equation (2) provides a good representation of the interface and allows topological

changes in a manner similar to the Eulerian methods. The approach for reconstruction is to first create a set of globally numbered markers on all the Cartesian cell edges at indicator = 0.5 and establish the connectivity information cell-by-cell as represented by Fig.10 for a 2D interface.

Reconstruction is performed when topological changes are expected based on certain criteria. At present, a simple criteria presented below is used to detect the need for reconstruction.

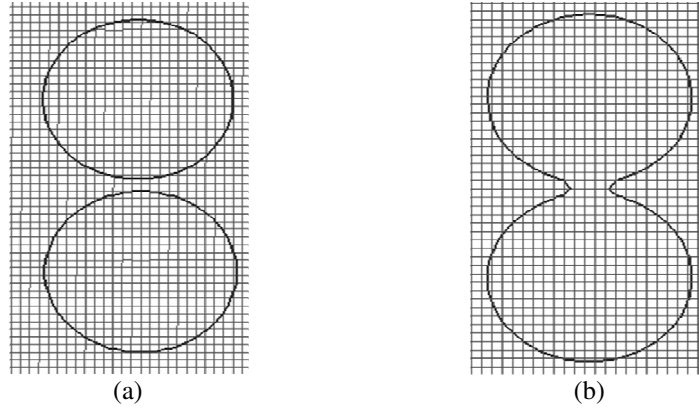


Figure 9: Two interfaces in close proximity and their representation by the indicator function. (a) Two interfaces within two cell-distance from each other; (b) indicator = 0.5 contour representing a merged topology.

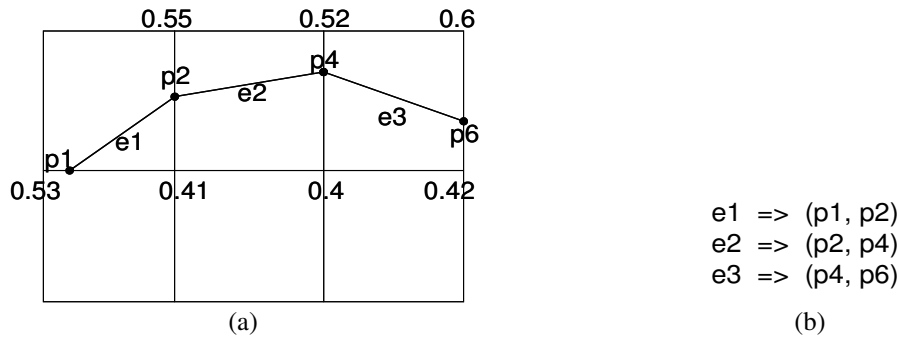


Figure 10: Two dimensional interface reconstruction by creating (a) globally numbered markers (p1, p2, p4, p6) at edges where indicator = 0.5 and (b) the edge-connectivity information. The values at vertices denote the indicator function values at corresponding locations.

A. Reconstruction Criteria

The current implementation sends two probes along the local normal vector from the marker points and interpolates the indicator value at these probes (Fig. 11). The length of each probe is set to one and half times the background cell-size. The reconstruction is initiated if both the probes have indicator function value greater than half or less than half. If both the probes have indicator values smaller than half, it indicates that these probes possibly lie inside two different interfaces involving a merger situation. Similarly, both probe-indicator values greater than half represent a case where a part of the same interface is thinner than two cells and reconstruction will, in effect, either delete a sharp tail or break the interface into multiple droplets.

In order to minimize the effect of some local sub-grid undulations undesirably calling for reconstruction, the local normal vector used at the interface is obtained from the indicator function gradient in the background grid-cell. Furthermore, a call for reconstruction from a marker point is entertained only if all other markers connected to it also produce the same request.

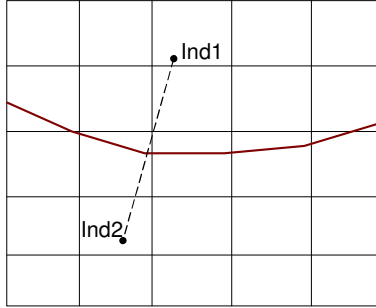


Figure 11: Reconstruction is performed if the indicator (Ind1 and Ind2) values at two probes in opposite directions along local interface-normal are both either smaller or greater than 0.5.

B. Reconstruction Procedure

Cell-by-cell interface reconstruction warrants some clarifications to deal with special cases such as interface intersecting a Cartesian cell at a cell-vertex or running almost parallel to an edge or face of the cell. A simple approach by assigning binary flags to Cartesian cell-vertices handles some of these difficulties with ease. Each vertex of the cell is flagged either zero or one using the indicator value computed in equation (2) and a marker is created on an edge only if the two vertices of the edge have different flag values. It should be noted that the markers are created on one edge at a time and no consideration is given to a possibility that two or more markers (created on different edges) may have same physical location. One such situation arises when the indicator = 0.5 contour lies on a cell-vertex as shown in Fig. 12(a). Figure 12(a) represents two overlapping markers created on edge E1 and E2 because of different vertex-flags.

During cell-by-cell reconstruction, no special attention is given to possibility of a reconstructed triangle (edge in 2D) having zero area. For example, the reconstructed edges e1 and e2 in Fig. 12(b) may be of zero length if the reconstructed markers p1, p2 and p3 all lie at the cell vertex (same physical location). Irrespective of the edge length or the physical location of markers, the edges e1 and e2 are treated as two ordinary edges in the data structure. Such edges are later on deleted by the conservative restructuring algorithm anyways. Figure 13 shows some common interface polygons reconstructed inside a cell with up to six markers. The cases of six markers creating two triangles inside a cell were observed only occasionally.

$$\text{vertex Flag} = \begin{cases} 1 & : \text{if indicator} \geq 0.5 \\ 0 & : \text{otherwise} \end{cases} \quad (13)$$

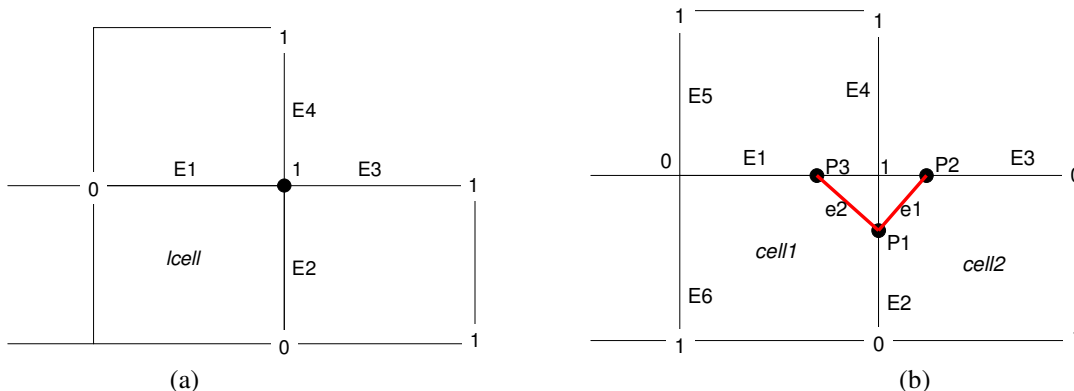


Figure 12: Illustration of 2D cell-by-cell reconstruction handling degenerate cases of duplicate markers and zero length edges. (a) Two markers created on Edges E1 and E2 have same physical locations but are counted as two different markers. (b) Newly created markers P1, P2 and P3 are connected in cells cell1 and cell2 to produce edges e2 and e1, respectively. Reconstruction procedure does not give any attention to the length of these two edges that may be zero if markers P1, P2 and P3 have same physical location.

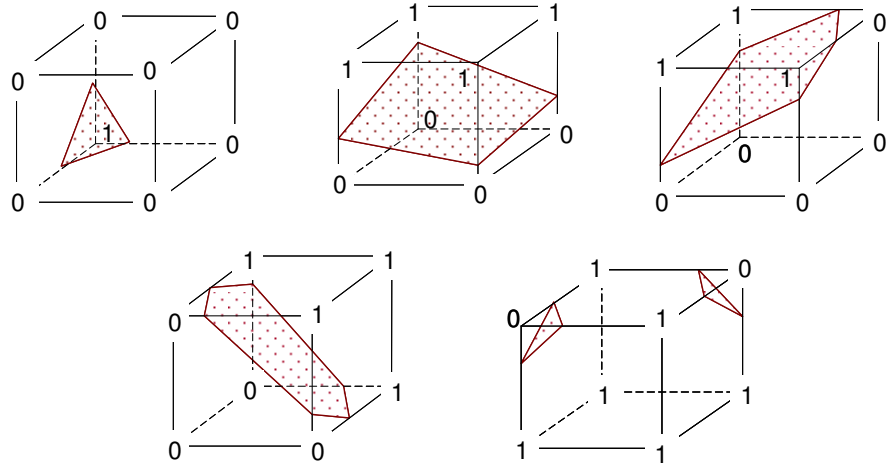


Figure 13: Some common polygons and integer vertex-flags with the first four being the most common occurrences. The last two cases contain six markers each but one has single six-sided polygon while the other has two triangles. The cases with six markers are treated separately to distinguish these two possibilities.

As shown in Fig. 14, if more than six markers are seen on the edges of a Cartesian cell, assembling these markers into a polygon becomes difficult due to the limited knowledge available within a cell making it non-trivial to define a unique surface segment with the cell. Fortunately, due to the smooth nature of the indicator function such cases were not observed in any of the tests so far. Using the vertex flags, Fig. 14(c) shows that ambiguities created by more than six markers in a cell are avoided with tetrahedral cells. Tetrahedral cells can contain at the most three markers and hence a unique surface patch is always definable. Such an observation can be used, if required, by temporarily breaking the Cartesian cells into tetrahedrons.

The newly created triangles are oriented using the gradient of the indicator function (Fig. 15(a)) to point the normal vector in the direction of increasing indicator value (pointing outside the interface). Excessively small triangles/edges are deleted by conservative restructuring method following the reconstruction. Any phase-volume defect caused by reconstruction is explicitly corrected by perturbing the markers in local normal direction and using bisection method as described in the earlier section on marker advection procedure.

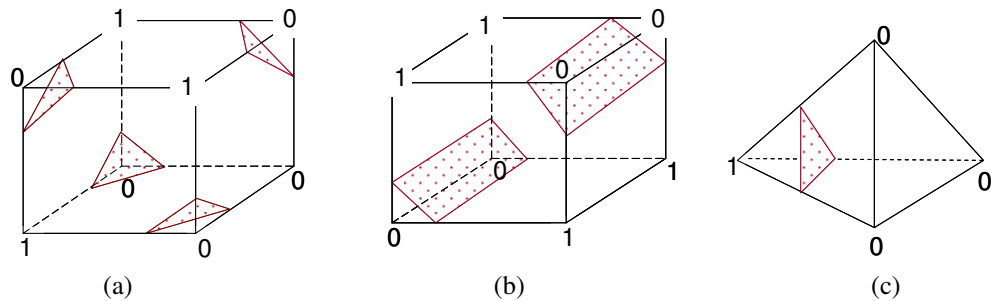


Figure 14: Difficulty in creating unique interface within a cell if more than six markers present. (a) One of several ways twelve markers may be connected to generate part of the interface inside a cell; (b) a cell containing eight markers and one of multiple ways to define the reconstructed interface; (c) a tetrahedron can have only three markers so it may be used by breaking Cartesian cells in to tetrahedrons to avoid degenerate cases shown in part (a) and (b).

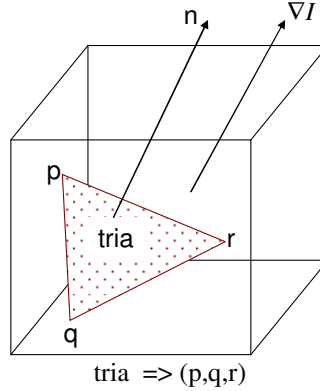


Figure 15: Orientation of triangles (tria) is set in the direction of increasing indicator function value and vertices of triangle are stored as (p, q, r).

C. Effect of Reconstruction on Mass Conservation

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. Although such volume errors are explicitly corrected by perturbing the interface in local normal direction, the magnitude of such losses and subsequent corrections are characterized below for reconstruction of a simple spherical interface.

A sphere of radius 0.2 at the center of a domain with dimensions (0.4x0.4x0.4) was reconstructed with different background grid resolutions. The volume losses with varying number of grid-cells per interface-diameter are presented in Table 2. The volume error convergence plot in Fig. 16 shows a quadratic convergence as observed by Shin and Juric¹⁵. The volume errors produced by reconstruction are corrected explicitly by perturbing the markers in local normal direction. Denoting the surface area and volume of the interface as A_0 and V_0 , an approximate radial perturbation caused by reconstruction may be estimated using equation (14) where ΔV is the volume error due to reconstruction. Since the volume defect has second order convergence, it shows a second order convergence behavior for the radial perturbation as well. The magnitude of this perturbation can also be quantified in terms of the background cell size Δ representing the grid resolution. If the diameter (d) of the interface is resolved with N grid cells, the radial perturbation can be rewritten in terms of grid cell size (Δ) as shown in equation (14). The last column of Table 2 shows the behavior of this quantity and reveals that this perturbation is approximately one tenth of the grid cell even for the coarsest grid used. Considering the fact that fluid properties are smeared over nearly four to five cells across interface making the immersed boundary method first order accurate, the amount of radial disturbance caused by even moderately resolved grids ($N = 20$) is reasonably small.

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

Table 2: Effect of reconstruction showing the volume defect in second column and approximate measure of the radial perturbation required to correct this defect explicitly in third column.

Grid cell per diameter $N = d/\Delta$	% Volume defect without explicit correction $100 \Delta V/V_0 $	Approximate radial perturbation $ \Delta r \approx (\Delta V 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	$1.64 \times 10^{-2} \Delta$

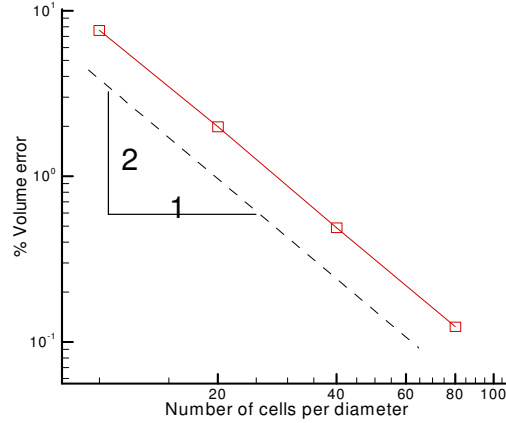


Figure 16: Volume error due to reconstruction shows quadratic convergence (dashed line as a reference) with grid resolution.

4. Adaptive Cartesian Grid-Based Computation

An isotropically adaptive Cartesian grid was used for solution of the flow governing equation. The grid is stored using an unstructured data format described in the work Aftosmis²³ and Ham and Lien²⁴. Brief information on the grid generation and Navier-Stokes solution procedure is provided in the following sections.

A. Adaptive Grid Generation

Figure 17 shows the face-based data-structure used for storage. Four integers (*level*, *i*, *j*, *k*) are stored for every cell along with a list of faces forming a cell. For every face, the face-orientation and the two cells sharing the face are stored. The variable named *level* stores the information of the number of times a cell has been split. The variable *orientation* for a face gives the orientation: faces normal to *x*, *y* and *z* axes have *orientation* = 1, 2 and 3, respectively. The variable *sideCell1* is identity of the cell sharing this face in the direction of the face orientation and *sideCell2* is cell on the other side of this face.

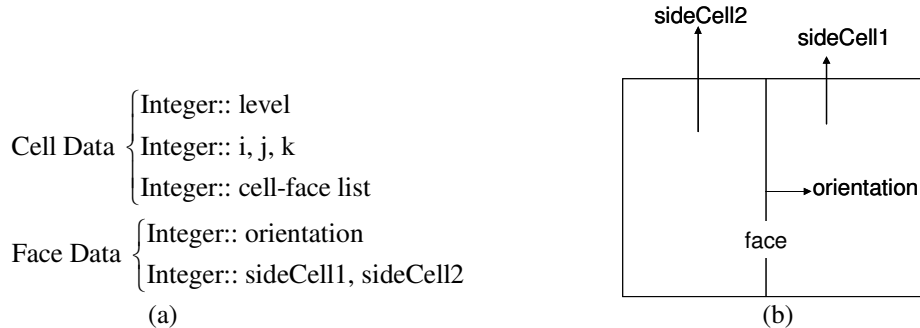


Figure 17: (a) Cartesian cell and face data-structure, (b) the convention for two side cells for a face with orientation along *x*, *y* or *z* co-ordinate direction.

The grid generation process starts by uniformly dividing the entire domain into prescribed number of cells in each coordinate direction producing the base grid over which a cell-by-cell adaptation is conducted. These base cells are the largest cells possible in the computational domain and have their *level* coordinates set to zero. The dimension and cell-center (*xc*, *yc*, *zc*) of a 3D cell with coordinates (*level*, *i*, *j*, *k*) are computed using equation (15) where (*L_x*, *L_y*, *L_z*) is the dimension of computational domain and *N_x*, *N_y*, *N_z* are the number of base grid cells in the three coordinate directions. The interface location and solution are used as two adaptation criteria. After creating a base grid, few layers of cells (8 to 10) across the interface are recursively adapted to produce desired interface resolution as represented by Fig. 18.

$$(dx, dy, dz) = \frac{1}{2^{level}} \left(\frac{L_x}{N_x}, \frac{L_y}{N_y}, \frac{L_z}{N_z} \right) \quad (15)$$

$$(xc, yc, zc) = ((i-0.5)dx, (j-0.5)dy, (k-0.5)dz)$$

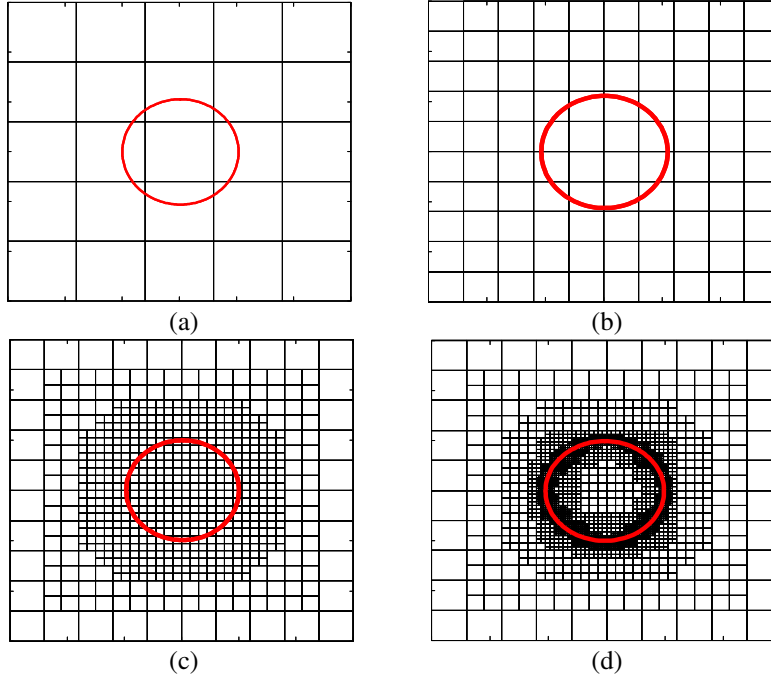


Figure 18: Steps of interface location based adaptation with 4~5 layers of cells across the interface refined uniformly, (a) Initial 5x5 base grid; (b) grid after one level of adaptation; (c) grid after three levels of adaptation; (d) grid after six levels of adaptation.

The computation is started on the grid obtained after interface based refinement. To avoid non-uniform cells around the interface and maintain grid-quality, 8 to 10 layers of cells across the interface are always kept at the same resolution. To enforce a smooth variation in grid cell-size, cells sharing a face are not allowed to differ by more than one level of refinement. A solution based grid adaptation method is employed for cells away from the interface. A velocity-curl based criterion used in the work of Wang²⁵ is employed. It computes a parameter ζ for each cell and standard deviation S_d as shown in equation (16) where l represents a length scale and N_{cell} is the total number of Cartesian cells. The length scale was computed as the cube root of cell-volume. The decision to refine or coarsen a cell is made by comparing ζ_{cell} with S_d as shown in equation (17) with the parameters α and β set to 1.0 and 0.1, respectively.

$$\zeta_{cell} = |\nabla \otimes u| l^{3/2}$$

$$S_d = \sqrt{\frac{1}{N_{cell}} \sum_{i=1, N_{cell}} \zeta_i^2} \quad (16)$$

$$\begin{aligned} \zeta_{cell} > \alpha S_d &\Rightarrow \text{Refine cell} \\ \zeta_{cell} < \beta S_d &\Rightarrow \text{Coarsen cell} \end{aligned} \quad (17)$$

B. Navier-Stokes Flow Solution Procedure

The overall immersed boundary algorithm is outlined below. The flow governing equations in step 5 of the algorithm shown below are solved using a collocated grid arrangement. This approach stores the flow variables such as velocity vector (u), pressure (p) and all the fluid properties at the cell-center. Additionally, it also stores the face normal velocities (U) on cell-faces to enforce mass-conservation (Fig. 19). A projection method^{26, 27} is used to solve the governing equations in finite volume formulation. Second order Adam-Bashforth and Crank-Nicholson schemes are used for temporal discretization of the convection and viscous terms respectively.

ALGORITHM: Immersed boundary computation

Step 1: Interpolate marker velocities and move markers to new location. Explicitly conserve any phase volume error by perturbing markers in local normal direction.

Step 2: Restructure interface by adding/deleting markers to maintain the desired marker spacing.

Step 3: Compute indicator function and fluid properties in grid cells.

Step 4: Compute surface tension forces and distribute as momentum source term.

Step 5: Solve Navier-Stokes equation to advect the velocity field to next time step.

Step 6: Check for geometry and solution based grid adaptation at prescribed intervals.

Step 7: Check for interface reconstruction at prescribed intervals:

- Perform reconstruction if required.
 - Correct the global phase volume error caused by reconstruction. The magnitude of volume correction applied to individual interface (objects) is inversely proportional to their volumes.
-

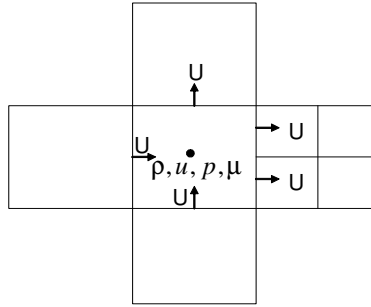


Figure 19: Collocated grid arrangement with two sets of velocity fields.

The cell-to-face interpolation of the velocities and velocity gradients at faces are computed using the auxiliary node method of Ham and Lien²⁴. The gradients at a cell center are computed using weighted least square fitting with cells sharing a common face (Fig. 20). The linear reconstruction of a scalar variable and the corresponding weighted errors at the participating neighbor cells (Fig. 20) are shown in equation (18). The weights are defined as $w_i = 1/l_i$ where l_i is the distance between x_0 and x_i . The corresponding square error (equation (19)) is minimized to obtain a linear system in represented by equation (20) where the matrix A depends only on the geometry of the grid cells and hence can be inverted once and saved.

$$f = f_0 + a(x - x_0) + b(y - y_0) + c(z - z_0) \quad (18)$$

$$\epsilon_i = w_i (a(x_i - x_0) + b(y_i - y_0) + c(z_i - z_0) + f_0 - f_i)$$

$$\epsilon^2 = \sum_{neighbor} \epsilon_i^2 \quad (19)$$

$$\frac{\partial \epsilon^2}{\partial a} = 0; \quad \frac{\partial \epsilon^2}{\partial b} = 0; \quad \frac{\partial \epsilon^2}{\partial c} = 0 \Rightarrow [A] \begin{pmatrix} a \\ b \\ c \end{pmatrix} = S \Rightarrow \begin{pmatrix} a \\ b \\ c \end{pmatrix} = [A]^{-1} S \quad (20)$$

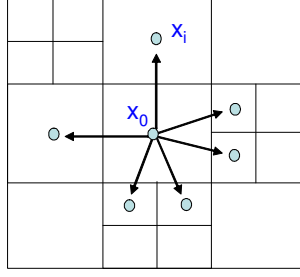


Figure 20: Participating neighbor cells for least square reconstruction of a scalar field in a cell with center at x_0 .

The discretized governing equations can be cast in the form of equation (21) and solved for an intermediate velocity field u^* . Term S contains all the know values such as surface tension source, gravitation, convection and old time-step viscous term. Intermediate face normal velocities U^* are computed by linearly interpolating cell-center velocities. These velocity fields are corrected using equation (22) where the correction term for cell-center velocity is linearly interpolated from the faces. The pressure field is computed by enforcing divergence free condition and the resulting pressure-Poisson equation equation (23) is solved using a conjugate gradient method.

$$\rho^{n+1} \left(\frac{\Delta V}{\Delta t} - a_v \right) u^* = f_{viscous}^* + S \quad (21)$$

$$U^{n+1} = U^{**} - \Delta t \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right)_{face} \quad (22)$$

$$u^{n+1} = u^{**} - \Delta t \left\langle \frac{\nabla p^{n+1}}{\rho^{n+1}} \right\rangle_{face \rightarrow cell}$$

$$\sum_{facesa} \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) \cdot ndA = \frac{1}{\Delta t} \sum_{faces} U^* \cdot ndA \quad (23)$$

Results and Discussion

A set of 3D single rising bubble and coalescence of two rising bubbles have been numerically simulated and compared with the observations in literature to establish the accuracy of the presented implementation. Before that, the spurious velocities observed in static bubble simulations are quantified to gauge some of the performance characteristics of the immersed boundary method and the interface reconstruction procedure.

A. Spurious Velocity 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. With U_{max} as maximum magnitude of velocity in domain for a bubble of diameter d , the magnitudes of these spurious velocities in terms of Capillary number ($Ca = \mu U_{max} / \sigma$) were verified to be of the order of 10^{-4} for tested Laplace numbers ($La = \rho \sigma d / \mu^2$) ranging from 250 to 12000. These magnitudes are found to be consistent with the 3D immersed boundary implementations of Shin and Juric¹⁵ and Sousa et al.¹⁷.

The effect of interface reconstruction on spurious velocity currents is same as in the tests of Shin and Juric¹⁵ who observed temporary spikes in spurious velocities after each reconstruction. For computations performed with $La = 2500$ (Fig. 21), these spikes tended to die down in time unless the reconstruction is performed too frequently depending on the Capillary time scale. ($t_{capillary} = d\mu/\sigma$). Figure 21(a) shows that if reconstruction is performed at every 10 Capillary time step, the temporary increase in spurious velocity tends to die down to the profile exhibited if no reconstruction is performed. However, if the reconstruction is performed at every Capillary time step, the spurious currents were found to be an order of magnitude larger and continued to grow (Fig. 21(a)). Further

observation can be made from Fig. 21(b) which shows the interface normal velocity at a selected marker point. It shows that the oscillatory behavior caused by the surface tension effects quickly dies down if no reconstruction is performed during the simulation. For reconstruction at every 10 Capillary time steps, the amplitude of velocity does not decay but remains stable with mean value of near zero. Reconstruction at every Capillary time step shows a completely different mean value that is several orders of magnitude higher.

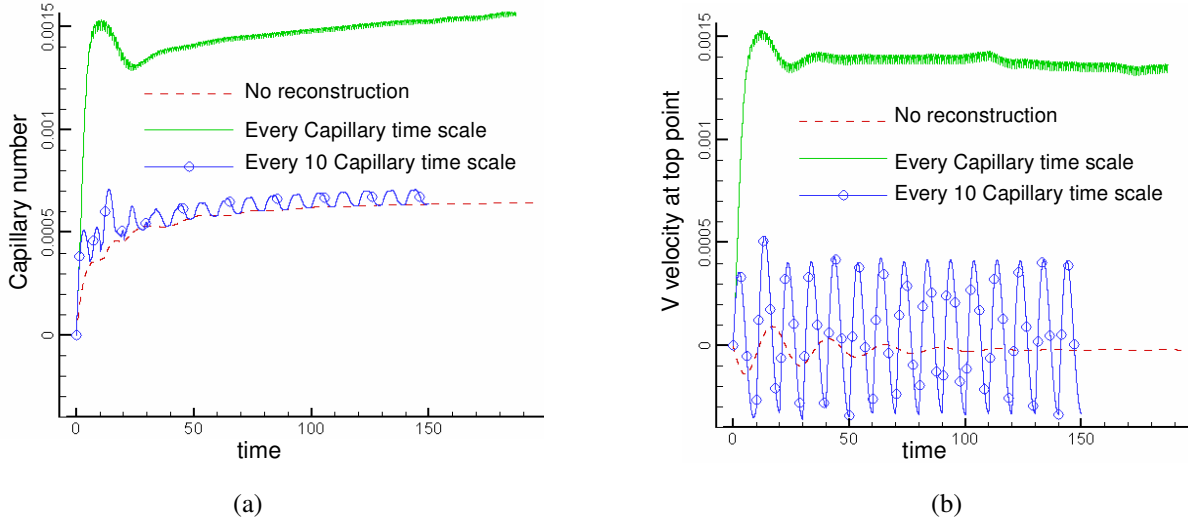


Figure 21: Effect of reconstruction frequency on spurious velocity currents compared with the case of no reconstruction. The time axes are non-dimensionalized using $t_{\text{capillary}} = d\mu/\sigma$ as reference. (a) Evolution of spurious velocity currents; (b) interface normal velocity at a selected marker point.

B. Rising bubbles in different shape regimes

A set of rising bubble simulations were conducted to establish the accuracy of the current implementation of the current adaptive immersed boundary code.

A large body of experimental data on shapes and rise velocities was condensed by Clift et al.²⁹ in the form of a single diagram using Eötvös number ($Eo = g\Delta\rho d^2/\sigma$), Morton number ($M = g\mu^4\Delta\rho/\rho^2\sigma^3$) and Reynolds number ($Re = \rho Ud/\mu$) as the non-dimensional parameters where $\Delta\rho$ is difference in density between fluids outside and inside the interface and these parameters are computed with respect to the properties of fluid outside the interface. The diagram of Clift et al.²⁹ serves as a general guideline for assessing the shape and rise velocity in terms of rise Reynolds number. The set of non-dimensional parameters considered for the simulation are given in the first column Table 3. The computation for $M = 0.971$ and $Eo = 97.1$ belonging to the skirted shape regime (case (d) in Table 3) was performed over two domains: a small domain of size = $(3d, 6d, 3d)$ with three refinement levels over $8 \times 16 \times 8$ base grid (resolution ~ 21 cells per diameter); large domain of size = $(5d, 10d, 5d)$ with three and four levels of refinement over $12 \times 24 \times 12$ based grid (resolution ~ 19 cells per diameter). All the walls of the domain were assigned free-slip boundary condition. As observed in Fig. 22, the effect of small domain size restricted the bubble rise whereas rise Reynolds number for larger domain was found to be in good agreement with computational results of Annaland et al.²⁸. All the computations in Table 3 were performed over $(5d, 10d, 5d)$ domain with three levels of refinement. All the computed shapes (Fig. 23) and Reynolds numbers are in good agreement with computations of Annaland et al.²⁸.

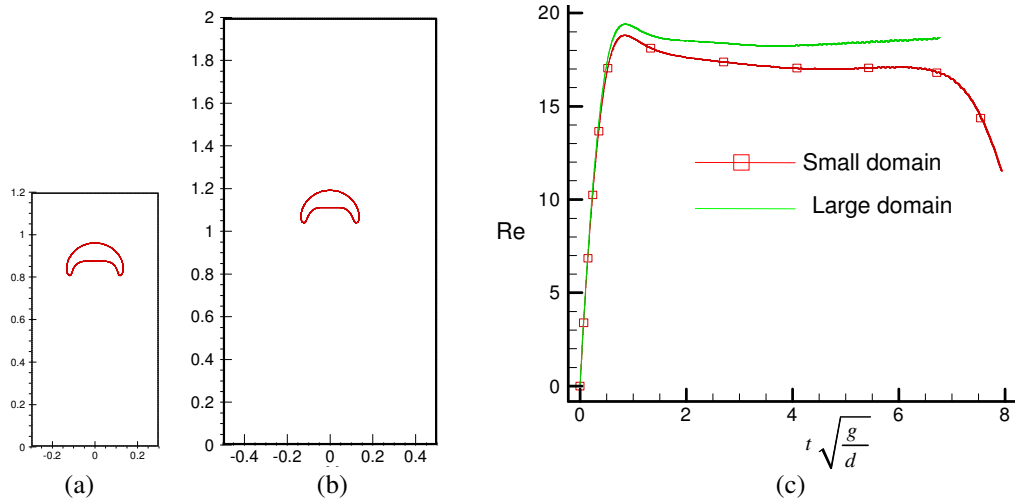


Figure 22: Computation of bubble rise for skirted shape regime ($M = 0.971$, $Eo = 97.1$) over two domain sizes: (a) small domain = $(3d, 6d, 3d)$ with $8 \times 16 \times 8$ base grid and three refinement levels; (b) large domain = $(5d, 10d, 5d)$ with $12 \times 24 \times 12$ base grid and three refinement levels; (c) rise Reynolds number evolution in time.

Table 3: Comparison of computed rise Reynolds number with values from the diagram of Clift et al.²⁹ and computations of Annaland et al.²⁸ for a given Morton and Eotvos number with density and viscosity ratios of 100.

Computational parameters		Expected terminal Shape from Fig. 23(e)	Terminal rise Reynolds number		
Morton number	Eotvos number		Clift et al. ²⁹	Annaland et al. ²⁸	Current Computation
1.26×10^{-3}	0.971	CASE (a): Spherical	1.7	1.6	1.6
1.0×10^{-1}	9.71	CASE (b): Ellipsoidal	4.6	4.3	4.6
1.0×10^3	97.1	CASE (c): Dimpled Ellipsoidal	1.5	1.7	1.6
9.71×10^{-1}	97.1	CASE (d): Skirted	20	18.0	18.7

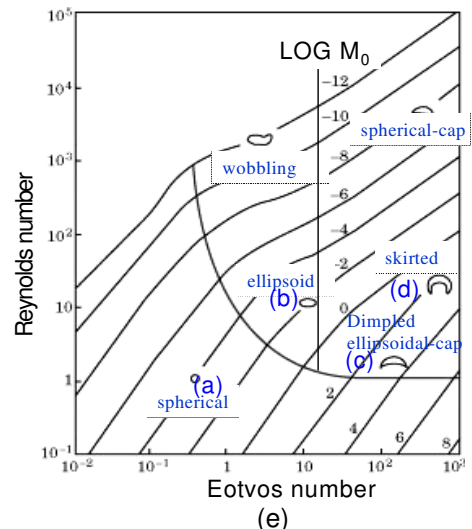
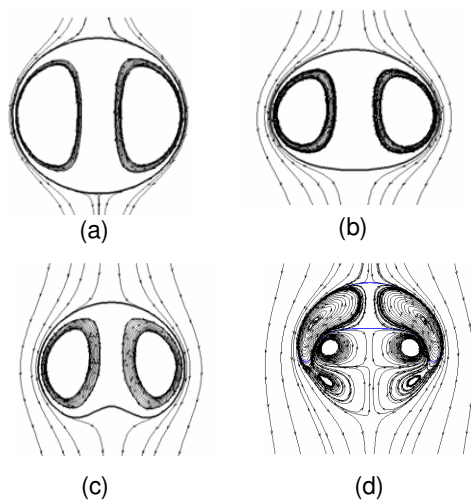


Figure 23: Computed shapes for cases in Table 3 falling in (a) spherical, (b) ellipsoidal, (c) dimpled ellipsoidal and (d) skirted regimes; (e) diagram of Clift et al.²⁹ (picture taken from Annaland et al.²⁸ and altered for clarity) with the corresponding locations of computed shapes.

C. Bubble Coalescence

A case of head-on and off-axis coalescence of two rising bubbles was computed for $M = 2 \times 10^{-4}$, $Eo = 16$, density and viscosity ratio = 100. The computational parameters shown in Table 4 including domain size, maximum grid resolutions were kept the same as in the volume of fluid computations of Annaland et al.²⁸.

Table 4: Computational parameters for head-on and off-axis bubble coalescence test.

Bubble diameter (d)	0.01m
Computational domain (4d,8d,4d)	(0.04, 0.08, 0.04)m
Initial grid: 10x20x10 base grid with three levels of refinement	80x160x80
Fluid properties outside interface (ρ_1, μ_1)	Density (ρ_1) = 1000 kg/m ³ Viscosity(μ_1) = 0.1 kg/ms
Fluid inside interface (ρ_2, μ_2)	Density (ρ_2) = 10 kg/m ³ Viscosity(μ_2) = 0.001 kg/ms
Surface tension (σ)	0.1N/m

1. Head-on coalescence of two rising bubbles

The computational domain and initial grid shown in Fig. 24 has dimensions (4d, 8d, 4d) discretized by three levels of refinement over 10x20x10 base grid resulting in a maximum resolution of 80x160x80 equivalent to twenty grid cells per bubble diameter. The initial centers of the top and bottom bubbles are at (2d, 2.5d, 2d) and (2d, d, 2d) with the origin located at the bottom left corner of the domain. Figure 24(b) and (c) show a slice of the 3D grid at $t = 0s$ and $t = 0.125s$.

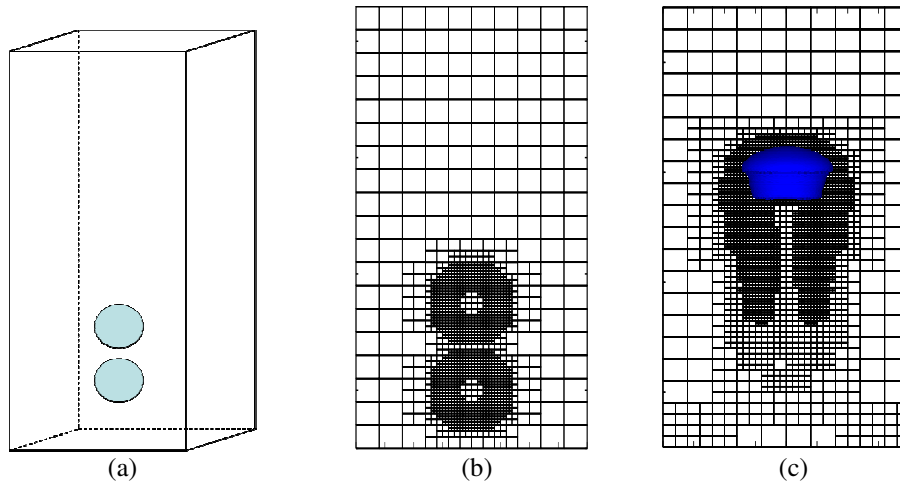


Figure 24: (a) Computational domain of size (4d, 8d, 4d); (b) cross section of the initial 80x160x80 adaptive grid generated by three levels of refinement over 10x20x10 base grid and (c) grid at $t = 0.125s$.

The trailing bubble at the bottom rises faster in the wake of the leading bubble resulting in an eventual collapse. The time history of the computed developments is presented in Fig. 25. It shows the leading bubble acquiring a spherical cap shape before coalescence which is consistent with respect to the shape diagram of Clift et al.²⁹ (Fig. 23(e)) for the chosen computational parameters. The computed results are in good agreement with the simulations of Annaland et al.²⁸ and follow the developments indicated by experimental photographs of Brereton and Korotney³⁰.

During the simulation, the bubble merger was picked at time = 0.116s and Fig. 26 shows the instance of reconstruction along with the background grid resolution for reference. Reconstruction procedure produced a 1.4% phase volume loss which was subsequently recovered by perturbing the markers of the reconstructed interface.

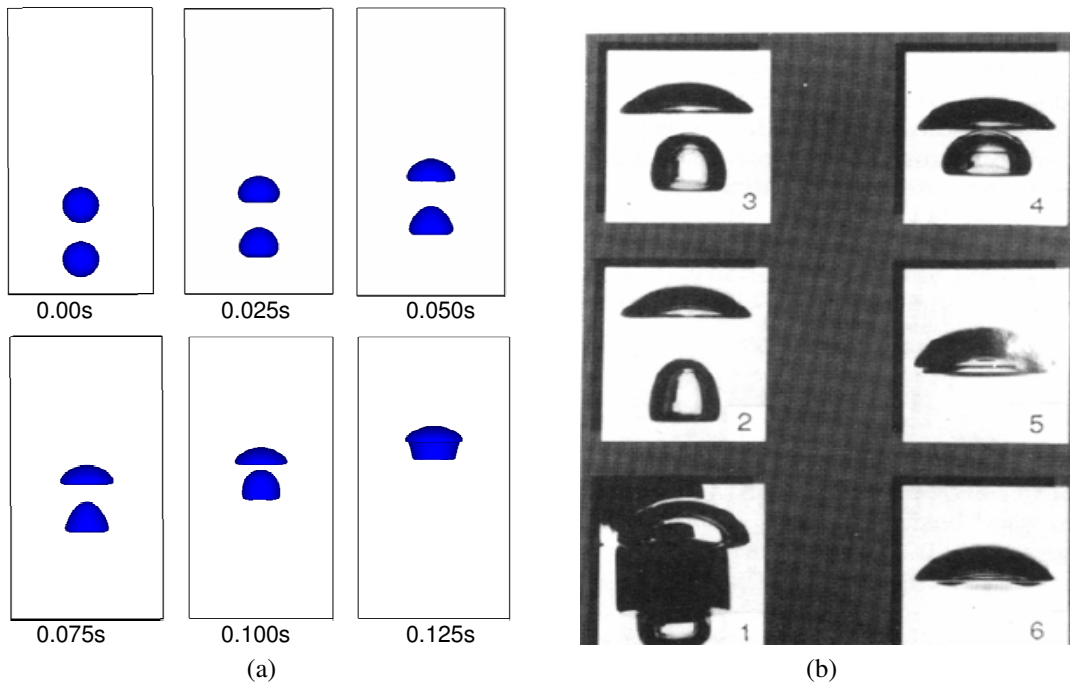


Figure 25: (a) Computed time history of head-on coalescence; (b) experimental photographs by Brereton and Korotney³⁰ (picture taken from Annaland et al.²⁸) with 0.03s time interval.

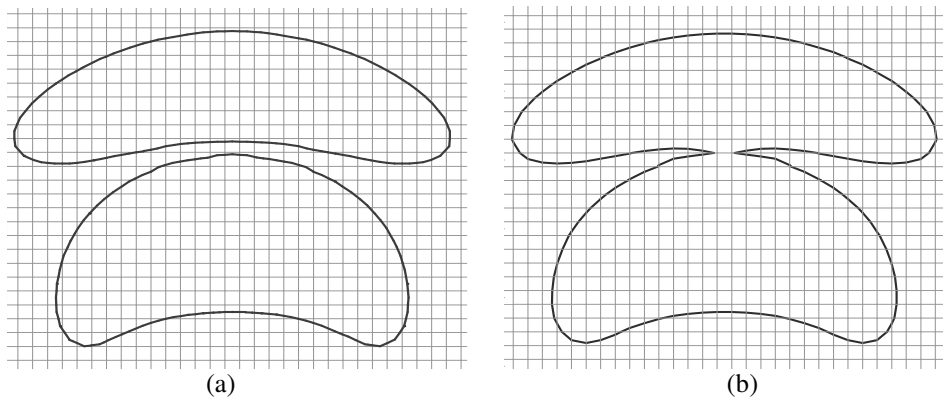


Figure 26: Cross section of the interface at $t = 0.116$ s (a) before reconstruction and (b) after reconstruction.

2. Off-axis coalescence of two rising bubbles

All the computational parameters used for off-axis collision are the same as for head-on coalescence except the initial locations of the two bubbles. The two bubbles are placed at coordinates $(2.8d, d, 2d)$ and $(2d, 2.5d, 2d)$. The computed results in Fig. 27 are in good agreement with the computations of Annaland et al.²⁸ and qualitatively follow the developments shown in the experimental photographs of Brereton and Korotney³⁰. The non-symmetric nature of the problem is further highlighted by the computed shape at 0.200s shown in Fig. 28.

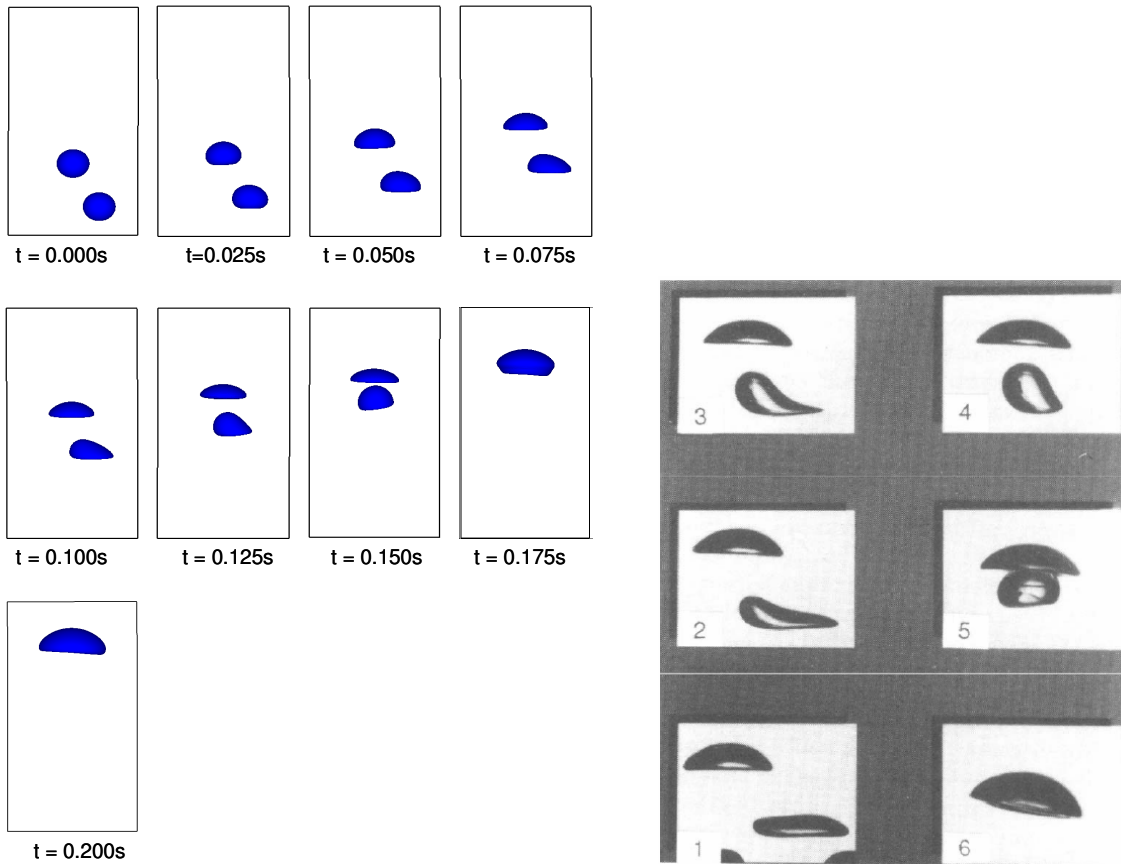


Figure 27: (a) Computed time history of shape for off-axis collision simulation; (b) experimental photographs by Brereton and Korotney³⁰ (picture taken from Annaland et al.²⁸) with 0.03s time interval.

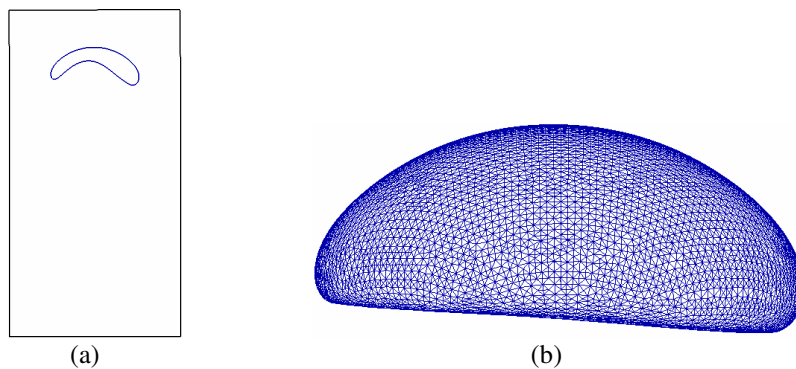


Figure 28: Interface shape at $t = 0.200s$ for off-axis bubble coalescence test. (a) A cross-section of the interface in the computational domain (after coalescence) and (b) the three dimensional triangulated surface grid.

Concluding Remarks

A computational system exploiting the advantages of adaptive Cartesian grids for immersed boundary method has been developed. Three dimensional interfaces are tracked using triangulated surface grid representation. The most serious difficulty associated with explicit marker based tracking involves the handling of topological changes. The connectivity-free tracking with level-contour-based reconstruction of Shin and Juric¹⁵ needs to reconstruct the entire interface to maintain the interface resolution even when no topology change is to be expected. The presented work uses the simplicity of level contour based reconstruction but maintains the interface connectivity information

to perform reconstruction only when topology changes are to be expected based on a simple criteria presented. The interface resolution is maintained at a desirable level by using a conservative restructuring technique that adds and deletes markers any loss of phase volume. The algorithms for interface reconstruction and restructuring were presented and their impacts of the computation were assessed with respect to mass conservation, effect of spurious velocities. The overall implementation was validated using rising bubbles in different shape regimes and coalescence of two bubbles rising due to buoyancy. The present effort has combined different approaches in a unified framework while adding further advancement.

Acknowledgments

The present effort has been supported by the NASA Constellation University Institute Program (CUIP), Ms. Claudia Meyer program monitor.

References

1. Shyy, W., Udaykumar, H. S., Rao, M. M. and Smith, R. W., Computational Fluid Dynamics with Moving Boundaries, Taylor & Francis, Washington, DC, 1996.
2. Shyy, W., Francois, M., Udaykumar, H. S., N'dri, N. and Tran-Son-Tay, R., "Moving Boundaries in Micro-Scale Biofluid Dynamics", Applied Mechanics Reviews, Vol. 54, No. 5, 2001, pp. 405-453.
3. Youngs D. L., "Time-dependent Multi-material Flows with Large Fluid Distortion", Numerical Methods for Fluid Dynamics, edited by K. W. Morton and M. J. Baines, Academic Press, New York, 1982.
4. Scardovelli, R. and Zaleski, S., "Interface Reconstruction with Least-square Fit and Split Lagrangian-Eulerian Advection", International Journal for Numerical Methods in Fluids, Vol. 41, 2002, pp. 251-274.
5. Osher, S. and Fedkiw, R. P., "Level Set Methods: An Overview and Some Recent Results", Journal of Computational Physics, Vol. 169, 2001, pp. 463-502.
6. Tryggvason, G., Bunner, B., Esmaeeli, A., Al-Rawahi, N., Tauber, W., Han, J., Nas, S. and Jan, Y.-J., "A Front Tracking Method for the Computations of Multiphase Flow", Journal of Computational Physics, Vol. 169, 2001, pp. 708-759.
7. Glimm, J., Grove, J. W., Li, X.L. and Tan, D.C., "Robust Computational Algorithms for Dynamic Interface Tracking in Three Dimensions", SIAM J. Sci. Comput, Vol. 21, 2001, pp. 2240-2256.
8. Udaykumar, H.S., Kan, H.-C., Shyy, W. and Tran-Son-Tay, R., "Multiphase Dynamics in Arbitrary Geometries on Fixed Cartesian Grids," Journal of Computational Physics, Vol. 137, 1997, pp.366-405.
9. Francois, M. and Shyy, W., "Computations of Drop Dynamics with the Immersed Boundary Method; Part 1- Numerical Algorithm and Buoyancy Induced Effect", Numerical Heat Transfer, Part B, Vol. 44, 2003, pp. 101-118.
10. Francois, M. and Shyy, W., "Computations of Drop Dynamics with the Immersed Boundary Method; Part 2- Drop Impact and Heat Transfer", Numerical Heat Transfer, Part B, Vol. 44, 2003, pp. 119-143.
11. Enright, D., Fedkiw, R. P., Ferziger, J. and Mitchell, I., "A Hybrid Particle Level-set Method for Improved Interface Capturing", Journal of Computational Physics, Vol. 183, 2002, pp. 83-116.
12. Aulisa, E., Manservigi, S. and Scardovelli, R., "A Surface Marker Algorithm Coupled to an Area-preserving Marker Redistribution Method for Three-Dimensional Interface Tracking", Journal of Computational Physics, Vol. 197, 2004, pp. 455-584.
13. Sussman, M., "A Second Order Coupled Level Set and Volume-of-fluid Method for Computing Growth and Collapse of Vapor Bubbles", Journal of Computational Physics, Vol. 187, 2003, pp. 110-136.
14. Torres, D. J. and Brackbill, J. U., "The Point-set Method: Front-tracking without Connectivity", Journal of Computational Physics, Vol. 165, 2000, pp. 620-644.
15. Shin, S. and Juric, D., "Modeling Three-dimensional Multiphase Flow using a Level Contour Reconstruction Method for Front tracking Without Connectivity", Journal of Computational Physics, Vol. 180, 2002, pp. 427-470.
16. Esmaeeli, A. and Tryggvason, G., "Direct Numerical Simulations of Bubbly Flows, Part II. Moderate Reynolds Number Arrays", Journal of Fluid Mechanics, Vol. 385, 1999, pp. 325-358.
17. Sousa, F. S., Mangiacavacchi, N., Nonato, L. G., Castelo, A., Tome, M. F., Ferreira, V. G., Cuminato, J. A. and McKee, S., "A Front-tracking/Front-capturing Method for the Simulation of 3D Multi-fluid Flows with Free Surfaces", Journal of Computational Physics, Vol. 198, 2004, pp. 469-499.
18. Peskin, C.S., "Numerical Analysis of Blood Flow in the Heart", Journal of Computational Physics, Vol. 25, 1977, pp. 220-252.
19. Al-Rawahi, N. and Tryggvason, G., "Numerical Simulation of Dendritic Solidification with Convection: Three-dimensional Flow", Journal of Computational Physics, Vol. 194, 2004, pp. 677-696.
20. Peskin, C. S. and McQueen, D. M., "Fluid Dynamics of the Heart and its Valves", Case studies in Mathematics Modeling: Ecology, Physiology, and Cell Biology, edited by H. G. Othmer, F. R. Adler, M. A. Lewis and J. C. Dallon, Prentice-Hall, Englewood Cliffs, NJ, 1996.
21. Prosperetti, A., "Numerical Algorithms for Free-surface Flow Computations: An Overview", Drop-Surface Interactions, edited by M. Rein, Springer, 2002.
22. Singh, R. K., N'Dri, N., Uzgoren, E., Shyy, W. and Garbey, M., "Three-dimensional Adaptive, Cartesian Grid method for multiphase flow computations", AIAA Paper-1390, 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV., 2005.

23. Aftosmis, M. J., Solution Adaptive Cartesian Grid Method for Aerodynamic Flows with Complex Geometries, Computational Fluid Dynamics VKI Lectures Series, Belgium, 1997.
24. Ham, F. E., Lien, F. S. and Strong, A. B., "A Cartesian Grid Method with Transient Anisotropic Adaptation", Journal of Computational Physics, Vol. 179, 2002, pp. 469-494.
25. Wang, Z.J., "A Quadtree-based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equations", Computers & Fluids, Vol. 27, 1998, pp. 529-549.
26. Kim, J. and Moin, P., "Application of a Fractional Step Method to Incompressible Navier-Stokes Equations", Journal of Computational Physics, Vol. 59, 1985, pp. 308-323.
27. Ye, T., Mittal, R., Udaykumar, H. S. and Shyy, W., "An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries", Journal of Computational Physics, Vol. 156, 1999, pp. 209-240.
28. Annaland, M. V. S., Deen, N. G. and Kuipers, J. A. M., "Numerical Simulation of Gas Bubbles Behavior using a Three-dimensional Volume of Fluid Method", Chemical Eng. Science, Vol. 60, 2005, pp. 2999-3011.
29. Clift R., Grace J. R. and Weber M., Bubbles, Drops and Particles, Academic Press, New York, 1978.
30. Brereton, G., Korotney, D., "Coaxial and Oblique Coalescence of Two Rising Bubbles", Dynamics of Bubbles and Vortices Near a Free Surface, edited by I. Sahin and G. Tryggvason, Vol. 119, ASME, New York, 1991.