Interface Reconstruction in Multi-fluid, Multi-phase Flow Simulations

Rao V. Garimella[1], Vadim Dyadechko[2], Blair K. Swartz[3], and Mikhail J. Shaskov[4]

[1] Los Alamos National Laboratory, Los Alamos, NM 87545 rao@lanl.gov
[2] Los Alamos National Laboratory, Los Alamos, NM 87545 vdyadechko@lanl.gov
[3] Los Alamos National Laboratory, Los Alamos, NM 87545 bks@lanl.gov
[4] Los Alamos National Laboratory, Los Alamos, NM 87545 shashkov@lanl.gov

1 Introduction

Hydrodynamic simulations of flows involving multiple fluids and/or multiple phases are an important research area with many applications such as droplet deposition, sandwich molding processes, underwater explosions, mold-filling in casting, simulations of micro-jetting devices, etc.

A very important feature of multi-fluid, multi-phase flow simulations is the interface between materials and phases, and it is often crucial to follow such interfaces at each step of the simulation. Lagrangian simulations (where the mesh nodes move with the flow) automatically maintain interfaces, but cannot handle large deformations and topology changes of these interfaces. On the other hand, Eulerian simulations (where the mesh nodes are stationary) must incorporate special procedures to keep track of the interfaces in the flow.

In general, methods to follow interfaces in hydrodynamic simulations fall into two broad categories - interface tracking (cf. [1]) and interface reconstruction [2, 3]. In interface tracking methods, an initial interface is advected with the flow so that the positions of at least some points on the interface are known at each time step [1]. In general, the global topology of the interface is fixed at the initial step and not changed during the simulation. This is obviously a disadvantage for flows in which materials coalesce or fragment. On the other hand, interface reconstruction methods attempt to recreate the interface at each time step using information about the flow. Level set methods reconstruct the interface by locating the zero contour line of a distance function from mesh points to the interface [4, 5]. Level set methods can reconstruct complex interface topology rather easily but usually do not conserve material volumes very well. Volume tracking or Volume-of-fluid (VOF) methods reconstruct the interface based on volume fraction information in each mesh cell (See [3] for an overview). The volume fraction is "advected" using the flow solution at each time step. Volume tracking methods can be accurate but the local topology of the interface in a cell is quite simple (such as a single line segment) making it difficult to capture subcell details such as thin filaments.

In this paper, we describe an advanced volume tracking technique for reconstructing complex material interfaces in unstructured meshes. The method currently allows one arbitrarily aligned linear interface segment per cell and thereby belongs to a venerable class of interface approximations that goes back at least to Debar [6] and Youngs [7, 8]. More recently, this class of interface approximations have been referred to as PLIC or Piecewise Linear Interface Calculation by Rider and Kothe [3]. The method described here incorporates several new techniques designed to make the reconstruction method more accurate (generally 2nd order), rapid and robust. These include the careful selection and use of interface neighbor cells, and a topological consistency checking and repair algorithm for the interface that is designed to minimize fragmentation of the material regions being reconstructed.

2 Interface Reconstruction Procedure

2.1 Overview

The primary problem being addressed in this research is to reconstruct a piecewise linear interface given the volume fractions of different materials in the cells of an unstructured mesh. Cells filled with only one material are referred to as *pure* cells, and cells with some amount of two or more materials are referred to as *mixed* cells. Each mesh cell may contain at most one linear segment representing the interface. The reconstruction procedure must conserve the different cellular material volumes, make the interface as continuous as possible and avoid non-physical holes and fragments in the material regions.

The main steps of the interface reconstruction procedure are given below:

1. *Interface Estimation:* A rough estimate of the interface is constructed using the volume fraction data specified on cells.

2. *Interface Smoothing:* Interface segments are adjusted taking into account other interface segments in the neighborhood so that the resulting interface is as smooth as possible. For example, straight line interfaces are typically recovered by this interface smoothing step.

3. *Interface Topology Repair:* The interface segments are adjusted so that they satisfy topological consistency checks (described below). This operation tries to ensure that the reconstructed material regions are continuous and do not have holes or fragments where the physics does not dictate it.

4. *Constrained Interface Smoothing:* The alterations made to the interface in the repair step are smoothed, with the constraint that topological consistency of any vertex cannot be destroyed.

2.2 Interface Estimation

Material interfaces are general curves in 2D and general surfaces in 3D. The interface normal with respect to a material is defined as the outward normal to the interface between this and another material. If an estimate of the interface normal is available at some point in the cell, then the interface can be approximated as a line segment perpendicular to the interface normal, properly positioned to cut off the right amount of material in the cell.

Like most volume tracking methods, it is assumed in this research that there is a continuous and differentiable function whose values are given as volume fraction data at the centroid of each cell. This function is often referred to as a volume fraction function[5]. Then, the gradient of this function is approximated in a mixed cell using the discrete values of the function in the neighborhood of the cell using Green's theorem in the plane (See for example [10]). Some alternative methods such as the least squares construction of Barth [11] are presented in [3, 12]. The estimate for the interface normal is then taken as the negative of the gradient of the function.

Once an interface normal estimate is obtained in a cell, the interface approximation in that cell is simply a straight line perpendicular to the normal. However, this line segment must still be positioned correctly in the cell to subdivide it in the specified proportions. Fortunately, this is an easy problem involving quadratic splines lending itself nicely to a closed form solution [9].

This estimate of the interface normal is quite crude due to the imprecisely defined nature of the volume fraction function, because of the character of its rapid local variability (e.g., the function steepens as the mesh size decreases), and because the data around the mixed cell is often too sparse to give an accurate estimate (there are typically far more pure cells than mixed cells, particularly in the normal direction). Therefore, this calculation rarely recovers the exact interface, even if the interface is just a straight line. Although some researchers have advocated concentrating on this step to make it as accurate as possible, that is not the approach taken here. Instead the interface normal calculation from this step is only taken to be a rough estimate that is improved by a procedure referred to in this research as interface smoothing (described in the following section).

[5] Most articles on this subject have not clarified what exactly this function means; it is often assumed to be a discontinuous function specific to each material that is unity on one side of the interface and null on the other. In this case, it is also unclear what is meant by taking the gradient of such a function. The notion used here is that the volume fraction function is akin to a color function that indicates the average color in a small window that is moved around the mesh particularly in a normal direction to the interface. Such a function is still an imprecise function which varies with the shape and orientation of the window, and steepens with decreasing size of the window. See [9] for a detailed exposition of this topic.

2.3 Interface Smoothing

Interface smoothing is defined in this paper as the process of making the piecewise linear reconstruction of the interface, with one line segment per cell, as accurate and smooth as possible. This means that the procedure tries to make the endpoints and slopes of interface segments in two adjacent mixed cells match as far as possible. Such a procedure would recover the exact interface if the volume fraction data came from straight line material boundary.

The interface smoothing procedure in this paper derives from work by Mosso and Swartz [2] in which they described an iterative interface smoothing algorithm based on finding the straight line that subdivides two non-overlapping cells according to two arbitrary volume fractions. The uniqueness and existence of such a line (and a plane in 3D) are discussed in detail in an earlier paper by Swartz [9]. The Mosso-Swartz procedure recovers a straight line interface in a general unstructured mesh. Second order accuracy for curved interfaces is expected with their algorithm except in unusual geometric circumstances – for example, one should avoid just barely cutting off the corner of one or more cells. Rider and Kothe [3] also propose an algorithm for interface smoothing based on the kernel of the Swartz's work [9] and Swartz and Mosso's work [2].

More specifically, consider two cells in a two material problem for which the volume fractions of the materials and an initial guess for the interface segments are known in each cell. Then the (quadratically convergent) iterative algorithm to find the common line that subdivides the two cells in the right proportions is as follows:

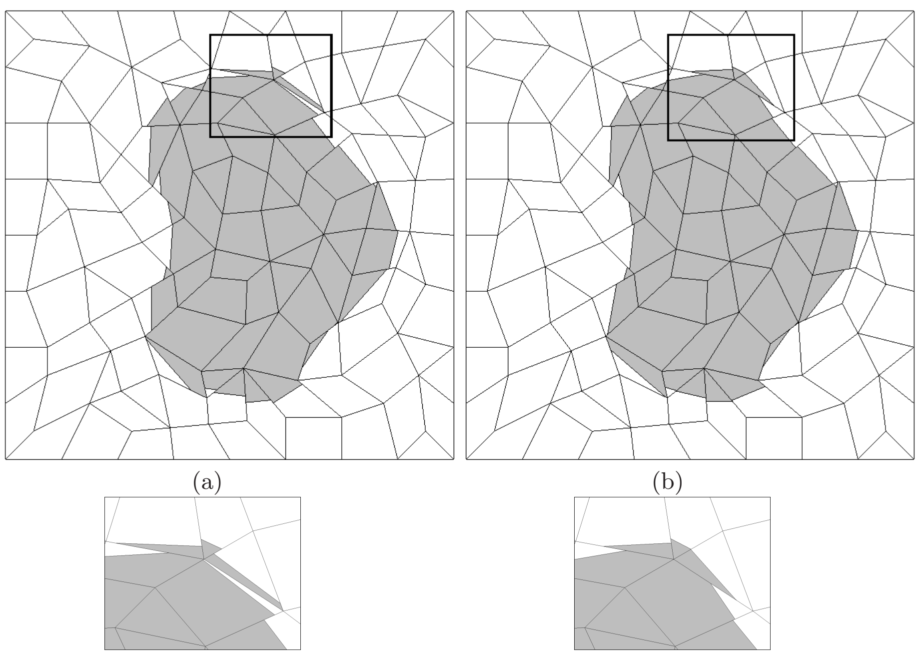
- Connect the centers of the interface segments in the two cells by a common line segment
- Use the perpendicular to this line as the estimate for the interface normal in each of the cells
- Move the interface segment in this normal direction in each cell to rematch the volume fraction associated with that cell.
- Repeat until convergence

The above procedure forms the basic building block for the interface smoothing algorithm of this paper. The interface smoothing procedure visits each mixed cell in the mesh and carries out the following steps:

- Obtain the mixed cell neighbors of the cell
- For each neighbor, find the normal of the common line that subdivides the cell and the neighbor according to the specified proportions
- Perform a weighted average of all such normals to obtain the smoothed interface normal for the cell
- Adjust the position of the interface so that the cell is subdivided according to specified proportions

If, for each cell, the weighting of the normals from its neighboring cells is the same, then no iteration is needed at the global level. However, in this implementation, the common line normal associated with a cell and a neighbor of the cell gets a higher weighting if the interface segments that currently exist in the two cells are probable continuations of each other. This makes the smoothing process order dependent in some situations and therefore, a few (2-3) global iterations may be necessary for convergence.

It is useful to point out a key difference between this interface reconstruction procedure and the Mosso-Swartz procedure [2]. In the current procedure, all pairwise normals of a given cell and its neighbors are each iterated to convergence. These are then averaged to give a final estimate for the interface normal in that cell. Global iterations are needed only to stabilize some changes as explained above. In the Mosso-Swartz algorithm, one iterative step is performed in a given mixed cell to improve each associated pairwise normal. These normals are then averaged to get an improved estimate for the interface normal in that cell. Then the algorithm moves on to process another mixed cell. Therefore, it must reiterate this procedure over the whole interface until it converge to a final solution. This difference may account for our numerical



(a) (b) (c) (d)

Fig. 1. Comparison of the reconstructed interface after (a) gradient estimation and (b) interface smoothing. Figures (c) and (d) show zoomed-in views of the interface in (a) and (b) respectively.

observation that the current interface reconstruction procedure is quadratically convergent while the Mosso-Swartz algorithm is linearly convergent.

2.4 Interface Topology Repair

Consider a hypothetical procedure that recovers the exact interface curves on a given mesh. In such a case, it is evident that every vertex in the mesh will lie inside some material region or exactly on a material interface (considering external boundaries to be interfaces between a material and unmodeled space). This condition is referred to here as *topological consistency* of the interface. Even if the interface reconstruction is approximate, it is reasonable to impose the topological consistency condition on it. Examples shown later demonstrate that imposing topological consistency on the interface results in material regions whose parts are more contiguous. This is invaluable in avoiding non-physical fragments and holes in the interface reconstruction process.

The rest of this section describes the process of making a reconstructed interface topologically consistent, referred to here as *interface topology repair*.

Given a mixed cell *and* an interface approximation in that cell, it is possible to classify each vertex of that cell as belonging to a particular material contained in that cell or as being on the interface between two or more materials. If the cell is a pure cell, then all of its vertices are classified with respect to that cell as belonging to the only material that the cell contains. From the vertex point of view, one can say that given a mesh and an approximate interface, every mesh vertex has a material classification with respect to each cell that uses this vertex, with the exception of cells where the vertex is on the interface.

Definition 1. *Topological Consistency of Vertex*

A mesh vertex is considered to be topologically consistent in the context of interface reconstruction, if the vertex is classified as being in the same material with respect to all cells connected to it.

Definition 2. *Topological Consistency of an Interface Approximation*

An approximately reconstructed interface is said to be topologically consistent if every vertex in the mesh is topologically consistent or is on the interface.

The procedure for interface topological repair visits each mesh vertex and checks its topological consistency. If the vertex is not consistent and is not classified as being on the interface, then the procedure tries to make the vertex topologically consistent.

The algorithm to make a vertex, V, topologically consistent is given below:

- Retrieve the cells, $\{C_i, i = 1, N_c\}$, connected to V.
- Catalog the materials, $m_j, j = 1, N_m$, present in the cells, $\{C_i, i = 1, N_c\}$.

- For each material, m_j, find the number of cells, N_j, in which V belongs to m_j.
- Arrange the materials in descending order of N_j.
- Pick the most populous material around the vertex — the material with the maximum N_j — as the primary material. Cells in which V does not belong to the primary material are called secondary cells.
- Adjust the interface in each mixed secondary cell so that the material classification of the vertex V in the cell changes to the primary material. This done by incrementally rotating the interface normal, readjusting its position to capture the right volume fraction and checking the material classification of V in that cell. Both positive and negative increments from 0 to 180 degrees are tried simultaneously to speed up the process.
- If all the secondary cells could not be repaired in this fashion, then try the procedure with the next most populous material as the primary material.

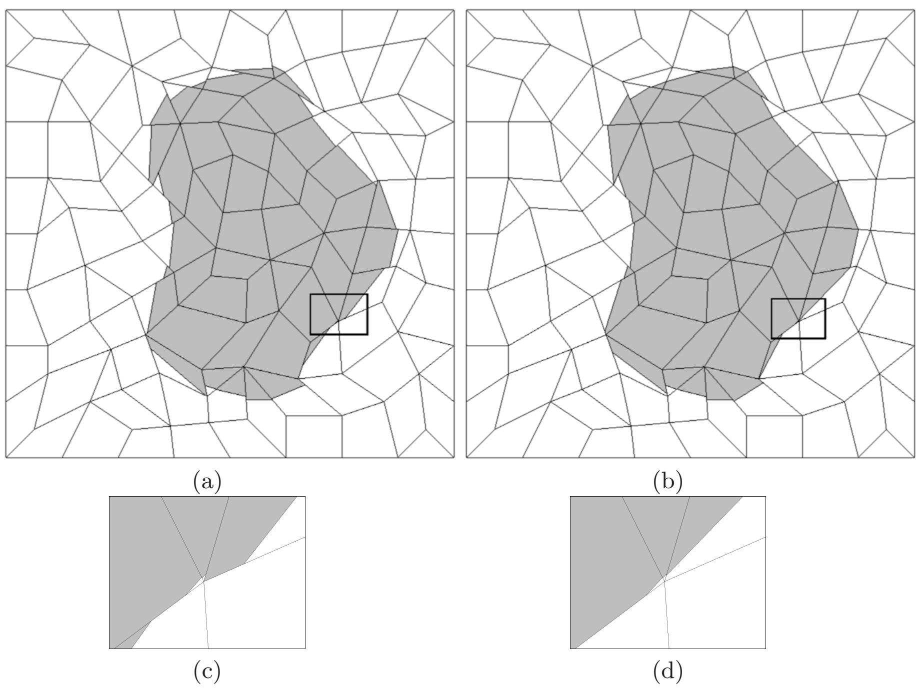


Fig. 2. Comparison of the reconstructed interface after (a) interface smoothing and (b) interface topology repair. Figures (c) and (d) show zoomed-in views of the interface in (a) and (b) respectively.

The interface topological repair process modifies interface normals only as much as necessary to fix topological inconsistencies. Therefore, there may

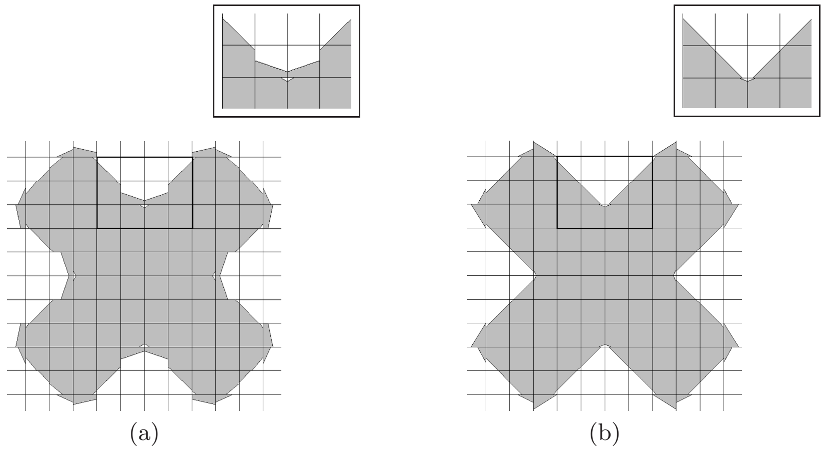


Fig. 3. Example showing elimination of gaps via interface topology repair.

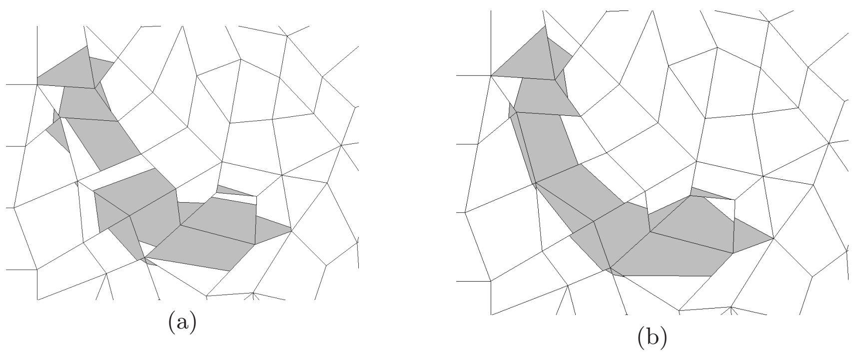


Fig. 4. Example showing reduced fragmentation via interface topology repair.

be some modified segments of the interface whose normals do not vary very smoothly with respect to their neighbors. A constrained interface smoothing procedure is reapplied to the interface to fix this problem. Unlike the first smoothing step, the constrained smoothing step takes care not to destroy the topological consistency of any vertex.

3 Tests

In this section, some test results are presented to demonstrate the accuracy and robustness of the interface reconstruction procedure.

First, results of interface reconstruction on some static examples are presented. In these examples, known analytical shapes are used to create volume fraction data in polygonal meshes and the interface is reconstructed.

Next, results of interface reconstruction in a dynamic setting similar to a fluid flow problem are presented. In these dynamic examples, a known shape is superimposed on a mesh to get the initial volume fraction data which is then advected using an analytical velocity field. The interface is reconstructed at each time step, and the quality of the interface is evaluated at the final step.

3.1 Static Tests

The first example, shown in Figure 5a, is a simple test to ensure that interface reconstruction procedure recovers a straight line exactly on a general unstructured mesh. Figure 5b shows the reconstruction of a circular interface on an unstructured triangular/quadrilateral mesh.

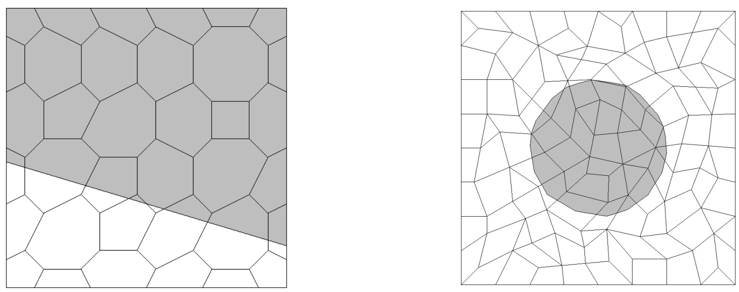


Fig. 5. Static Interface Reconstruction Tests: (a) Reconstruction of a straight line on an unstructured polygonal mesh (b) Reconstruction of a circular interface on an unstructured triangular/quadrilateral mesh

3.2 Advection

Described below is an advection scheme for incompressible two-phase fluid flow. In this case, it is sufficient to follow the evolution of the volume fractions of a single material further referred to as a *reference* material.

The time step Δt is determined by the minimal mesh spacing h_{min} and the Courant constant CFL: $\Delta t = CFL \cdot h_{min},\ CFL \leqslant 1$.

Given an analytical *solenoidal* velocity field and the interface between the materials at the moment $t_{k-1} = \Delta t\ (k-1),\ k = 0, 1, 2, \ldots$ one can calculate the reference materials at the moment $t_k = \Delta t \cdot k$:

for each cell $C_i,\ i = 1, N$ **do**
track the cell C_i *back in time* to $t = t_{k-1}$;
find the volume $\tilde{v}_{i,k-1}$ of the reference material enclosed in the *Lagrangian prototype* $C_{i,k-1}$ of the cell;
put the volume of the reference material $v_{i,k}$ for the cell C_i equal to the $\tilde{v}_{i,k-1}$.

end do

The algorithm is implemented as follows. The vertices of the polygonal cell C_i are tracked back along the streamlines of the given velocity field using a 4-th order Runge-Kutta integration scheme and then connected by straight segments. This results in a polygon that is referred to here as a *Lagrangian prototype* $C_{i,k-1}$ of the cell C_i (see Fig. 6).

Using a polygon intersection routine[6], the intersection of the $C_{i,k-1}$ with each covered cell of the grid at $t = t_{k-1}$ is found. Since $CFL \leqslant 1$, then it is sufficient to check intersections of the prototype only with the cell C_i and its neighbours. Moreover, if C_i along with all its neighbours is empty, then it is guaranteed to stay empty at the next discrete moment of time; there is no need to proceed with polygon intersections in this case.

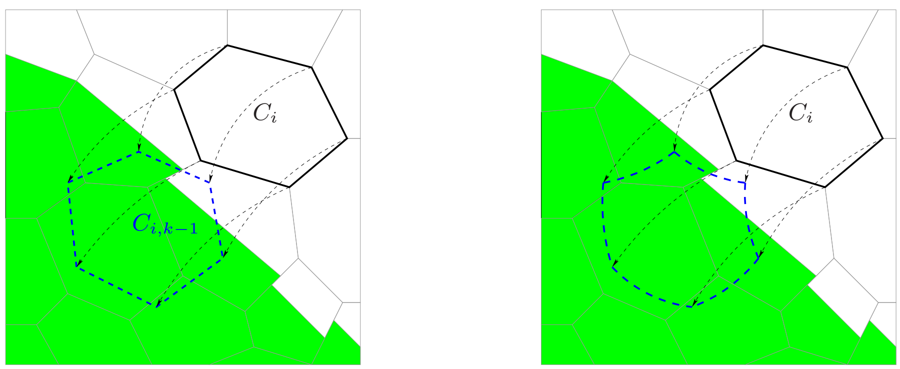


Fig. 6. Pseudo-Lagrangian (left) and true Lagrangian prototypes of the cell C_i.

This algorithm works quite well for the solenoidal velocity fields that correspond to the *rigid body motion*. Such a velocity field preserves straight lines and the scheme employed for tracking cells back in time results in a *true Lagrangian prototype* of the cell. In a uniform velocity field, the integration scheme used is exact, and the volume of the prototype $|C_{i,k-1}|$ matches the volume of the cell $|C_i|$. If the velocity field contains a rotational component, then the local area defect $|C_{i,k-1}| - |C_i|$ introduced by the Runge-Kutta scheme is $O(h^6)$, which vanishes rapidly as h (the average mesh spacing) goes down.

However, in general, straight lines in a solenoidal velocity field are not preserved and the Lagrangian prototype of the polygonal cell is not guaranteed to be a polygon. Therefore, the polygon $C_{i,k}$ introduced above is not a *true Lagrangian prototype* of the C_i but a *pseudo* one. The local area defect due to the fact that we ignore the curvature of the prototype edges is $O(h^3 \Delta t)$ and may eventually cause the advection algorithm to encounter problems. In par-

[6] Using the COnservative REmapping (CORE) library by M. Staley, T-7, LANL

ticular, it is possible that the volume, $\tilde{v}_{i,k-1}$, of the reference material enclosed in the prototype $C_{i,k-1}$ exceeds the cell capacity $|C_i|$. Another (though not as critical) situation occurs when the prototype, filled with the reference material, happens to have an area $|C_{i,k-1}| < |C_i|$. In this case, the cell becomes mixed, even though its prototype was pure (contained only one material).

In order to fix these deficiencies, a post-advection *repair* procedure is used. For every cell $C_i,\ i = 1, N$ we specify the lower and the upper bounds $\underline{v}_{i,k},\ \overline{v}_{i,k}$ of the reference fraction volume $v_{i,k}$ allowed:

$$\begin{aligned}
\underline{v}_{i,k} = \quad & \overline{v}_{i,k} = 0 \quad \text{if the prototype is empty,} \\
\underline{v}_{i,k} = \quad & \overline{v}_{i,k} = |C_i| \text{ if the prototype is full,} \\
\underline{v}_{i,k} = 0, & \overline{v}_{i,k} = |C_i| \text{ otherwise,}
\end{aligned}$$

and then force each volume $v_{i,k},\ i = 1, N$ to fit in:

```
for each cell C_i, i = 1,N do
    if the cell C_i is overfilled (v̄_{i,k} < v_{i,k}) then
        try to redistribute the excess between the non-overfilled neighbors
        while there are still some leftovers do
            redistribute them among the next layer of the surrounding cells
        end do
        else if the cell C_i is underfilled (v_{i,k} < v̲_{i,k}) then
        try to borrow the deficit from the non-underfilled neighbors
        while there is still some deficit of material do
            borrow it from the next layer of the surrounding cells
        end do
    end if
end do
```

Due to the local nature of the volume defect, the redistribution usually involves only direct neighbours of the cell making the complexity of the repair step $O(N)$.

3.3 Dynamic Tests

Rider and Kother argued in their paper [3] that simple translation and rotation tests are inadequate to predict the performance of interface reconstruction algorithms in real flow simulations which often involve stretching and shearing of materials. They proposed a test with an analytical velocity field (based on [13] and [14]) which sets up a single vortex that stretches a circular region into a spiral with several full rotations and reverses back to a circle. The circle is of radius 0.15, placed at (0.5,0.75) in a unit square whose lower left corner is at (0.0,0.0). The two indicators used to evaluate the performance of the interface reconstruction algorithm are how well the structure of the spiral is maintained at the point of maximum stretching and how close to the original shape the material is when the flow is fully reversed.

The forward motion of the vortex is defined by the stream function

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

,

from which we get the divergence free velocity field

$$u_x = -\frac{\partial \Psi}{\partial y} = -sin^2(\pi x) sin(2\pi y) \tag{2}$$

$$u_y = \ \frac{\partial \Psi}{\partial x} = \ sin(2\pi x) sin^2(\pi y) \tag{3}$$

This velocity is reversed in time by multiplying the components with the term $cos(\pi t/T)$, where $T/2$ is the point of maximum stretching [3, 14].

Figure 7 shows the results of the vortex test for three successively finer meshes at the point of maximum stretch and after full reversal of the flow. As earlier researchers have observed, volume tracking methods tend to exhibit breaking up of the tail of the vortex if the mesh is not sufficiently refined. This is evident in the 32x32 rectangular mesh examples but largely disappears as the mesh is refined to 128x128 cells. The primary reason for this behaviour is that the method assumes that one line segment per cell is sufficient to represent the interface whereas the procedure generally needs at least two disjoint line segments per cell to represent the vortex tail as it thins out in a coarse mesh. Failure to capture this structure leads to ever accumulating errors in the interface reconstruction (but not volume conservation), and to eventual disintegration of the vortex structure. It must be noted here that although there is break-up of the vortex tail, the results presented here show less fragmentation than those presented in some earlier works ([3],[15]).

4 Conclusions

A volume tracking interface reconstruction for multi-material, multi-phase hydrodynamic simulations was presented in this paper. The procedure incorporates several new techniques to make the reconstructed interface more accurate, smooth and consistent. Results of simulations indicate that the procedure performs better than previous technqiques on difficult problems. Extensions to handle more than two materials per cells are planned.

5 Acknowledgements

This research was performed at Los Alamos National Laboratory operated by the University of California for the US Department of Energy under contract W-7405-ENG-36. The authors also acknowledge the support of the ASC program at Los Alamos National Laboratory. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

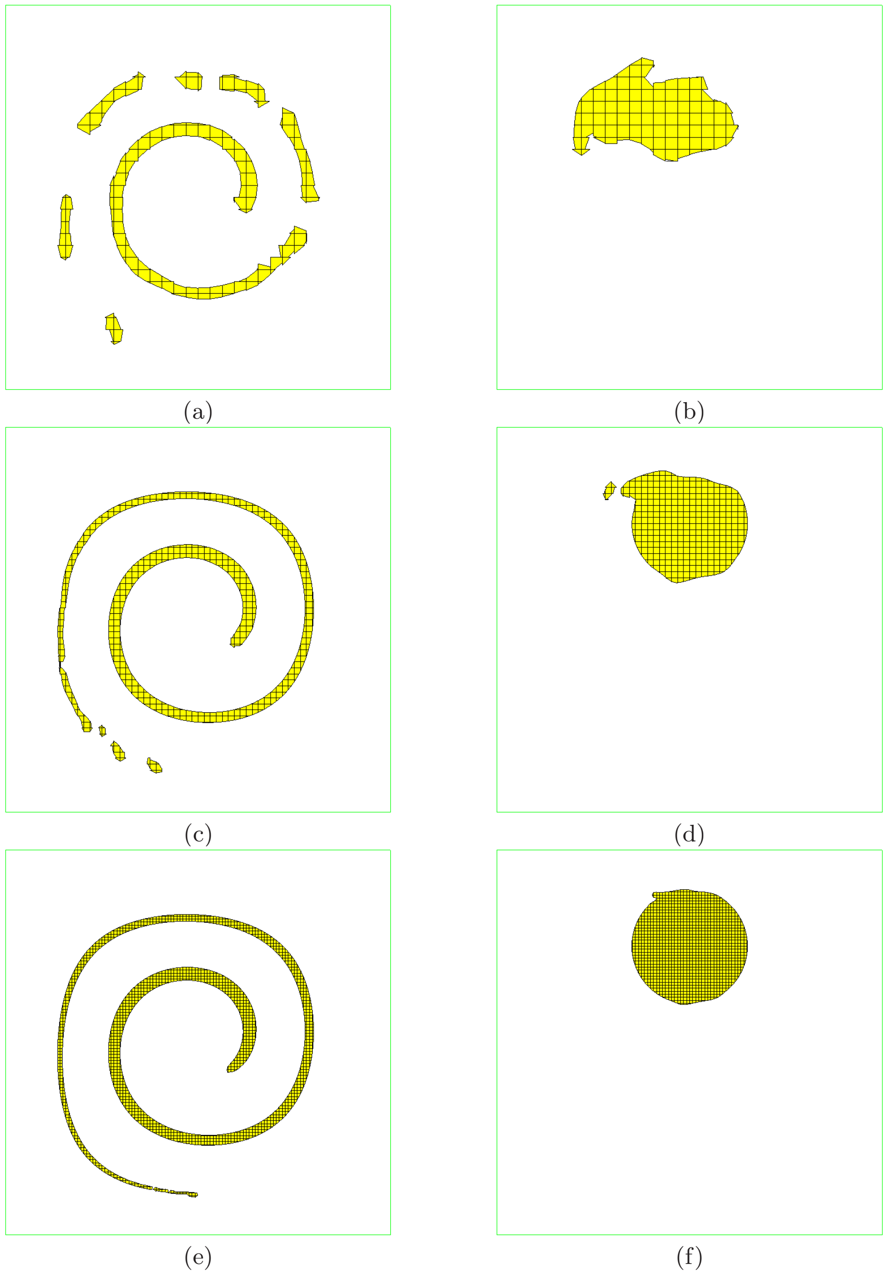


Fig. 7. Results of a time reversed single vortex simulation for different rectangular grids; (a) 32x32 grid at t=4 (maximal stretch) (b) 32x32 grid at t=8 (fully reversed flow) (c) 64x64 grid, t=4 (d) 64x64 grid, t=8 (e) 128x128 grid, t=4 (f) 128x128 grid, t=8

References

1. E. Aulisa, S. Manservisi, and R. Scardovelli. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *Journal of Computational Physics*, 188:611–639, 2003.
2. S. J. Mosso, B. K. Swartz, D. B. Kothe, and R. C. Ferrell. A parallel, volume-tracking algorithm for unstructured meshes. In P. Schiano, A. Ecer, J. Periaux, and N. Satofuka, editors, *Parallel Computational Fluid Dynamics: Algorithms and Results Using Advanced Computers*, pages 368–375. Elsevier Science B.V., 1997.
3. W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141:112–152, 1998.
4. S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *Journal of Computational Physics*, 169(2):463–502, May 2001.
5. M. Sussman, E. Fatemi, P. Smereka, and S. Osher. Improved level set method for incompressible two-phase flows. *Computers & Fluids*, 27(5-6):663–680, Jun-Jul 1998.
6. R. DeBar. Fundamentals of the KRAKEN code. Technical Report UCID-17366, Lawrence Livermore National Laboratory, 1974.
7. D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. In K. W. Morton and M. J. Baines, editors, *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, 1982.
8. D. L. Youngs. An interface tracking method for a 3d eulerian hydrodynamics code. Technical Report 44/92/35, AWRE, 1984.
9. B. Swartz. The second-order sharpening of blurred smooth borders. *Mathematics of Computation*, 52(186):675–714, Apr 1989.
10. M. J. Shashkov. *Conservative Finite-Difference Methods on General Grids*. CRC Press, Boca Raton, FL, 1996.
11. T. J. Barth and D. C. Jespersen. The design and application of upwind schemes on unstructured meshes. In *Proceedings of the 27th Aeroscience Meeting*, Reno, NV, Jan 1989. AIAA.
12. D. B. Kothe, W. J. Rider, S. J. Mosso, J. S. Brock, and J. I. Hochstein. Volume tracking of interfaces having surface tension in two and three dimensions. In *Proceedings of the 34th Science Meeting*, Reno, NV, Jan 1996. AIAA.
13. J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method of the incompressible navier-stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.
14. R. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal of Numerical Analysis*, 33:627–665, 1996.
15. D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183:83–116, 2002.