

LA-UR-

*Approved for public release;
distribution is unlimited.*

Title:

Author(s):

Submitted to:

Los Alamos

NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. 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.

Unstructured grids in 3D and 4D for a time-dependent interface in front tracking with improved accuracy

J. Glimm*
J. W. Grove**
X. L. Li*
Yingjie Li*
Zhiliang Xu*

*Department of Applied Mathematics and Statistics, SUNY at Stony Brook
Stony Brook, NY 11794 USA

linli@ams.sunysb.edu

**Los Alamos National Laboratory

Los Alamos, NM 87545 USA

jgrove@lanl.gov

Abstract

Front tracking traces the dynamic evolution of an interface separating different materials or fluid components. In this paper, we describe three types of the grid generation methods used in the front tracking method. One is the unstructured surface grid. The second is a structured grid-based reconstruction method. The third is a time-space grid, also grid based, for a conservative tracking algorithm with improved accuracy.

Introduction

The computation of a dynamically evolving material interface demands an effective algorithm to detect and resolve the changing topology of a moving front. The front tracking method explicitly traces the lower dimensional manifold embedded in space. As the front vertices propagate in the triangulated interface, the surface needs to be re-adjusted to maintain uniformity of the mesh. When bifurcation occurs, a reconstruction is needed to suite the new topology after the bifurcation. Two methods [1, 2] have been used for these purposes. The first, independent of the structured spatial grid, is called grid-free tracking. This method combines redistribution of the surface grid meshing and the crossing detection based untangling (bifurcation). This method is able to provide quality simplices to ensure accurate and uniform propagation of the interface but fails when multiple surface sections meet and the bifurcation becomes overly complicated. A second method, grid-based tracking, reconstructs the front using the rectangular

structured grid as a base frame. It was previously used in computer graphics [5]. This method assumes that the interface intersects each edge of a rectangular mesh block only once. This assumption greatly simplifies the topological variation of the interface within the block. However, due to the arbitrariness of the interface-grid line intersection, the quality of the triangulation is not guaranteed. Our preferred approach will combine the two methods: namely use of the grid-free interface during regular expansion and contraction but use of the grid-based method for topological bifurcations.

Another grid generation requirement arises from a new algorithm for the conservative front tracking method [3]. Tracking methods previously employed the ghost-cell method for the construction of finite difference stencils crossing different fluid components [4]. This method has been successful in the computation of physical problems requiring the maintenance of a sharp discontinuity, but it is not conservative because the fluxes calculated through ghost-cells cannot cancel at two sides of the interface. Our new conservative tracking method is based on the matching of the dynamic flux at the moving cell boundary. To do this, we need to create computational cells in time-space coordinates. Such cells are located between two time levels t^n and t^{n+1} and are bounded in part by the moving front of the interface. They are generated by the grid based algorithm applied in space-time.

The Dynamic Surface Grid for the 3D Front

An interface is a representation of wave fronts in a flow. Geometrically, these fronts consist of a set of surfaces, curves along the boundaries of the surfaces, and the curve boundaries called nodes. A valid interface is one where each surface and curve is connected, surfaces only intersect along curves, and curves only intersect at nodes. For the three dimensional flows, we assume the interface is embedded in a compact subset of the R^3 computational domain. The surfaces divide this domain into a set of connected components. Topological consistency requires that a component is uniquely assigned at each given space point. The discretization of an interface is given by representing each surface as the union of a set of planar triangular elements, the vertices of which are called points.

Within each connected component we assume a fixed fluid equation of state. Physical quantities such as density, velocity, or energy, as well as material properties (such as equation of state parameters) and even the dynamical equations may have a jump discontinuity across a surface.

Points on an interface are propagated at each time step in a spatially operator split fashion, using a local coordinate system aligned with the interface. The normal propagation step is based on the solution of Riemann prob-

lems using data from either side of the surface together with the method of characteristics for incorporating the influence of incoming and outgoing waves at the interface.

As the wave fronts evolve, some of their sections will converge while others diverge. This causes stretching and compression of the surfaces on the interface and will eventually lead to severe distortion in the triangles that make up the interface elements. This is a common problem with all Lagrangian hydrodynamic methods, with which front tracking shares many features. We use a time periodic retriangulation of the interface to maintain uniformity of the triangles that form the surfaces.

An additional difficulty occurs due to wave interactions such as droplet pinch off or droplet merger. Since the propagation algorithm for the front points does not use global topological information, surface intersections are not detected during front propagation. The interface must be explicitly checked for intersections after each time step, and when detected, these intersections must be resolved.

The topological bifurcation algorithm in the grid-free method has three main steps, intersection detection, retriangulation of intersecting triangles, and interface surgery where unphysical interface sections are removed and the surfaces are reconnected into valid non-intersecting sections.

All pairs of distinct triangles not sharing a common point are tested for intersections. The basic algorithm is elementary. First the line of intersection of the two planes determined by the triangles is computed. The two triangles are checked to determine whether they both cross this line and if so we compute the overlap of the line segments formed by triangle-line intersections. If such an overlap exists then we record the two intersecting triangles and their segment of intersection.

For each intersecting triangle on the two intersecting surfaces, the intersection bonds divide it into two polygonal parts each bounded by the bonds of the crossing curve and the original triangle sides. Briefly, the algorithm uses a divide and conquer scheme. The domain is first embedded in a finite rectangle, which is then subdivided into parallel strips with a single vertex per strip. The basic operations are to triangulate a strip and to adjoin adjacent strips.

The first two steps produce an untangled interface that satisfies all the requirements for a valid interface except for the consistency of its embedding into the computational domain. For surfaces meeting along the intersection curves, the component numbers of the common side of the surfaces meeting at this curve are inconsistent, so that it is impossible to assign components

to the regions of the computational domain defined by this interface. Once these surface segments have been identified and removed, the remaining surfaces will define a topologically consistent interface. Local information near the intersection is not sufficient to resolve the inconsistency, and global knowledge of the interface topology and geometry must be used to identify the unphysical surface segments. We use a coboundary method to supply the missing information. If a surface bordering an intersection curve has other bounding curves, not involved in the intersection and interface reconstruction process, then this surface must be physical and is retained. If no such bounding curves exist (as for example in the collision of two closed surfaces), then we appeal to geometric information and remove surface segments from the intersection that have small areas compared to the other surface segments.

The deficiency of the grid-free algorithm is that the intersections of surfaces can be arbitrarily complicated and therefore make the untangle process less robust.

Another method to resolve changing interface topology is to reconstruct the interface using micro-topology within each rectangular grid block cell on a user specified lattice. In practice the lattice used for this reconstruction is the dual lattice of the computational grid, i.e. the lattice defined by the cell centers of the computational grid. This scheme is divided into three steps: (1) Compute the crossings of the interface and the grid block edges, (2) Determine component values at the grid block corners and eliminate inconsistent crossings, and (3) Reconstruct a new interface using the remaining consistent grid block edge and interface crossings.

The intersection algorithm is again elementary. Compute the intersection of the plane of the triangle with the line of the cell edge, and then determine whether this point lies both inside the edge and within the convex hull of the three triangle vertices. To reduce the number of intersection tests, we use a hashing type scheme (also used for the grid free case) to create for each grid block in the reconstruction lattice the list of triangles that intersect that block.

The intersections of the interface with the cell edges divides each edge into a set of subintervals. In regions where the interface is not tangled, the component labels on the interface provide a well defined component label for each of these subintervals. If tangles are present some subintervals will have different component labels at their opposite endpoints. We process each cell edge to eliminate crossings that produce inconsistencies in the assignment of component values to the subintervals.

Once the unphysical crossings have been removed, we use the remaining edge crossings to reconstruct the interface.

We use an algorithm similar to the one by Lorensen and Cline [5] in computer graphics to reconstruct the interface in a single grid block and the assemble of the block surface elements into global surfaces. Note that the reconstruction algorithm automatically ensures consistency of the surface elements generated from adjacent grid blocks since interface crossings along a specified lattice edge are the same for all cells that contain that edge, and the generated triangles are required to have sides that link adjacent crossings along a cell face.

Figure 1 and Figure 2 show the two interface grid structures in a simulation of the Rayleigh-Taylor instability. Both the grid free and grid based methods described above have advantages and deficiencies. The grid free method produces a high quality distribution of triangle sizes and shapes (see Fig. 1), and accurately controls numerical diffusion. It suffers from code complexity and is subject to failure when the interface is complex. The grid based method is over diffusive, which is manifested as an over smoothing of the interface. It also tends to produce poorly conditioned surface triangles due to the constraint of reconstructing the surface elements within a single grid block, see Figure 2. On the other hand, this method is highly robust and always reconstructs a topologically valid interface. For optimal results we use a hybrid strategy, alternating (at some frequency) the two methods.

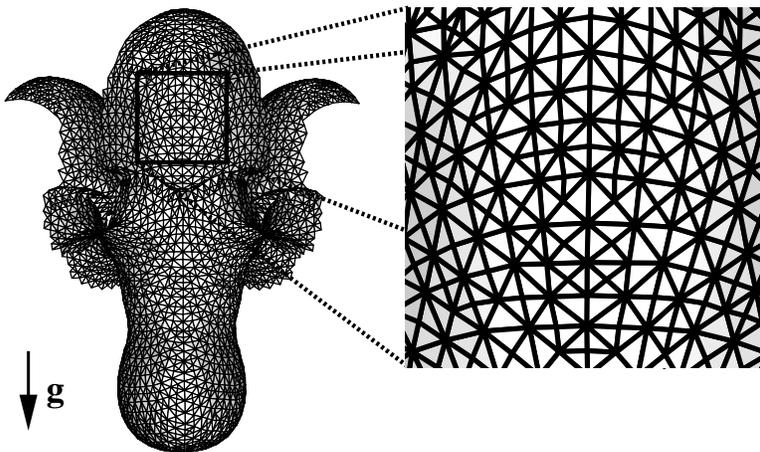


Figure 1. An evolved fluid interface using the grid-free redistribution algorithm. A detail of the surface triangulation is shown on the right.

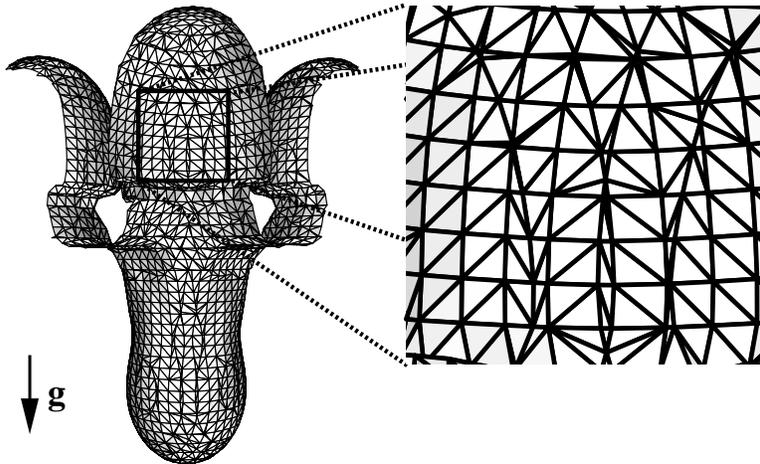


Figure 2. An evolved fluid interface using the grid-based redistribution algorithm. A detail of the surface triangulation is shown on the right.

The Two Dimensional Space-Time Grid

Consider the two space dimensional system of conservation laws

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0, \quad (1)$$

defined in a spatial domain Ω . Assume that Ω can be partitioned by a uniform square grid whose boundary lies along grid lines. The side of a grid cell is of length Δx .

Our algorithm is organized into three main steps. The first is to identify the space-time propagated interface. To do this, we identify the crossings of the approximate space time interface with the edges of the space time hexahedra. The second is to construct a finite volume decomposition which respects the space-time interface. We split the space time hexahedra whose interior is cut by the space time interface into parts each of which belongs to only one side of the space time interface. We merge those with small top area to form a polyhedron with top area bigger than $0.5\Delta x^2$. Finally, as a third step, we derive a finite volume discretization associated with the space time interface and the conforming space time polyhedra. This step is similar to the related step in [6], where a conservative higher order algorithm is constructed for a stationary boundary. Additional references can be found in [6], including ones to the cell merging problem.

In the present study, we require that at each time level, the 2D discretized spatial interface is a disjoint union of non intersecting curves. Each curve is piecewise linear and connected, and composed of bonds. Each bond is a pair of interface points or points, and (conceptually) the straight line segment joining them. Each curve is assigned an orientation which remains unchanged during the propagation of the interface.

Propagation of the points of a grid based interface will yield a general interface, not grid based, as there is no reason for a propagated point to lie on a grid cell edge, just because it starts as one. According to the grid based construction, we consider this propagated interface as a collection of polygonal curves in \mathbb{R}^2 . Crossing points of the curve with grid cell edges are inserted as new points. The propagated old points will be deleted (named images of propagation in this sense). The curve is then reconstructed, as straight line segments joining these new points. In this process, the curve is displaced by an amount $\mathcal{O}(\Delta x^2)$, assuming that the curve is smooth, so that all angles between neighboring bonds are $\mathcal{O}(\Delta x)$. Finally the space time interface is reconstructed to be grid based relative to the space time grid cells. We assume that there is no topological change of the interface during the time interval of computation. (Exceptions are treated separately using the spatial ghost cell algorithm.) See Fig. 3.

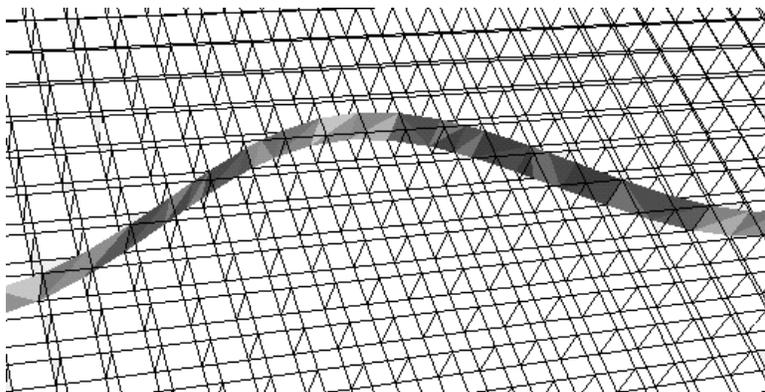


Figure 3. Space time interface for $D = 2$ space dimensions

We also assume that the CFL number is less than $1/2$. so that each point of the interface can be propagated a distance less than $\Delta x/2$. This condition may not be sufficient if the interface intersects the boundary at a small angle. We adjust the CFL number so that the intersection point between them moves a distance less than Δx along the boundary during the time step in order to reach the same property. We construct the approximate

space time interface first by joining each point P at the time level t_n with its image at the time level t_{n+1} (before constructing a grid based interface at the time level t_{n+1}). Then we connect one diagonal of each quadrilateral to fully triangularize the approximate space time interface. Finally, the approximate space time interface is linearly reinterpolated locally according to its crossings with the edges of a space time hexahedron. There are only 16 different topological structures of the linearized space time interface within each space time hexahedron modulo rotation and reflexion. This also creates a grid based interface at the time level t_{n+1} . It is easy to see that this is a locally second order reconstruction process. See Fig. 4.

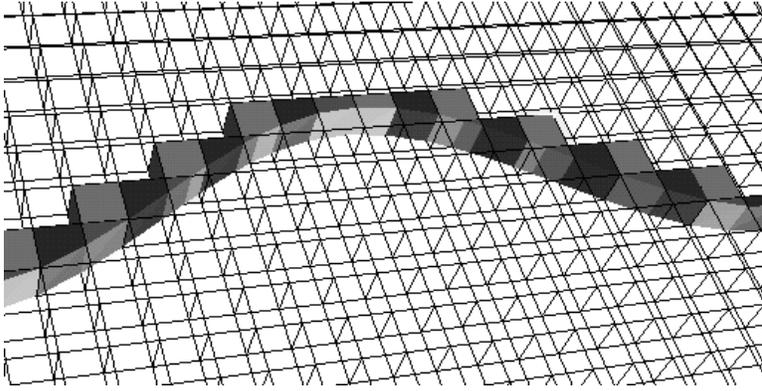


Figure 4. Partial hexahedra on one side of space time interface

Construction of the Space-Time Hexahedra

We connect the nodes of a cell D_i^n at time $t = t_n$ to the nodes of its corresponding cell D_i^{n+1} at time $t = t_{n+1}$ to form a space-time hexahedron. We call D_i^{n+1} the top of the hexahedron and D_i^n the bottom. We call a hexahedron mixed if the interface passes through its interior; otherwise it is pure. The mixed hexahedra are divided into pure partial hexahedra, and if necessary, these are combined with neighbors to form the finite volume space-time grid suitable for construction of a conservative difference algorithm.

Two space-time polyhedra are adjacent if they share a non-trivial surface which is not on the space time interface. Two space-time polyhedra are neighboring if they share a non-trivial vertical line segment which is part of the line connecting two corresponding grid nodes at the time levels t_n and t_{n+1} (denoted as a vertical grid line), and not on the space-time interface. It is easy to see that two adjacent or neighboring polyhedra must be on

the same side of the space time interface. It can be proven that if a space-time polyhedron is constructed by merging any number of adjacent partial hexahedra with no top, then the polyhedron will be adjacent to a pure or partial hexahedron. This ensures the eventual success of the merging algorithm.

The merging process can be accomplished as follows: Merge every pure or partial hexahedron having a top area greater than or equal to $\frac{1}{2}\Delta x^2$ with adjacent partial hexahedra having no top or top area smaller than $\frac{1}{2}\Delta x^2$ which have not been merged elsewhere. Denote the resulting space-time polyhedra the intermediate hexahedra. Merge every intermediate hexahedron with all adjacent partial hexahedra having no top or top area smaller than $\frac{1}{2}\Delta x^2$ which have not been merged elsewhere.

After the merging process, we also call the remaining pure and partial hexahedra big hexahedra for equivalence in the finite volume scheme. It is easy to see that a big hexahedron contains no more than a fixed number of pure or partial hexahedra. Actually in most cases the merging process yields big hexahedra consisting of two pure or partial hexahedra. The number of pure or partial hexahedra in the big hexahedron could become larger if the radius of curvature of the moving curve is closer to the mesh size. See Fig. 5.

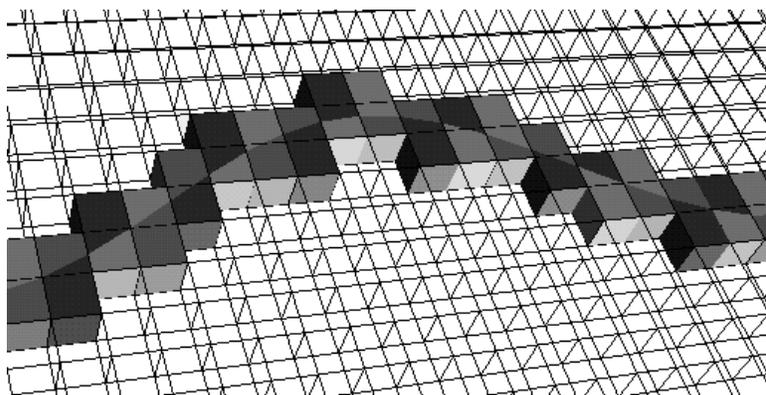


Figure 5. Merged partial hexahedra on both sides of space time interface

Extensions to Three Spatial Dimensions

All aspects of the above construction save one have an obvious extension to three spatial dimensions, and are not discussed here. However, the reconstruction of the space time interface in four space time dimensions will

encounter the following problem. It is based on enumeration, and while this construction will apply theoretically in four dimensions, the number of cases is too large for a practical algorithm. Thus in three spatial dimensions, we construct a space time grid not out of rectangular grid elements, but out of 4D simplices obtained by division of the rectangular grid elements. The grid based reconstruction starts with the intersections of an arbitrary space time interface with edges of simplices. These are assumed to occur at most once per simplex edge, and at interior points of the edge only. Then it can be shown that at most two cases occur for the reconstruction of the interface within the simplex, up to isomorphism. Thus the above proof of the reconstruction, when based on simplices, is not only possible theoretically, but in practice as well.

Acknowledgements

This work was supported in part by the NSF grant DMS 0102480, the ARO grant DAAD19-01-10642, and DOE grants DE-AC02-98CH1086, DEFG0398DP00206, and the Los Alamos contract 267300010141.

References

- [1] J. Glimm, J. W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, and Y. Zeng. Three dimensional front tracking. *SIAM J. Sci. Comp.*, 19:703–727, 1998.
- [2] J. Glimm, J. W. Grove, X.-L. Li, and D. C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comp.*, 21:2240–2256, 2000.
- [3] J. Glimm, X.-L. Li, and Y.-J. Liu. Conservative front tracking in higher space dimensions. *Proceedings of International Workshop on Computational Methods for Continuum Physics and Their Applications (IWC-CPA)*. In Press, 2001. Report SUNYSB-AMS-01-17.
- [4] J. Glimm, D. Marchesin, and O. McBryan. Subgrid resolution of fluid discontinuities II. *J. Comp. Phys.*, 37:336–354, 1980.
- [5] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [6] R. Pember, John Bell, Phillip Colella, William Cruchfield, and Michael Welcome. An adaptive cartesian grid method for unsteady compressible flow in irregular regions. *J. Computational Phys.*, 1995.