Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 213 (2006) 613–628
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A simple package for front tracking

Jian Du [a,b,1], Brian Fix [b,1,2], James Glimm [a,b,1,3,4], Xicheng Jia [a,1], Xiaolin Li [a,*,2], Yuanhua Li [a,1], Lingling Wu [a,1]

[a] *Department of Applied Mathematics and Statistics, University at Stony Brook, Stony Brook, NY 11794-3600, United States*
[b] *Computational Science Center, Brookhaven National Laboratory, Upton, NY, 11793, United States*

Received 20 January 2005; received in revised form 25 August 2005; accepted 27 August 2005
Available online 20 October 2005

Abstract

A general purpose software package for the geometry and dynamics of an interface has been extracted from the FronTier code developed by the authors and colleagues, and is now publicly available. The purpose of this paper is twofold. We describe significant improvements to the Front Tracking package, especially in the 3D handling of topological bifurcations. We also assess the performance of the package, in comparison with publicly distributed interface codes (the level set method), with published performance results (VOF and other methods) and with previous versions of front tracking.

The major new algorithm presented here is Locally Grid Based tracking, which combines the best features of two previous 3D tracking algorithms. It combines the robustness of grid-based tracking with the accuracy of grid free tracking, and thus it is a significant improvement to both of these algorithms. We also discuss the surface curvature and normal algorithms and a higher order propagation algorithm, used for the comparison studies presented here.

The code is downloadable from the web, and is accompanied by a web-based testing and evaluation site and extensive web-based documentation.

Keywords: Front tracking; FronTier; Interface geometry; Topological bifurcation

1. Introduction

Systematic comparisons of algorithms for the numerical modeling of interfaces were performed by Rider and Kothe [17,18]. Tests show superior performance for particle methods, followed by PLIC-VOF, level sets and capturing, in that order. The purpose of this paper is to extend this comparison to front tracking,

* Corresponding author. Tel.: +1 516 6328352; fax: +1 516 6328490.
E-mail address: linli@ams.sunsyb.edu (X. Li).
[1] Supported in part by the U.S. Department of Energy DE–FG02–90ER25084.
[2] Supported in part by the U.S. Department of Energy, Grant DE–FC02–01ER25461.
[3] Supported in part by the National Science Foundation, Grant DMS–0102480.
[4] Supported in part by the U.S. Department of Energy, Grant DE–AC02–98CH1-886.

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.08.034

to present new (3D) front tracking algorithms and to compare the new to previous front tracking algorithms. In this extended comparison, front tracking is comparable or superior to particle methods, and superior to the others.

Front tracking relies on marker particles, and of the four groups of methods mentioned above, it is closest conceptually to the marker particle methods. It differs from marker particles in that the particles are located only on the interface, rather than in a volume region near the interface, and in that the particles are connected to each other, to form a triangulation (3D) or piecewise linear (2D) description of the interface. It is significantly faster than particle methods, since fewer particles (one or two in 2D) are used per cell in front tracking than the number used (4^D) in typical particle method simulations. 2D bifurcations of interface topology in front tracking were resolved a number of years ago [5], while a robust, accurate treatment of 3D bifurcations is the central feature of the new front tracking algorithms presented here. We address the complexity issue through public release of the front tracking code and through its modularization into user callable libraries. Web-based documentation supports this release.

The front tracking method has showed its advantage in the computation of several important physical problems such as the study of fluid interface instabilities [7,2,9,10,20], providing the first or the only physically validated simulation for some important fluid instability problems.

Front tracking starts from a set of discrete marker points, topologically organized through an interface data structure. The method provides a set of functions to maintain its organization after dynamic propagation and bifurcation [6,8,3]. For use below, we call this method grid free (GF) tracking, as the interface handling has no logical relation to a finite difference grid. The difficulties associated with topological bifurcations for GF tracking are amplified in 3D. As a result, a robust semi-Eulerian reconstruction method, called grid-based (GB) tracking [4], was introduced. GB tracking, although robust, suffered from excessive interpolation and smoothing errors, the same inaccuracies as the level set method. Here we present a new method which combines the best features of GF and GB tracking, called the locally grid-based (LGB) front tracking method. It is explained in detail here. The essence of LGB tracking is to use accurate GF propagation except locally in space and time where bifurcations occur. In the bifurcating regions, we use the GB method. The treatment of the surface normal and curvature given here and the higher order front propagation are also new for front tracking.

A similar method to the front tracking is the boundary integral method [13,16,1,11] which also describes the moving surface by Lagrangian marker points. The major difference between the front tracking method and the boundary integral method lies in the calculation of the velocity of the markers. To handle the topological bifurcation, the boundary integral method is coupled with the level set method and the volume of fluid method.

2. Static interface functions

In the front tracking method, we describe the interface as a discretized geometrical manifold represented by a set of topologically linked marker points. In 3D, the interface is made up of points, curves, and surfaces. Each of these geometrical objects (except the points) have pointers to the objects which form its boundary, and to the objects it bounds. The 2D data structure is similar, but simpler as it consists of points and curves only. The start and end points of a curve, called nodes, have additional structure. The curves and surfaces are made of linear segments (simplexes). Each linear segment has pointers to its neighbors [8,3].

2.1. Interface initialization

Initialization of the interface geometry is made by calls to a set of functions such as make_curve() and make_surface(). The call of make_curve() requires setting the start and end nodes, as well as the left and right components (two integers each representing the index of the two regions the curve bounds). The curve has a pointer to the first and the last elements (bonds). The internal bonds can be filled by consecutively inserting points into existing bonds.

A similar method can be used for a three dimensional surface. The boundary and its surface are initially represented by two triangles. We iteratively insert boundary and interior points into the existing triangles, thereby splitting them, to define the full surface.

The GB method provides an general initialization for the interface [14,4]. This method is important for the present paper as it provides the local part of the LGB algorithm. It uses the structured rectangular grid as the skeleton and finds the crossings of the interface curves (2D) or surfaces (3D) with the grid edges. The interface elements are constructed in each rectangular block from this crossing data. Isomorphically, there are three distinct cases in 2D and 14 distinct cases in 3D. After creating the elements of the manifold, each block is checked with its neighbors for possible topological linkage of the elements. Due to the constraint of crossings on the block edges, the geometrical elements may not possess high quality. A redistribution of the interface is used to optimize the bonds in 2D and triangles in 3D.

2.2. Interface optimization

The front tracking method provides several algorithms for the redistribution of marker points on curves (2D) and surfaces (3D). The most frequently used method in 2D is the equal-bond redistribution. This function measures the total length of the curve, which is then divided by the optimal bond length to obtain N, the total number of bonds. The new bond list is created with equal length starting from the first point (node).

In three dimensions, both the area and the aspect ratio of the triangles are calculated and those with low quality are placed in a queue. A cyclic optimization procedure is called for elements of this queue to insert, delete or re-triangulate the triangles until all the triangles satisfy a preset geometrical criterion. Each of the optimization functions performs a complete set of operations to guarantee that the topological linkage of the interface is correct after its operation.

2.3. Interface geometry

A new formula by Max [15] is used to compute the normal at a vertex of the discretized 3D surface. It is expressed as a weighted sum of the normals to the facets surrounding the vertex. The key point is to assign large weights for smaller facets; it is especially suitable when the facets surrounding the vertex differ greatly in size. Both this new algorithm and our previous one use only points adjacent to the point at which the normal is evaluated, and so both have a stencil radius of Δx. Our curvature algorithm also has approximately this same stencil.

Suppose O is a vertex of the polyhedron which is located at the origin and has adjacent vertices $\mathbf{V}_0,\mathbf{V}_1,\ldots,\mathbf{V}_{n-1}$. The normal of the triangle $\mathbf{V}_i O \mathbf{V}_{i+1}$ can be calculated as

$$\mathbf{N}_i = \mathbf{V}_i \times \frac{\mathbf{V}_{i+1}}{|\mathbf{V}_i||\mathbf{V}_i + \mathbf{1}| \sin\alpha_i}, \tag{1}$$

where α_i is the angle between $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$. The Max weight of the $\mathbf{N}_i$ in the calculation of $\mathbf{N}$ is $\sin\alpha_i/(|\mathbf{V}_i||\mathbf{V}_{i+1}|)$. That is

$$\mathbf{N}(O) = \sum_{i=1}^{K} \frac{\mathbf{N}_i \sin\alpha_i}{|\mathbf{V}_i||\mathbf{V}_{i+1}|} = \sum_{i=1}^{K} \mathbf{V}_i \times \frac{\mathbf{V}_{i+1}}{|\mathbf{V}_i|^2 |\mathbf{V}_{i+1}|^2}. \tag{2}$$

This new algorithm for calculating the interface normal has been compared with the old area weighted algorithm

$$\mathbf{N}(O) = \frac{1}{A} \sum_{i=1}^{K} \mathbf{N}_i A_i, \tag{3}$$

where A_i is the area of the ith surrounding triangle and A is the summation of A_is. Table 1 shows the results of the comparison. We note that comparison results are sensitive to surface regularity, and that the Max algorithm is specifically superior for regular (nearly spherical) surfaces.

We also revised the calculation of curvature for a three dimensional surface. By the similar procedure described in [21], we first construct a rotated principal coordinate frame with the origin at vertex O, where curvature is to be estimated. The z axis is aligned with the surface normal and x and y axes are chosen arbitrarily within the plane passing O and normal to z. Then after mapping the surface data into the new coordinate system, we approximate the surface as a quadratic surface

Table 1
Comparison of mean curvature and normal between the Nelson Max method and the area weighted method

Method	Mesh size	$\|\kappa - \kappa_e\|_1/\|\kappa_e\|$	$\|\|\vec{n} - \vec{n}_e\|\|_1$
Sphere radius = 0.1			
Max with	$40 \times 40 \times 40$	0.05700	1.1e − 5
Quadratic fitting	$80 \times 80 \times 80$	0.01440	2.8e − 5
Area weighted	$40 \times 40 \times 40$	0.10885	0.0863
Integral method	$80 \times 80 \times 80$	0.13996	0.0506
Sphere radius = 0.3			
Max with	$40 \times 40 \times 40$	0.0061	3.6e − 5
Quadratic fitting	$80 \times 80 \times 80$	0.0015	2.3e − 5
Area weighted	$40 \times 40 \times 40$	0.1149	0.0323
Integral method	$80 \times 80 \times 80$	0.1487	0.0154

$$z = Ax^2 + Bxy + Cy^2 + Dx + Fy. \tag{4}$$

To determine the coefficients, at least five neighboring vertices must be included. If the number of neighboring points is less than five, we add the closest next to the adjacent points. If the number of neighboring points is greater than five, least square fitting is used to determine the coefficients A, B, C, D, E. The curvature of the quadratic surface is used as the approximation for the surface curvature at O. The improved definition for the normal yields an improved curvature. See Table 1, from which we infer a second order rate of convergence for κ, whereas the previous method appears to be $\mathcal{O}(1)$, i.e., bounded but not convergent.

3. Interface propagation functions

In addition to the static interface library, the dynamic front package provides a set of functions for the propagation of the interface in a given velocity field and for geometry-dependent interface motion. The driver function advance_front() contains preset pointers to functions based on dimension and algorithms selected by the user. This function will also detect the geometrical and topological correctness of the interface after the propagation. When necessary, it will perform a sequence of operations to guarantee the soundness and quality of the interface before the next propagation.

3.1. Interface propagation

The front package allows users to provide their own velocity function for the propagation of the interface. To do so, the user needs to create a data structure containing all necessary parameters for the velocity function. This data structure is casted into a pointer after it is initialized. It will be passed to the velocity function, which is also a pre-assigned anonymous pointer. Inside the velocity function, the pointer of velocity parameters will be cast back to point to the original data structure and the velocity function retrieves its parameters for the computation of velocity of each marker point.

There are two point propagation functions. The first type of the point propagation is for a velocity field which is a function of space coordinates and time only

$$\mathbf{v} = \mathbf{v}(\mathbf{x}, t). \tag{5}$$

In such a velocity field, the moving front is advanced through the propagation of the discretized interface points via a simple ordinary equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t). \tag{6}$$

The package provides the first order Euler forward and the fourth order Runge–Kutta methods to solve Eq. (6).

The second type of front propagator is the hyper-surface propagator. This propagation function is needed when the velocity of the front is not a pure field function of space and time, but is dependent on the front geometry as well. Typical examples are the normal propagation of a front and curvature dependent front propagation,

$$\mathbf{v} = \mathbf{v}(\mathbf{x}, t, \mathbf{N}, \kappa). \tag{7}$$

Since as a marker point propagates, so do its neighbors and thus N and κ change. For each sub-step in the Runge–Kutta method, to ensure a uniform order of truncation error, the geometry variables (normal and curvature) must be updated. Therefore, the surface must be advanced as a whole in each sub-step.

3.2. 2D Interface bifurcation

For the front package, the geometrical optimization and topological bifurcation are automatic. These operations are wrapped under the driver function advance_front() and are invisible to the user. Unless instructed for special operations, after each time step, the user will obtain from the driver an interface which has already been optimized and resolved.

The 2D topological bifurcation algorithm is grid free [5]. It starts by hashing the curve elements (bonds) to the rectangular mesh blocks. A intersection check is performed over each block with tangled bonds recorded. Temporary nodes inserted at the intersections cut the curve into new partitions. A component check is performed to determine the unphysical section of the curves followed by the removal of the redundant segments and temporary nodes.

4. Locally grid-based bifurcation

4.1. Grid free and grid-based tracking compared

In Fig. 1, we show a comparison of an interface under purely Lagrangian GF propagation with the same problem solved via GB tracking. We start with the same initial interface, an ellipsoid, and propagate with a spiraling velocity field whose angular velocity is described by

$$\omega(r) = \omega_0 + kr, \tag{8}$$

where $r = \sqrt{x^2 + y^2}$ is the distance to the spiral center. From the comparison we see that the propagation with Eulerian reconstruction introduces a large error in regions where the curvature of the surface is large.

The one step error due to reconstruction of the interface can be analyzed via Fig. 2. Let the local error in the front position be e_l, the local mean curvature be κ_l, the grid spacing $\Delta x = \Delta y = \Delta z = h$, and let d represent the

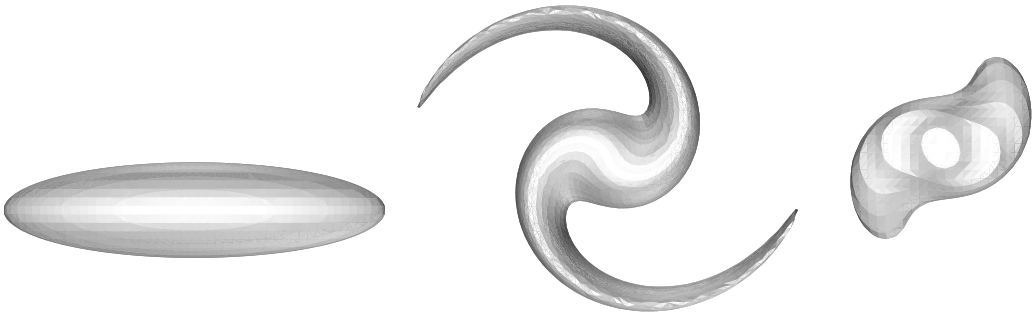

Fig. 1. Comparison of GF and GB interface propagation using a common 80^3 mesh. Left: Common initial initial conditions (an elongated ellipsoid). Center: Lagrangian (grid free) propagation. Right: Eulerian (grid based) propagation. The plots are after 500 steps in a spiraling velocity field. which is after about 1.6 rounds of revolution at the center and 2.9 rounds of revolution at the outer edge (the angular velocity is a linear function of r).

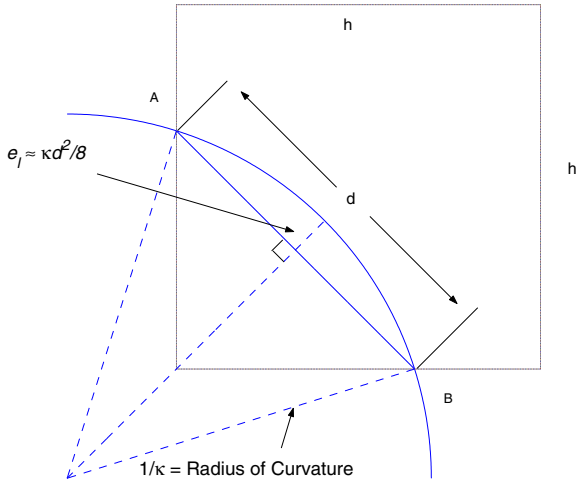

Fig. 2. Grid-based interface reconstruction introduces a truncation error of the interface position which is of $O(h^2/\Delta t)$.

distance between two crossings of the interface with the grid edges. The replacement of the interface within the grid block by a linear segment results in an $O(h^2)$ error in the position of the front:

$$e_l = 1/\kappa_l - \sqrt{(1/\kappa_l)^2 - (d/2)^2} \approx \frac{\kappa_l d^2}{8} \leqslant \frac{\kappa_l h^2}{4}, \tag{9}$$

when $\frac{1}{\kappa_l} \gg \frac{d}{2}$. Assuming $\kappa^n_{\max} = \max(\kappa_l)$ is the upper bound of the local mean curvature at time step n, we obtain an L_∞ error bound of the reconstruction at this time step:

$$E^n_\infty = \frac{\kappa^n_{\max} h^2}{4}. \tag{10}$$

Let $P^n(x)$ be the position of the interface at time step n, so that $P^0(x)$ is the position of the initial interface. Let Δt being the time step. From equation above, we have

$$\left\|P^{n+1}(x) - P^n(x)\right\|_\infty \leqslant E^n_\infty. \tag{11}$$

Therefore,

$$\left\|P^{n+1}(x) - P^0(x)\right\|_\infty \leqslant \sum_{i=1}^{n} E^i_\infty = \frac{h^2 \sum_{i=1}^{n} \kappa^i_{\max}}{4} = O\left(\frac{h^2}{\Delta t}\right). \tag{12}$$

This error is inversely proportional to Δt, which means that for a given grid size, reducing the time step will increase the truncation error. Assuming a CFL determined time step, the method is first order accurate.

The error can be assessed quantitatively using a velocity map for a certain time interval followed by its inverse (the negative of v in place of v). The analytic solution is then the initial conditions (taken to be a sphere) and errors computed by an L_1 norm are easy to determine. In Table 2, we compare grid-based propagation errors to locally grid-based propagation errors (with fourth order propagation), and a velocity field given by a single vortex:

Table 2
Comparison of the L_1 error for the 3D Front Tracking method with grid based and locally grid based algorithms

Case	Grid based	Locally grid based
$T = 4$	5.70×10^{-3}	2.47×10^{-4}
$T = 8$	1.22×10^{-2}	3.37×10^{-4}

The simulation is in a single vortex velocity field Eq. (13) followed by a reversal to the initial conditions. The simulations are performed in a $1 \times 1 \times 1$ domain with a 100^3 computational mesh. The initial interface is sphere of radius 0.15 centered at (0.5, 0.75, 0.5).

$$\begin{cases} u(x,y,z) = -\sin(2\pi y)\sin^2(\pi x)\cos\left(\frac{\pi t}{T}\right), \\ v(x,y,z) = \sin(2\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right), \\ w(x,y,z) = 0.0. \end{cases} \tag{13}$$

The initial interface is a sphere of radius 0.15 centered at (0.5, 0.75, 0.5) and the computational domain is a $1 \times 1 \times 1$ cube. The result of the comparison is shown in Table 2. We observe that the local grid-based error is one order of magnitude smaller than the grid-based error (see Fig. 3).

4.2. Locally grid-based tracking

To reduce the GB interface interpolation error, we introduce a new method, LGB, or the locally grid-based tracking, which combines the advantages of both methods. We use the fully Lagrangian GF method to propagate the interface to obtain an accurate solution of the interface position. Eulerian GB reconstruction of the interface is only used in small regions where topological bifurcation is detected. The detection of topological changes is through a fast algorithm which walks through the Eulerian grid to check the consistency of the indices assigned to grid nodes of each subdomain and the corresponding side of the interface. In the first step of the procedure, the intersections between the interface and cell edges of the Eulerian grid are inserted and the index of every subdomain is assigned to each corner point of the Eulerian grid. We then check the consistency between the indices of each crossing point and the node point it faces. If inconsistency is detected, the corresponding mesh block is recorded. This intersection detection algorithm is approximate in that it will miss bifurcations totally internal to a single mesh cell.

The construction advances through four steps.

1. Those recorded blocks will be assembled to form boxes and overlapping boxes will be merged.
2. Surgery is performed within each box. Triangles crossing the box boundaries are recorded for later use. All the triangles inside the box or attached to the box will be deleted leaving only the crossing points of the interface and grid edges. A grid-based reconstruction is followed to build the new section of the interface inside the box.
3. The triangles totally outside the box form the exterior interface.
4. The region between the exterior and interior interfaces is re-triangulated to join the two smoothly.

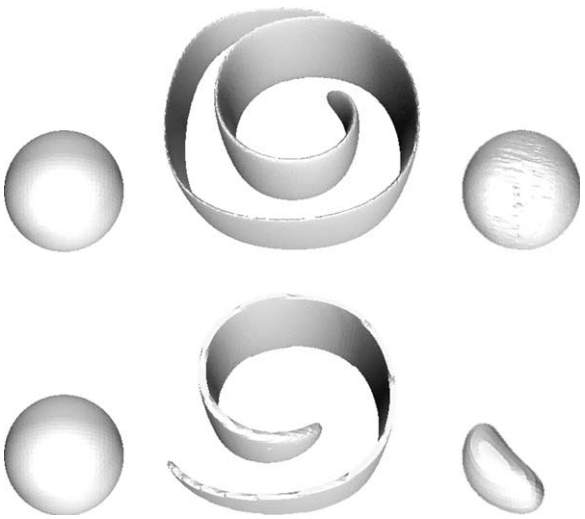

Fig. 3. Comparison of LGB and GB in a single vortex velocity field with reversal. using a common 100^3 mesh. The initial interface is sphere of radius 0.15 centered at (0.5, 0.75, 0.5). The upper row shows the locally grid-based tracking and the lower row shows the grid-based tracking. From left to right are $t = 0$, 4, 8, respectively.

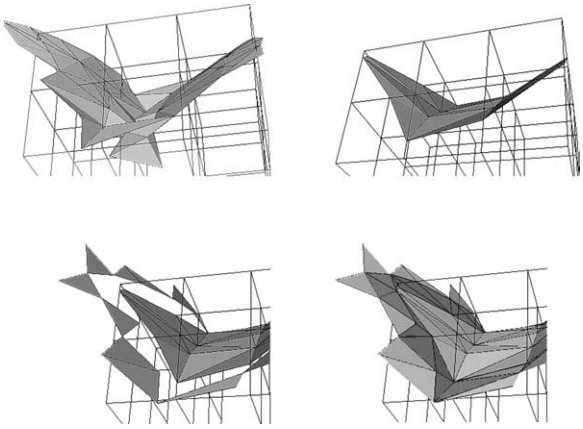

Fig. 4. Steps to reconstruct a tangled section of the three dimensional interface. From left to right and top to bottom: (1) assemble blocks which contain unphysical edges; (2) delete triangles attached to the box and rebuild the interface through the grid-based method, using the grid-based method to reconstruct the interface topology inside the box; (3) align triangles inside the box and outside the box; (4) relink the interface topology for triangles inside and outside the box, and thereby obtain the final interface with new topology.

Fig. 4 shows the procedural steps of the local reconstruction of the interface. The reconnection step 4 is the most crucial step in the LGB method. We modify the triangles recorded in step 2 by a series of steps to make the triangles in the reconnection region also grid based relative to the box boundary. These substeps of step 4 are summarized as follows:

4.1 We split triangles crossing the box boundary which are recorded in step 2 using the intersection points between triangles and box boundaries. There are two types of intersection points: one is the intersection points between triangles and grid cell edges on the box boundary (TYPE I), the other is the intersection points between the sides of triangles and the box boundaries (TYPE II). We first divide each triangle by recursively inserting the TYPE I intersection points (if any) inside the triangle. Each triangle will be divided into three smaller triangles by joining the TYPE I intersection point with the vertices of the triangle. Then we insert the TYPE II intersection points recursively. Each triangle is divided into two smaller triangles by joining the TYPE II intersection point and the vertex opposite to the side containing this intersection point. After this step, the triangles which crossed the box boundaries are split into triangles lying either entirely inside the box or entirely outside the box. We keep only the triangles entirely outside the box. These triangles meet the box surface along a closed curve which is actually a piece-wised linear curve connecting all the intersection points.

4.2 The curve gives an ordering to these triangles, and in this ordering, we merge triangles whose vertices are TYPE II intersection points until there is no TYPE II intersection points in the curve. After this step it is sufficient to examine that each triangle meets the box boundary only as a line joining adjacent grid edge. i.e., the triangle meets the box in a grid-based manner.

After this operation, all triangles will meet the box edge only as a line joining adjacent grid edges. That is, all triangles, inside and outside, meet the box only in a grid-based manner. The reconnection between the outside triangles and the newly reconstructed triangles inside the box is then a simple match of the triangle sides at the box boundaries.

This method reduces the use of the Eulerian reconstruction to a minimum. It is particularly useful for the computation of interface motion in which the interface has regions of large curvature. It reduces interface interpolation errors and minimizes the unphysical disappearance of the fragmented components of the material after bifurcation.

5. Comparison with other interface methods

5.1. Comparison with the level set method

Comparison of the Lagrangian front tracking method and the Eulerian level set method has focused on some important geometrical properties of the interface. The truncation error of the level set method has following sources (1) interpolation of the level set function, (2) re-initialization of the level set function, and (3) extrapolation of the velocity field when the velocity is a function of the geometry (normal direction, curvature) only. Such errors are amplified in the neighborhood of a highly curved segment of the front. The resolution of the level set method is dependent on the Eulerian mesh size and the order of the PDE solver for the level set equation. We have conducted two tests used by Fedkiw et al. The level set code comes from http://www.cs.ubc.ca/mitchell/ToolboxLS.

5.1.1. Rotating slotted disk

A common test for the quality of front propagation is the rigid rotation of a slotted disk. Two sets of comparison are made between the FronTier code and the level set ToolBox package. The first comparison uses the first order Euler forward scheme for the point propagation in the front tracking method and the first order scheme for the solution of the level set functions. The radius of the disk is $R = 0.5$, the width and depth of the slot in the disk are $W = 0.1$ and $H = 0.4$, respectively. A CFL condition is applied to both runs. After one circulation, the slot in the level set solution is completely flattened and the disk has shrunk by about 15%. In the front tracking computation, the radius of the radius of the disk has expanded by 5%, while its slot is still well maintained (see Fig. 5).

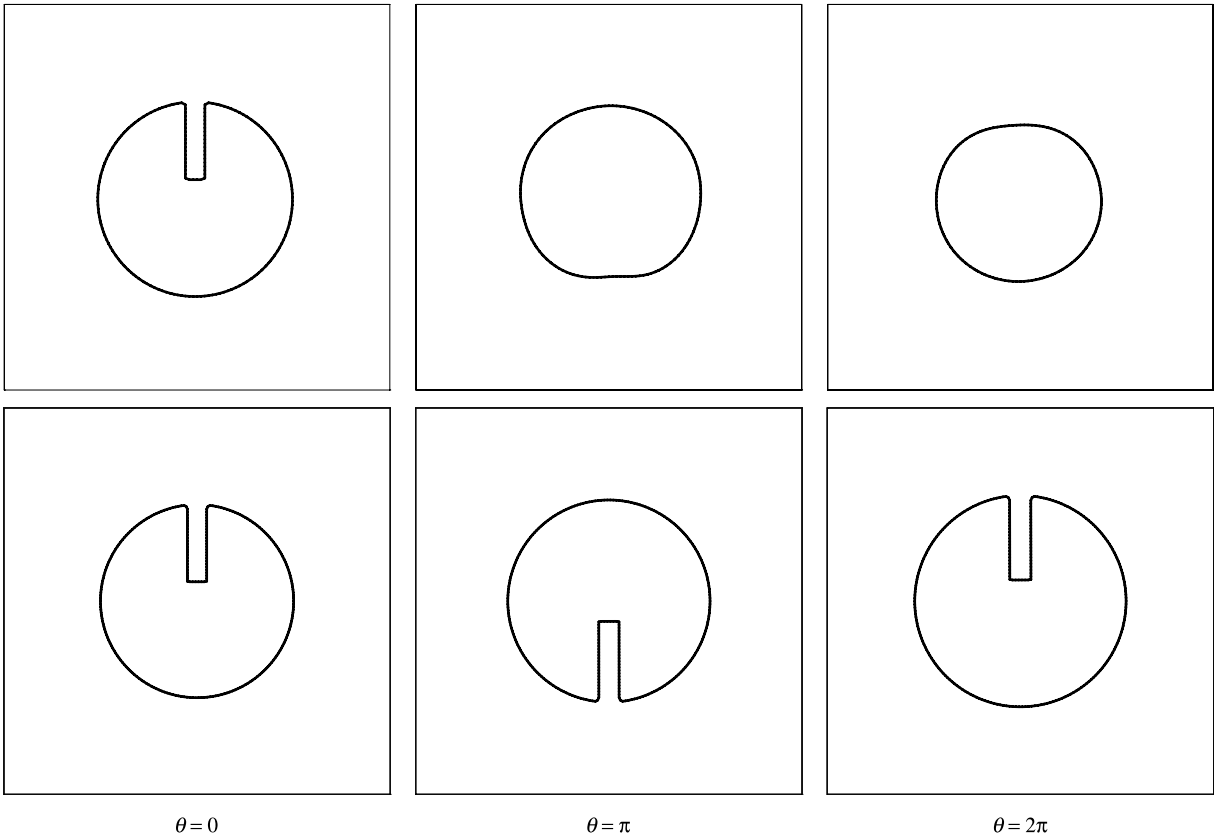

Fig. 5. Comparison of slotted disk simulation using first order methods. The upper sequence shows the result of the level set method while the lower sequence shows the result for the front tracking method. The grid for both is 100×100.

In the second comparison, we used the fifth order WENO scheme for the convection of the level set function, while for the front tracking code, we used the fourth order Runge–Kutta method for the point propagation. The difference after the first round of circulation was not evident. So we continued the computation for 13 revolutions. The fourth order Runge–Kutta method appears to be extremely accurate in the front tracking simulation. Even after 13 rounds of rotation, the change of both radius of the disk and the slot are invisible. The level set computation began to show edge smoothing after the second rotation. At the end of the 13th circulation, the slot is closed at the top resulting in a topologically incorrect bifurcation (see Fig. 6).

5.1.2. Swirling ellipsoid and reversal

Another set of tests comparing the resolution between level set method and the front tracking method is to place an ellipsoid (ellipse in 2D) in a swirling velocity field and then reverse the velocity direction when it reaches a preset time T. In a 2D example, we place a ellipse in a dipole velocity field:

$$\mathbf{v} = \frac{\mathbf{k} \times (\mathbf{r} - \mathbf{r}_1)}{|\mathbf{r} - \mathbf{r}_1|^2} w_1 - \frac{\mathbf{k} \times (\mathbf{r} - \mathbf{r}_2)}{|\mathbf{r} - \mathbf{r}_2|^2} w_2, \tag{14}$$

where w_1 and w_2 are the strength of the vortices, and $\mathbf{r}$, $\mathbf{r}_1$ and $\mathbf{r}_2$ are the position vectors of the point, the center of the first vortex and the center of the second vortex. In the example shown in Fig. 7, we placed the center of the ellipse at the origin, while the two radii of the ellipse are $a = 0.4$ and $b = 0.2$, respectively. The center of the two vortices are at $(-0.3, 0)$ and $(0.3, 0)$ and the strengths of the two vortices are $w_1 = w_2 = 0.1$.

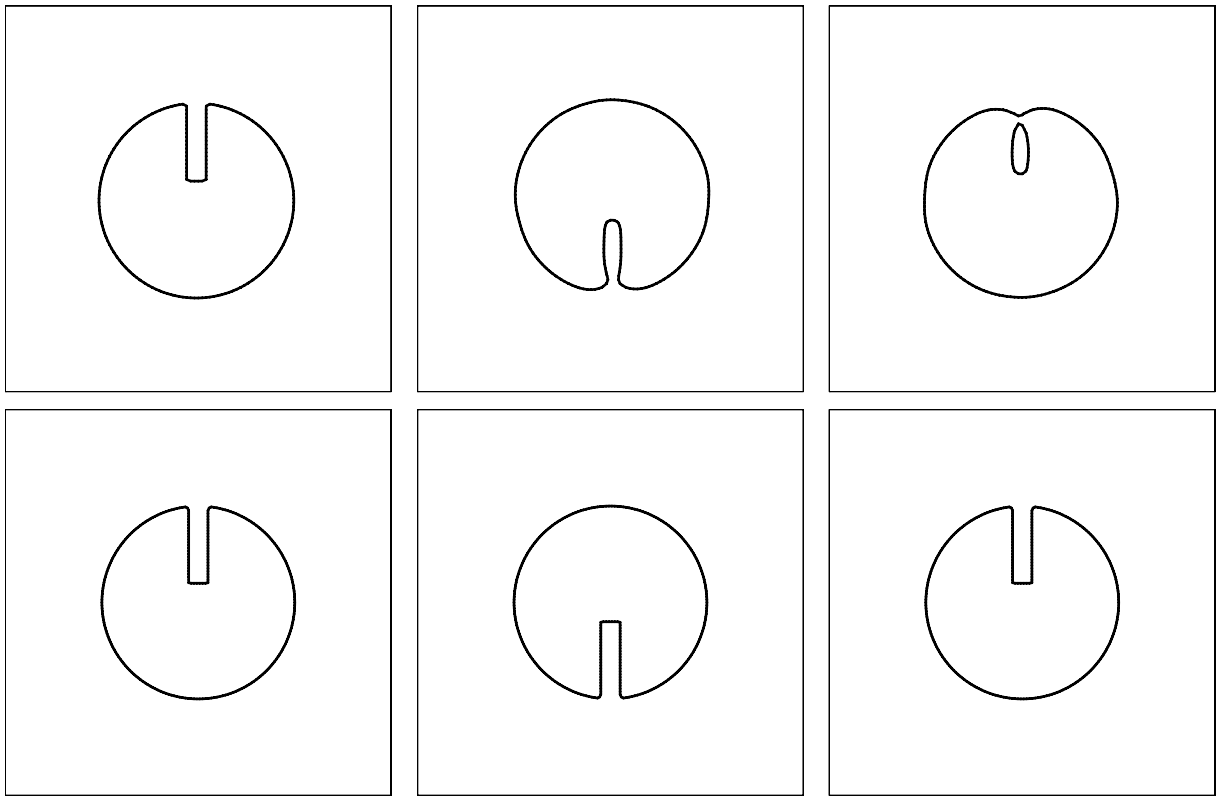

Fig. 6. Comparison of slotted disk simulation using high order methods. The upper sequence shows the result of the level set method using the fifth order WENO scheme and the lower sequence shows the result of front tracking using the fourth order Runge–Kutta method.

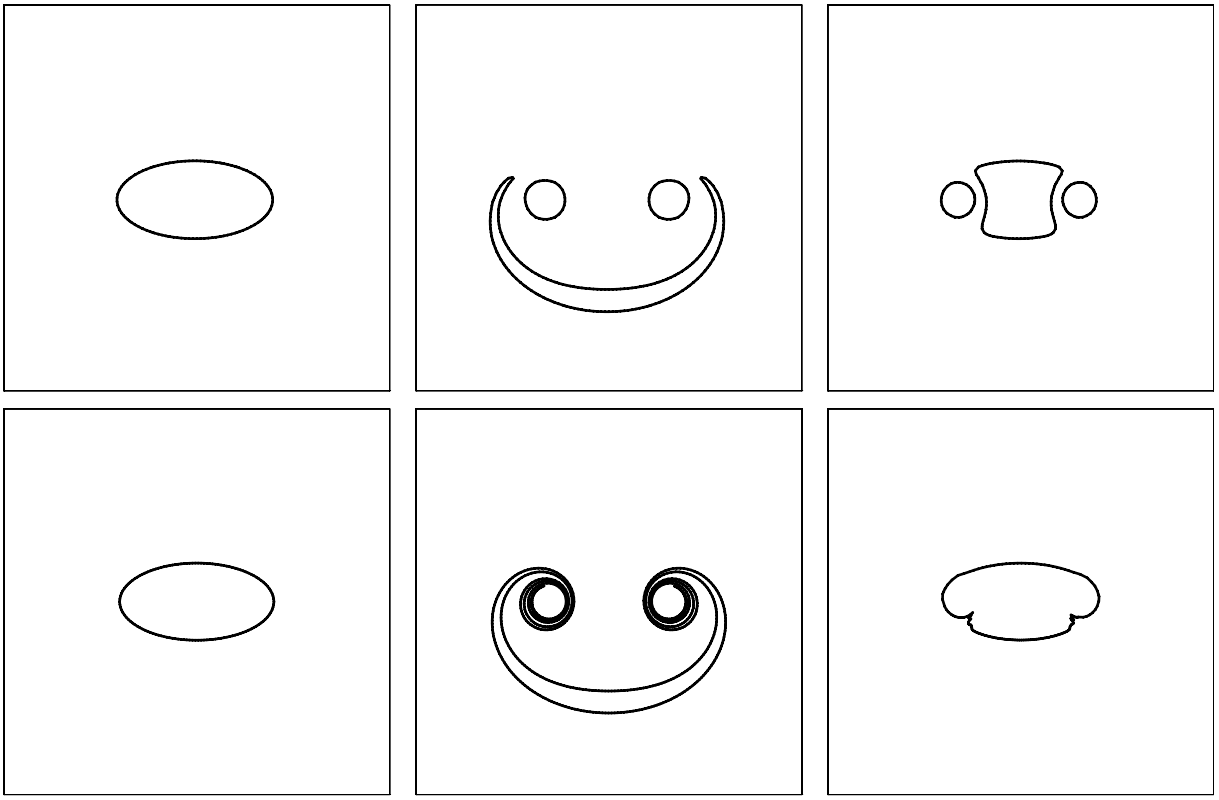

Fig. 7. Comparison between level set and front tracking on velocity reversal. The upper sequence shows the results of level set using fifth order WENO while the lower sequence shows the results of front tracking using the fourth order Runge–Kutta for the point propagation. The ellipse is placed in a dipole velocity field. The velocity is reversed at $t = 10$.

If we set the reversal time at $T = 10$, the swirling vortex tails become so thin that the level set method makes an incorrect topological bifurcation. On the reversal, this bifurcation cannot be recovered. The front tracking solution also looses some resolution, but it maintains the correct topology.

5.2. PLIC-VOF comparisons

To compare with the multiple interface methods studied in [17,18,12], we have simulated several test problems. These tests were done with exactly the same initial conditions and velocity fields as used by Rider and Kothe. These tests include

(1) the single vortex velocity field in two dimensions,
(2) the deformation velocity field in two dimensions,
(3) the deformation velocity field in three dimensions,
(4) the vortex shearing flow velocity field in three dimensions.

The computational domains and initial conditions for (1) and (2) are identical. It is a circle of radius 0.15 centered at (0.50, 0.75) within a 1×1 computational domain. For each test, we plotted the interface and calculated the L_1 norm to compare with the counterparts in [17,18]. We also performed convergence tests of (1) in comparison with that in [17].

5.2.1. Two dimensional single vortex simulation

The single vortex problem has the velocity field described by the stream function [19]

$$\Psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y). \tag{15}$$

To compare the results in [17,18], we let the simulation continue to $t = 3$ and then with a reversed velocity field to $t = 6$. A CFL number of 1.0 is used. Fig. 8 shows the interface evolution. The left plot shows the comparison of the initial circle and the interface after reversal, and the right plot shows the interface at $t = 3$. Topologically, the resolution of the vortex tail of the interface matches the best plot of that in [17,18].

We tested the convergence of the L_1 norm of the error. In L_1 norm tests, we multiplied Eq. (15) by a factor of $\cos(\pi t/T)$ so that the interface begins to reverse at $t = T/2$ and is fully reversed at $t = T$. The convergence tests are done for $T = 2$, 4, 6, respectively and are summarized by Table 3. Although the point propagation used the 4th order Runge–Kutta method, the order of convergence is only approximately equal to 2. The reason for the reduced order is due to the redistribution of the front points which is linear.

5.2.2. Two dimensional deformation test

The stream function for the two dimensional deformation velocity field is given by

$$\Psi = \frac{1}{4\pi}\sin\left(4\pi\left(x+\frac{1}{2}\right)\right)\cos\left(4\pi\left(y+\frac{1}{2}\right)\right)\cos\left(\frac{\pi t}{T}\right). \tag{16}$$

In this experiment, we have $T = 2$. For the purpose of comparison, the interface plot of the FronTier advection to $t = 1$ is on a 128^2 grid which is shown in Fig. 9. The plot is visually superior to all comparison solutions of [17,18], including the particle methods. We also performed a 64^2 reversal simulation to $t = 2$ to compare the L_1 norm with that in [17]. The L_1 norm the of front tracking simulation is $L_1 = 2.25 \times 10^{-3}$. This result is better than all methods in Table 4 of [17] except the Marker Particle (4) and Marker Particle (16) methods.

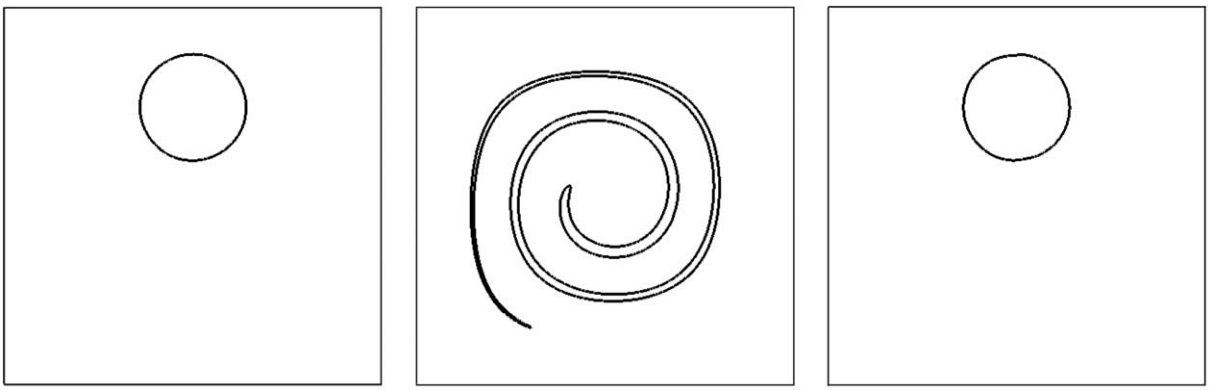

Fig. 8. Reversal test of a 2D interface in the single vortex velocity field. The computation is performed in a 128^2 computational mesh. In comparison with Figure 4 of [17], the resolution of the interface matches the best results by the Marker Particle methods.

Table 3
Convergence test of the 2D front tracking method under the single vortex velocity field with CFL = 1.0

Case	32^2	64^2	128^2
$T = 2$	1.03×10^{-3}	2.39×10^{-4}	9.54×10^{-5}
$T = 4$	2.12×10^{-3}	5.22×10^{-4}	9.35×10^{-5}
$T = 6$	2.79×10^{-3}	9.42×10^{-4}	1.19×10^{-4}

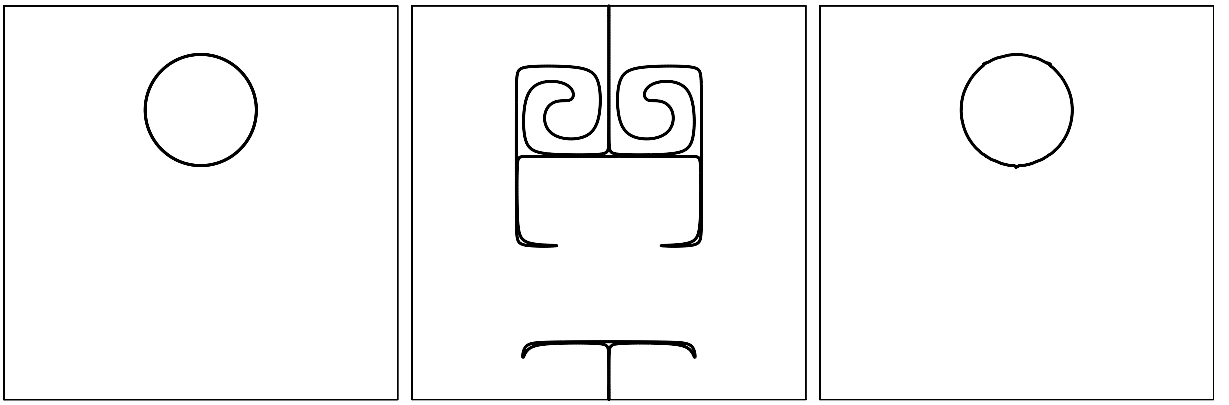

Fig. 9. Reversal test of a 2D interface in the 2D deformation velocity field. The figure is visually superior to all comparison solutions, including the particle methods in [17,18].

5.2.3. Three dimensional deformation field test

The velocity field in this experiment is described by the equations

$$u(x,y,z) = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \tag{17}$$

$$v(x,y,z) = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \tag{18}$$

$$w(x,y,z) = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T). \tag{19}$$

The interface evolves dynamically from an initial sphere of radius 0.15 centered at (0.35, 0.35, 0.35) to $t = 1.5$. The velocity field will then reverse its direction. At $t = 3.0$, the interface comes back to its initial state. The error comparison with the two PLIC methods in [12] is given in Table 4, and shows superior performance for LGB Front Tracking (see Fig. 10).

5.2.4. Three dimensional shear flow

The shear flow velocity field is described by

$$u = \sin(2\pi y)\sin^2(\pi x)\cos\left(\frac{\pi t}{T}\right), \tag{20}$$

$$v = -\sin(2\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right), \tag{21}$$

$$w = \left(1 - \frac{r}{R}\right)^2 \cos\left(\frac{\pi t}{T}\right), \tag{22}$$

where $r = \sqrt{(x-x_0)^2 - (y-y_0)^2}$, $R = 0.5$, $x_0 = y_0 = 0.5$.

The initial sphere is centered at (0.5,0.75,0.25) with radius 0.15. In Fig. 11 we plotted the interfaces of two runs using the LGB method at $t = T/2$ and $t = T$. In comparison with [12], The LGB method is visually superior. It maintains a singly connected topology throughout the simulation while the simulations in [12] using

Table 4
L_1 norms at $t = 3$ for the LGB method in the three dimensional deformation simulation compared to the two interface methods used in [12] with CFL = 0.5

Mesh	LGB	Order	CVTNA	Youngs
32^3	5.72×10^{-3}	3.72	7.41×10^{-3}	7.71×10^{-3}
64^3	4.33×10^{-4}	1.82	1.99×10^{-3}	2.78×10^{-3}
128^3	1.23×10^{-4}	N/A	3.09×10^{-4}	7.58×10^{-4}

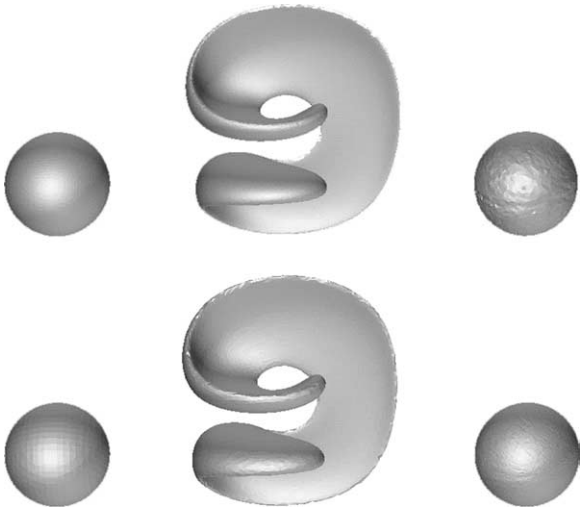

Fig. 10. Reversal test of a 3D interface in deformation velocity field with CFL = 0.5. The sequence above has the mesh of 64^3, and the sequence below has the mesh of 128^3. From left to right are $t = 0$, 1.5, 3, respectively.

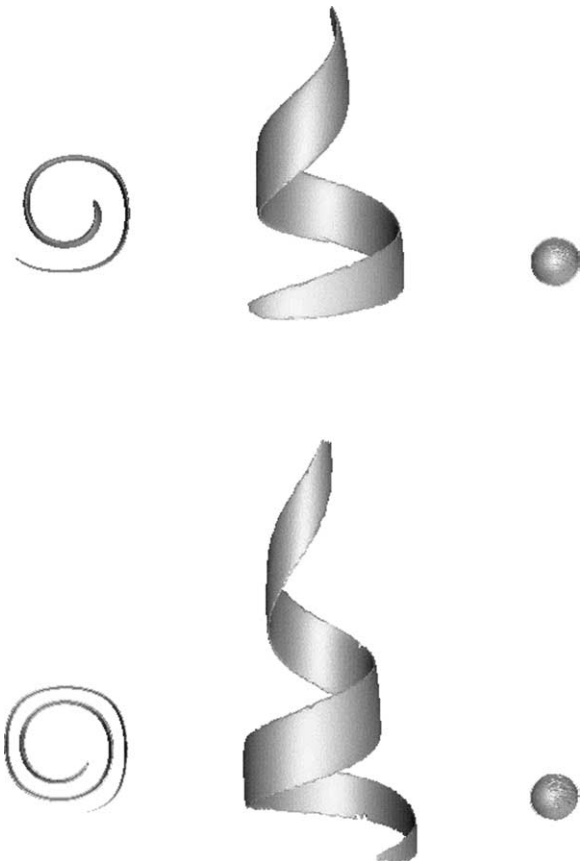

Fig. 11. Profiles at maximum deformation for the 3D shearing flow using the locally grid-based front tracking. The mesh is $64 \times 64 \times 128$ and CFL = 1.0. The first row is for $T = 6$ and the second row is for $T = 9$. The first column is the xy-plane view and the second column is the yz-plane view of the interface at $T/2$. The third column gives the interfaces after complete reversal $t = T$.

Table 5
L_1 norms for $T = 3$, $T = 6$, and $T = 9$ for the comparison of the LGB method with the two other methods in [12] in the three dimensional deformation simulation in the shear flow velocity field

Method	$T = 3$	$T = 6$	$T = 9$
LGB	6.28×10^{-4}	9.07×10^{-4}	1.19×10^{-3}
CVTNA	6.20×10^{-4}	3.64×10^{-3}	8.01×10^{-3}
Youngs	9.99×10^{-4}	4.38×10^{-3}	8.59×10^{-3}

The computational mesh is $64 \times 64 \times 128$ and CFL = 1.0 is used.

two PLIC methods showed bifurcation of the interface. Table 5 gives the comparison of the L_1 norms, and again shows the advantage of the LGB method.

6. Conclusion and availability

A major component of this work is to provide users who prefer the greater accuracy of Lagrangian type of front propagation with an assessment regarding its quality in comparison to the Eulerian level set method and VOF methods. The underlying geometrical calculations belong to basic analytical geometry and vector analysis. Optimization, such as the trade off between memory and CPU time have been performed through comparison and case-by-case testing and need not to be repeated. The code contains a detailed computation of geometry which is robust and efficient. The computation is wrapped by a comprehensible and simple user interface to allow many different applications to physical and scientific problems. Comparison to alternate interface methods shows the superior quality of front tracking.

The concept of the Lagrangian front tracking method is simple. Its complexity lies in the geometrical and topological handling of the interface dynamics. As the method matures, we begin to build a computational interface to shield its detailed operation and provide the users with a simple and user-friendly interface. The implementation of the high order Runge–Kutta method and the improved calculation of geometrical variables have made this method a highly accurate method in handling the propagation of interfaces with regions of high curvature. The method is robust, meaning that it has run for thousands of processor-days, solving complex interface dynamics, without the need for human intervention or case by case debugging.

Do high quality interface methods matter? This depends on the problem, but there are some problems for which the difference is large. Documented elsewhere, we have shown that interface smoothing and interpolation associated with lower quality interface methods can significantly reduce the growth rates for Rayleigh–Taylor mixing, as can the numerical diffusion associated with the absence of a tracked interface.

We have put the front tracking package in the public domain: `http://frontier.ams.sunysb.edu/download` where it is down loadable for use in scientific applications.

References

[1] G. Baker, D. Meiron, S. Orszag, Vortex simulations of the Rayleigh–Taylor instability, Phys. Fluids 23 (1980) 28–64.
[2] E. George, J. Glimm, X.L. Li, A. Marchese, Z.L. Xu, A comparison of experimental, theoretical, and numerical simulation Rayleigh–Taylor mixing rates, Proc. Natl. Acad. Sci. 99 (2002) 2587–2592.
[3] J. Glimm, J.W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, Y. Zeng, Three dimensional front tracking, SIAM J. Sci. Comp. 19 (1998) 703–727.
[4] J. Glimm, J.W. Grove, X.-L. Li, D.C. Tan, Robust computational algorithms for dynamic interface tracking in three dimensions, SIAM J. Sci. Comp. 21 (2000) 2240–2256.
[5] J. Glimm, J.W. Grove, W.B. Lindquist, O. McBryan, G. Tryggvason, The bifurcation of tracked scalar waves, SIAM J. Comput. 9 (1988) 61–79.
[6] J. Glimm, E. Isaacson, D. Marchesin, O. McBryan, Front tracking for hyperbolic systems, Adv. Appl. Math. 2 (1981) 91–119.
[7] J. Glimm, X.-L. Li, R. Menikoff, D.H. Sharp, Q. Zhang, A numerical study of bubble interactions in Rayleigh–Taylor instability for compressible fluids, Phys. Fluids A 2 (11) (1990) 2046–2054.
[8] J. Glimm, O. McBryan, A computational model for interfaces, Adv. Appl. Math. 6 (1985) 422–435.
[9] J.W. Grove, R. Holmes, D.H. Sharp, Y. Yang, Q. Zhang, Quantitative theory of Richtmyer–Meshkov instability, Phys. Rev. Lett. 71 (21) (1993) 3473–3476.

[10] R.L. Holmes, J.W. Grove, D.H. Sharp, Numerical investigation of Richtmyer–Meshkov instability using front tracking, J. Fluid Mech. 301 (1995) 51–64.
[11] T. Hou, J. Lowengrub, M. Shelley, Boundary integral methods for multicomponent fluids and multiphase materials, J. Comp. Phys. 169 (2001) 302–362.
[12] P. Liovic, M. Rudman, J-L. Liow, D. Lakehal, and D. Kothe. A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. Comput. Fluids, in press.
[13] M. Longuet-Higgins, E.D. Cokelet, The deformation of steep surface waves on water. I. A numerical method of computation, Proc. R. Soc. Lond. Ser. A 350 (1976) 1–26.
[14] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Comput. Graph. 21 (4) (1987) 163–169.
[15] N. Max, Weights for computing vertex normals from facet normals, J. Graph. Tools Arch. 4 (1999) 1–6.
[16] J.M. Rallison, A. Acrivos, A numerical study of the deformation and burst of a viscous drop in an extensional flow, J. Fluid Mech. 89 (1978) 191–200.
[17] W.J. Rider, D.B. Kothe, Stretching and tearing interface tracking methods, in: The 12th AIAA CFD Conference, San Diego, CA, June 20, 1995, AIAA-95-1717.
[18] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comp. Phys. 141 (1997) 112–152.
[19] P.K. Smolarkiewicz, The multi-dimensional crowley advection scheme, Mon. Weather Rev. 110 (1982) 1968–1983.
[20] Z.L. Xu, M. Kim, W. Oh, J. Glimm, R. Samulyak, X.L. Li, C. Tzanos, Atomization of a high speed jet. Phys Fluids, in press (SB Preprint Number: SUNYSB-AMS-05-08).
[21] A.M. McIvor, R.J. Valkenburg, A comparison of local surface geometry estimation methods, Machine Vision and Applications 10 (1) (1997) 17–26.