SIAM J. SCI. COMPUT.
Vol. 24, No. 1, pp. 208–236


# INTERFACE TRACKING FOR AXISYMMETRIC FLOWS*

JAMES GLIMM†, JOHN W. GROVE‡, AND YONGMIN ZHANG§

**Abstract.** A front tracking method for inviscid gas dynamics is presented. The key constructions and algorithms used in the code are described and the interrelations between shock capturing, interface dynamics, computational geometry, grid construction, and parallelism are discussed for the code as a whole. Validation is carried out by comparing the single mode bubble velocity for Rayleigh–Taylor instability with theoretical models and experimental results. The calculations are validated by mesh refinement studies and by the comparison of the asymptotic limit of the minimum radius $r_{\min} \to \infty$ to a pure planar computation in two dimensions.

**Key words.** front tracking, *FronTier*, interface, Rayleigh–Taylor

**AMS subject classifications.** 76N15, 76T05, 76M20

**PII.** S1064827500366690

## 1. Introduction.

Front tracking is an adaptive computational method in which a lower dimensional moving grid is fit to and follows distinguished waves in a flow. Tracked waves explicitly include jumps in the flow state across the waves and keep discontinuities sharp. A key feature is the avoidance of finite differencing across discontinuity fronts and thus the elimination of interfacial numerical diffusion including mass and vorticity diffusion. In addition, nonlinear instability and postshock oscillations are reduced at the tracked fronts. Front tracking as implemented in the code *FronTier* includes the ability to handle multidimensional wave interactions in both two [21, 29, 33] and three [20, 19] space dimensions and is based on a composite algorithm that combines shock capturing on a spatial grid with a specialized treatment of the flow near the tracked fronts. Applications have included Rayleigh–Taylor (RT) [17, 27, 24, 53, 54] and Richtmyer–Meshkov (RM) [32, 52, 37, 55, 36] instability in two space dimensions and three dimensional planar RT instability [41, 42, 22, 18]. RT instability occurs when a fluid interface is accelerated in a direction opposite to the density gradient across the interface, while RM instability is induced by the refraction of shock waves through a fluid interface. *FronTier* is implemented for distributed memory parallel computers, and some of the fundamental algorithms used in this code are described in [23, 8, 30, 34, 22, 31, 18].

*Received by the editors January 24, 2000; accepted for publication (in revised form) December 4, 2001; published electronically June 18, 2002. This paper first appeared as Los Alamos National Laboratory Report LA–UR–01–448.

http://www.siam.org/journals/sisc/24-1/36669.html

†Center for Data Intensive Computing, Brookhaven National Laboratory, Upton, NY 11793-6000 (glimm@ams.sunysb.edu). This author's work was supported by the MICS Program of the U.S. Department of Energy under grant DE-FG02-90ER25084, by the Department of Energy Office of Inertial Fusion, by the Army Research Office under grants DAAG559810313 and DAAD190110642, by the National Science Foundation grants DMS-9732876 and DMS-0102248, and by Los Alamos National Laboratories under contract C738100182X.

‡Methods for Advanced Scientific Simulations, Computer and Computational Science Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (jgrove@lanl.gov). This author's research was supported by the U.S. Department of Energy.

§Department of Applied Mathematics and Statistics, University at Stony Brook, Stony Brook, NY 11794–3600 (yzhang@ams.sunysb.edu). This author's work was supported by the MICS Program of the U.S. Department of Energy under grant DE-FG02-90ER25084, by the Department of Energy Office of Inertial Fusion, by the Army Research Office under grants DAAG559810313 and DAAD190110642, by the National Science Foundation grants DMS-9732876 and DMS-0102248, and by Los Alamos National Laboratories under contract C738100182X.

We validate our code by comparing the terminal bubble velocity of axisymmetric RT instability to both experimental and theoretical predictions for the limit of incompressible flow. The validation is difficult because the bubble terminal velocity shows considerable oscillation about its limiting value as the density ratio across the interface becomes large, and because of the small time step needed to simulate nearly incompressible regimes.

The paper is organized as follows. Section 2 describes the general features of the front tracking algorithm. Section 3 discusses the geometry package used to represent the tracked fronts. Section 4 describes the basic algorithms used to propagate the front and to couple shock capturing on a computational lattice and the propagation of the tracked fronts. In section 5, we discuss validation calculations for RT instability in axisymmetric geometry. The effects of curved geometry, grid size, and density ratio across the interface on bubble velocity are investigated. We compare the computed value of the terminal bubble velocity with several theoretical predictions and laboratory experiments. Our numerical results lie approximately within the range of the theoretical and experimental values.

**2. The front tracking method.** Examples of tracked fronts include shocks, contact discontinuities, material interfaces, and rarefaction wave edges. The discrete representation of the flow is based on a composite grid that consists of a spatial grid representing the flow field in the bulk fluid, together with a codimension one grid that represents the fronts. Figure 2.1 shows a two dimensional schematic of a time step snapshot of such a grid. The front is represented by a piecewise linear curve, the sections of which are called bonds. In contrast to the spatial grid, which is fixed in time (i.e., Eulerian), the fronts move according to the dynamics of the wave fronts that they represent. A single time step is divided into two processes: the propagation of the fronts and the updating of the solution on the spatial grid.

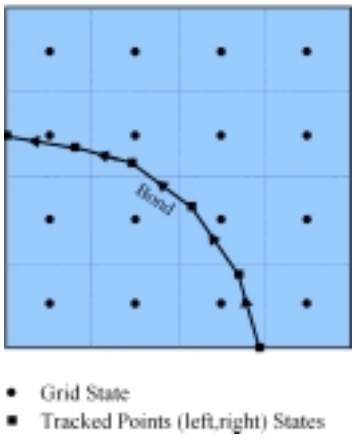



FIG. 2.1. *A representation of the grid for a front tracking computation. The solution is represented on the union of a spatial finite difference grid and a dynamic grid that follows the fronts.*

Front tracking has several features in common with the arbitrary Lagrangian Eulerian (ALE) [35, 13, 2, 3, 4, 44] family of methods in which the solution grid is adapted to the flow. ALE methods are based on a moving mesh formulation of the flow equations where the mesh motion is chosen to maintain desirable features of both the Lagrangian (fluid following) or Eulerian (fixed grid) formulations. Front

tracking is a special case where adaptivity is used on a lower dimensional manifold corresponding to wave fronts.

Two key features are exploited by the tracking. When the orientation and position of a front is known, one can locally rotate into a coordinate system that is aligned with the wave front so that the normal and tangential directions to the front are grid lines in the local coordinate system. Such a grid substantially improves the quality of the solution near the front. Furthermore, explicit representation of the front allows the inclusion of analytic information regarding the rate of change of the flow variables and the velocity of the wave front.

*FronTier* uses a variety of computational tools in its implementation. At the core of these is a geometry package [26, 34, 22, 20, 19] for the description and manipulation of the wave fronts. This package provides services for interface geometry representation and interface manipulation routines for querying, creating, destroying, and modifying the tracked fronts.

Another important operation is to evaluate the flow state at an arbitrary location. This is an essential component in coupling the propagation of fronts with the spatial flow field. This operation, which we call the solution function, is implemented via an interpolation function using the cell values on the spatial grid and the states on the fronts. The front states are bivalved since in general fronts correspond to flow discontinuities. Each front point is associated with two states that correspond to the limit at that point of the spatial field on either side of the front. The key feature of the interpolant is that it respects front discontinuities; i.e., no interpolations occur between states on opposite sides of a tracked front.

Additional packages include libraries for handling interactions between tracked waves, finite difference solvers, Riemann problem solution packages, equation of state packages, flow initialization, and printout. Also, a package for the redistribution of front points is needed to control numerical instabilities produced due to expanding and converging interface sections. The next two sections will describe some of these packages in more detail.

**3. Geometry package.** As mentioned above, one of the core packages in *FronTier* is a geometry package, called the interface library, for the description and manipulation of interfaces. A more detailed description of this package can be found for two dimensional interfaces in [26, 34] and in [22, 20, 19] for three space dimensions; for completeness we will summarize some of the basic terminology here.

A tracked interface is a collection of geometric objects describing the location of a set of fronts at a given time. They consist of discrete representations of points, curves, and surfaces. A curve is a connected oriented piecewise linear list of bonds, which are connectors between adjacent points. A surface is a connected oriented piecewise linear collection of triangles, which are in turn connectors between three adjacent points. Both bonds and triangles are linking objects in the sense that they contain pointers to their neighbors. Each bond points to both the previous and following bonds that share its endpoints. Similarly, triangles share a pair of points along a common side with their neighbor and contain pointers to that neighbor's address. The entire list of triangles that share a common vertex can be found by starting from any member of that set and looping through the adjacent triangles until either a terminating edge is found or until the original triangle is reached and the loop closes. Both surfaces and curves are examples of discrete manifolds with boundaries. The boundaries of surfaces are curves, while the boundaries of curves are called nodes. It is assumed that a valid interface is nonintersecting in the interior of the surfaces and curves. Thus curves

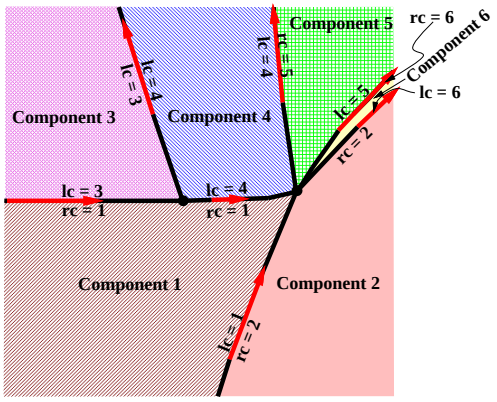



FIG. 3.1. *Components formed by an interface.*

may only intersect along nodes or surfaces along curves. Front intersections produced during propagation correspond to wave interactions and require special treatment to resolve the interaction and untangle the interface.

Interface topology is described in terms of the notion of a component. The wave fronts divide the computational domain into a set of connected components. Each such region is identified by an integer tag that we call the component number or simply the component of that region. Figure 3.1 illustrates the components formed in a neighborhood of a shock refraction we call anomalous reflection [33]. In order to unify the treatment of components across one, two, and three dimensional flows, we introduce a hypersurface data class. Topologically, a hypersurface locally separates space. Hypersurfaces are associated with either points, curves, or surfaces depending on whether the flow is one, two, or three dimensional. As a data structure a hypersurface can be viewed as an adjunct that provides additional topological information for a geometric object. For a two dimensional flow, there is a one-to-one correspondence between curves and hypersurfaces, with each curve containing a pointer to the unique hypersurface to which it is associated. Similarly, in a three dimensional flow the surfaces and hypersurfaces are in one-to-one correspondence. Each hypersurface has a pointer to its associated geometric object. Although implemented as distinct as data structures, we use the standard terminology that identifies a hypersurface with its corresponding geometric object. Thus a hypersurface may refer to either a point, curve, or surface depending on the spatial dimension of the flow, and by points on a hypersurface we mean the points on its associated geometric object.

At each point on a hypersurface there is a specified normal direction, and we speak of the positive or negative side of the hypersurface as that into which or out of which the normal vector points, respectively. Components are identified by associating with each hypersurface a pair of integers that give the component number of its two bounding sectors. These labels are called the positive or negative components of the hypersurface depending on whether they correspond to its positive or negative side. It is required that this indexing be well defined so that separate hypersurfaces bounding a common region must have equal component indices corresponding to their common region. Violations of this condition imply that the interface is tangled, and the wave interactions that produced the tangles must be resolved before the time step is complete.

The interface library provides a number of services including creators and destructors of interface objects, and query functions for the nearest interface point to a given location or the component number at that location. Another important operation determines the normal vector to an interface point. Several possible normal algorithms are implemented. In two space dimensions the simplest normal algorithm computes a secant vector joining the two points adjacent to the given point and then rotates this vector by 90 degrees. In three space dimensions, one algorithm computes the least squares plane fit to the set of adjacent points to the given point and uses the resulting plane normal vector as its output. The interface library is a low level utility, but each data structure can be extended by inheritance so that high level libraries may add additional functionality and override the default behavior of the interface class members.

**4. Time stepping.** *FronTier* is written as a set of hierarchical libraries ranging from basic geometry and topology to physics solvers. The current implementations include inviscid gas dynamics, rate dependent elastoplasticity, and flow in porous media. For simplicity we will restrict the remaining discussion to the case of inviscid gas dynamics. However, many of the generic topics apply to all of the separate physics implementations. Care has been taking in designing the separate libraries in *FronTier* so that physics models are hidden from the lower level modules. These modules can then be used in common across all physical implementations. These generic modules provide inheritance structures that can be tailored to specific models in a physics or even problem dependent fashion.

As mentioned above, we consider inviscid compressible flows. The equations of motion are given by the Euler equations

$$\begin{aligned} \rho_t + \nabla \cdot (\rho \mathbf{v}) + \alpha \frac{\rho u}{x} &= 0, \\ (\rho \mathbf{v})_t + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) + \nabla P + \alpha \frac{\rho u \mathbf{v}}{x} &= \rho \mathbf{g}, \\ (\rho E)_t + \nabla \cdot \rho \mathbf{v}(E + P/\rho) + \alpha \frac{\rho u (E + P/\rho)}{x} &= \rho \mathbf{v} \cdot \mathbf{g}, \end{aligned} \tag{4.1}$$

where the state variables are the mass density $\rho$, the fluid velocity $\mathbf{v}$, the specific total energy $E = e + \frac{1}{2}\mathbf{v} \cdot \mathbf{v}$, the specific internal energy $e$, and the thermodynamic pressure $P$. The vector $\mathbf{g}$ is the net body force per unit mass. The parameter $\alpha$ is equal to 1 if the flow exhibits cylindrical symmetry with respect to the $z$ axis, 2 for spherical symmetry with respect to the origin, and zero otherwise. The variable $u$ is the component of $\mathbf{v}$ in the $x$ direction. System (4.1) describes the laws of conservation of mass, momentum, and energy, respectively, and is closed via a thermodynamic equation of state that relates the density, pressure, and specific internal energy. For simplicity we regard the pressure as a material dependent function, $P = P(\rho, e)$. For multiple material flows we assume no molecular mixing so that distinct fluids are separated by a tracked contact discontinuity across which the equation of state may change. It is traditional to replace $x$ by $r$ in the case of cylindrical or spherical symmetry since in those cases this variable represents the distance from either the $z$ axis or the origin, respectively. Also in the case of cylindrical or axisymmetry, the second independent variable is usually denoted by $z$ and its velocity component by $w$.

A flow chart outlining the main components in the time step algorithm is shown in Figure 4.1. As we see from the chart, the computation can be broken up into roughly three main steps: the propagation of the front, the construction of the interpolation

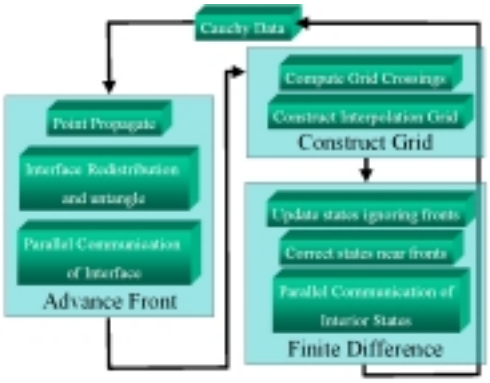



FIG. 4.1. *Schematic of control flow for time stepping.*

grid, and the finite difference or shock capturing solver. At the end of each cycle we obtain the time updated flow state, which includes the location and states at the tracked points and the values of the state variables on the spatial grid. The following subsections will give a brief description of the main features in each of the propagation steps.

**4.1. Advance front.** The advance of the front, i.e., the time step updating of the tracked front positions and states, is broken up into three basic subsets. First, we propagate the points on the fronts ignoring possible wave interactions. Subsequently, the interface is checked for self-consistency to detect wave interactions. If interactions are found, they must be resolved using specialized routines designed to model the appropriate wave interactions. In addition, the interface points may be reinterpolated to improve the distribution of points on the interface elements. Finally, there is a parallel communication step required to include propagation information from neighboring subdomains.

**4.1.1. Propagation of front points.** Point propagate is a fundamental operation in a front tracking computation. This operation is performed at each point on the interface and computes the time updated states and position of the front. Several point propagate algorithms are implemented in *FronTier*, using either directionally split or unsplit methods. Here we describe a directionally split version as specialized to the case of a contact discontinuity.

Figure 4.2 shows a schematic representation of the states used in updating a front point. In the directionally split version, point propagate consists of two parts: normal point propagate and tangential point propagate. The former uses a one dimensional projection of the flow equations into the direction normal to the point $\mathbf{p}$ being propagated. The latter projects the flow equations onto the tangent space of the hypersurface at $\mathbf{p}$ and solves two sets of equations for the flow on either side of the hypersurface. In contrast to normal point propagate there is no assumed discontinuity in the data for the tangential propagation step, and standard shock capturing schemes are used to update the tangential contribution to the flow state. In the directionally split version normal propagate is performed first, and the data for the tangential sweep is the output from the normal sweep. The interface motion is computed as part of the normal sweep, and no additional front motion is included in the tangential sweep. Unsplit implementations of point propagate have the ability to couple interface motion

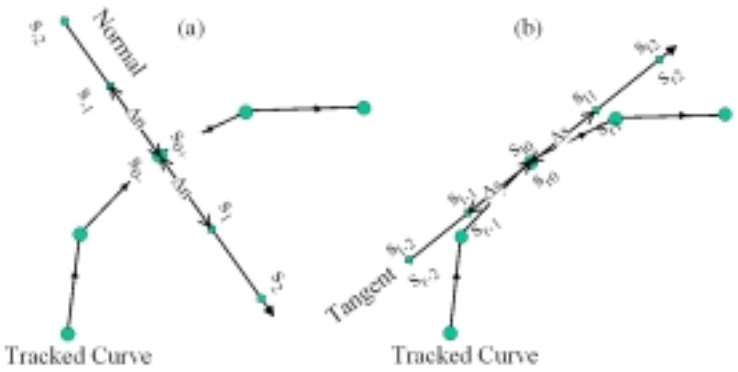



FIG. 4.2. *A schematic showing the stencil of states used in propagating a front point. For simplicity the diagram is shown for two space dimensions. The normal propagate stencil is shown in* (a)*, while* (b) *shows the stencil used in the tangential update.*

with both normal and tangential data. The simplest of these uses Strang splitting [48, 49] (i.e., alternating the order of normal and tangential propagation). The effect of curvature on the interface velocity and states can also be explicitly included in an unsplit algorithm.

*Normal point propagate.* Normal point propagate seeks to solve a generalized Riemann problem for the projection of the flow equations onto the direction normal to the front at a point $\mathbf{p}$. A generalized Riemann problem is an initial value problem with a single embedded discontinuity in a nonconstant flow. We seek to determine the normal velocity of the front at $\mathbf{p}$ and the time updated states at the front. The first step in the normal point propagate algorithm is to evaluate the flow state in the normal direction near $\mathbf{p}$. If $\mathbf{N}$ is the computed normal to the hypersurface containing $\mathbf{p}$, we evaluate states $s_i$ at positions $\mathbf{p}_i = \mathbf{p} + i\Delta n\mathbf{N}$ for $i = 0\pm, \pm 1, \ldots, \pm m$, where $m$ is the stencil radius of the specific normal sweep solver being used. Two state values at the front $s_{0\pm}$ are evaluated corresponding to the states on either side of the discontinuity at the front. Current implementations use either $m = 1, 2$, or 3 depending on the specific solver being used.

The value of $\Delta n$ is determined from the mesh spacing in the coordinate directions, $\Delta x_i$, by the formula

$$\Delta n = \left( \sum_{i=1}^{d} \left( \frac{N_i}{\Delta x_i} \right)^2 \right)^{-\frac{1}{2}},$$

where $d$ is the spatial dimension of the flow. Since $\mathbf{N}$ is a unit vector, $\Delta n$ reduces to the common mesh size for a square grid.

The states $s_{0\pm}$ are given as data on the tracked fronts. The off front states are obtained using the solution function mentioned above.

The projection of system (4.1) into the direction $\mathbf{N}$ yields the one dimensional

system

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial \rho v_N}{\partial \mathbf{N}} + \frac{\alpha N_0}{x}\rho v_N &= 0,\\
\frac{\partial \rho v_N}{\partial t} + \frac{\partial (\rho v_N^2 + P)}{\partial \mathbf{N}} + \frac{\alpha N_0}{x}\rho v_N^2 &= \rho g_N,\\
\frac{\partial \rho \mathbf{v}_T}{\partial t} + \frac{\partial \rho v_N \mathbf{v}_T}{\partial \mathbf{N}} + \frac{\alpha N_0}{x}\rho v_N \mathbf{v}_T &= 0,\\
\frac{\partial \rho E}{\partial t} + \frac{\partial (\rho E v_N + P v_N)}{\partial \mathbf{N}} + \frac{\alpha N_0}{r}(\rho E v_N + P v_N) &= \rho g_N v_N.
\end{aligned} \tag{4.2}
$$

Here $N_0$ is the $x$ component of the normal vector $\mathbf{N}$, $\frac{\partial}{\partial \mathbf{N}} = \mathbf{N} \cdot \nabla$ is the directional derivative in the direction $\mathbf{N}$, $v_N = \mathbf{v} \cdot \mathbf{N}$ and $g_N = \mathbf{g} \cdot \mathbf{N}$ are the normal components of $\mathbf{v}$ and $\mathbf{g}$, and $\mathbf{v}_T = \mathbf{v} - v_N \mathbf{N}$ is the tangential velocity component. The noncharacteristic version of system (4.2) is

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + v_N \frac{\partial \rho}{\partial \mathbf{N}} + \rho \frac{\partial v_N}{\partial \mathbf{N}} + \frac{\alpha N_0}{x}\rho v_N &= 0,\\
\frac{\partial v_N}{\partial t} + v_N \frac{\partial v_N}{\partial \mathbf{N}} + \frac{1}{\rho}\frac{\partial P}{\partial \mathbf{N}} &= g_N,\\
\frac{\partial \mathbf{v}_T}{\partial t} + v_N \frac{\partial \mathbf{v}_T}{\partial \mathbf{N}} &= 0,\\
\frac{\partial e}{\partial t} + v_N \frac{\partial e}{\partial \mathbf{N}} + P\left(\frac{\partial V}{\partial t} + v_N \frac{\partial V}{\partial \mathbf{N}}\right) &= 0,
\end{aligned} \tag{4.3}
$$

where $V = 1/\rho$ is the specific volume. The characteristic form of system (4.2) is

$$
\begin{aligned}
\frac{dv_N}{d\lambda_+} + \frac{1}{\rho c}\frac{dP}{d\lambda_+} + \frac{\alpha N_0}{x} c v_N &= g_N,\\
\frac{dv_N}{d\lambda_-} - \frac{1}{\rho c}\frac{dP}{d\lambda_-} - \frac{\alpha N_0}{x} c v_N &= g_N,\\
\frac{d\mathbf{v}_T}{d\lambda_0} &= 0,\\
\frac{de}{d\lambda_0} + P\frac{dV}{d\lambda_0} &= 0,
\end{aligned} \tag{4.4}
$$

where $c^2 = \left.\frac{\partial P}{\partial \rho}\right|_S$ is the sound speed and $S$ is the specific entropy. More explicitly, in terms of the equation of state $P = P(\rho, e)$, we have $c^2 = \frac{\partial P}{\partial \rho} + \frac{P}{\rho^2}\frac{\partial P}{\partial e}$. The characteristic derivatives are defined by

$$
\begin{aligned}
\frac{d}{d\lambda_\pm} &= \frac{\partial}{\partial t} + (v_N \pm c)\frac{\partial}{\partial \mathbf{N}},\\
\frac{d}{d\lambda_0} &= \frac{\partial}{\partial t} + v_N \frac{\partial}{\partial \mathbf{N}}.
\end{aligned}
$$

From the first law of thermodynamics, $TdS = de + PdV$, we see that the fourth equation in system (4.4) is equivalent to the statement that specific entropy is constant along the $\lambda_0$ characteristic lines.

A comment regarding systems (4.2)–(4.4) is in order. The projection is based on a directionally split solver for system (4.1) in a neighborhood of the front. This

formulation does not explicitly include the effect of curvature of the front on the wave speed and states, nor does it account for changes in the front normal direction over the time step. Both effects are included indirectly in the solution over time since the normal direction is recomputed at every front point and every time step. The main consequence is that the solver will be at best first order accurate near the front for a given time step. Furthermore, the operator splitting into normal and tangential sweeps also introduces a first order error. These errors are consistent with the usual statement that numerical methods are at best first order accurate in regions of large flow gradients. Indeed, tracking means that the flow gradients are infinite at the front. The overall accuracy of the solver, as say measured by the $L_1$ integral of the solution error, will approach the order of accuracy of the interior solver, by which we mean the solver on the spatial grid, as the grid spacing is refined, since the measure of the set near the front approaches zero as the mesh size decreases to zero. This does not imply that the front tracking calculation is less accurate than a shock capturing algorithm since the order of accuracy estimates for such algorithms break down in regions of large gradients. In particular, for a capturing algorithm, the front location is never known to be better than first order.

The point propagate algorithm has three basic parts: slope reconstruction to compute approximations to the flow gradients along the normal line, prediction using Riemann problem solutions, and correction to account for flow gradients on either side of the front and to include body or geometry source terms. At material interfaces surface tension may also be included by modifying the Riemann problem solution to allow a pressure jump proportional to the mean curvature at the point being propagated.

The reconstruction step is similar to that used in many shock capturing methods [51, 10, 28, 40, 50], with one important exception. The reconstruction is used to define a one dimensional interpolant for state values along the normal line, with the added condition that the existence and location of a discontinuity is explicitly known and represented in the reconstructed slopes so that no differencing is performed between states on opposite sides of the front. The profiles are chosen to be linear in some set of appropriate variables (density, pressure, and velocity usually) on the segments $\mathbf{p}_{i-1/2} \to \mathbf{p}_{i+1/2}$ for $i = \pm 1, \ldots, \pm m$ as well as on the segments $\mathbf{p} \to \mathbf{p}_{\pm 1/2}$. Here $\mathbf{p}_{i+1/2} = (\mathbf{p}_i + \mathbf{p}_{i+1})/2$. We wish to compute slopes $ds_i$ for each of these intervals for the reconstructed profile so that the reconstructed state variables are given by the linear profile

$$s(\mathbf{x}) = s_i + ds_i \frac{(\mathbf{x} - \mathbf{p}_i) \cdot \mathbf{N}}{\Delta n}. \tag{4.5}$$

For $\mathbf{x}$ on the line segment from $\mathbf{p}_{i-1/2}$ to $\mathbf{p}_{i+1/2}$, a similar expression applies for the half width intervals from $\mathbf{p}$ to $\mathbf{p}_{\pm 1}$. The following discussion applies to three point slope limiters like the van Leer limiter [51]:

$$ds = \epsilon \min\left(2|s_{i+1} - s_i|, \frac{|s_{i+1} - s_{i-1}|}{2}, 2|s_i - s_{i-1}|\right), \tag{4.6}$$

where $\epsilon$ is the common sign of the forward, central, and backward differences or zero if their signs differ. The extension to higher order slope limiters requires some additional modifications at the off front locations whose limiter stencil overlaps the front. The slopes $ds_{0\pm}$ are first chosen so that the one-sided linear profiles centered at $\mathbf{p}$ connect $s_{0\pm}$ and $s_{\pm 1}$. That is, these innermost slope values are found by differencing the

on front states and the adjacent off front state. At the remaining off front locations standard limiters, such as (4.6), are used to compute the reconstructed slopes. If the sign of the slope $ds_{0\pm}$ is not equal to the sign of $ds_{\pm 1}$, we set $ds_{0\pm} = 0$; otherwise $ds_{0\pm}$ is further reduced to $\text{sign}(ds_{\pm 1}) \min(|ds_{0\pm}|, |ds_{\pm 1}|)$. The result is a piecewise linear profile for the state variables that is monotone on either side of the front.

The prediction step is illustrated in the upper right frame shown in Figure 4.3. We use the front states $s_{0\pm}$ as data for a one dimensional Riemann problem for system (4.2). Each tracked wave carries a wave type that identifies it with one of the waves from the Riemann problem solution and the velocity $w_0$ of that wave is used to predict the front motion in the normal direction. The velocity $w_0$ can be interpreted as the velocity of the front at the beginning of the time step. In addition the Riemann problem solution yields states $s_{m-}$ and $s_{m+}$ on either side of the new wave. For simplicity, in the following we will consider only the case of the propagation of a contact discontinuity. In this case $s_{m-}$ and $s_{m+}$ agree with the usual midstates from the Riemann problem solution, and $w_0$ is equal to the common normal velocity component of the two midstates. A description of algorithms that include the case of shocks can be found in [8].

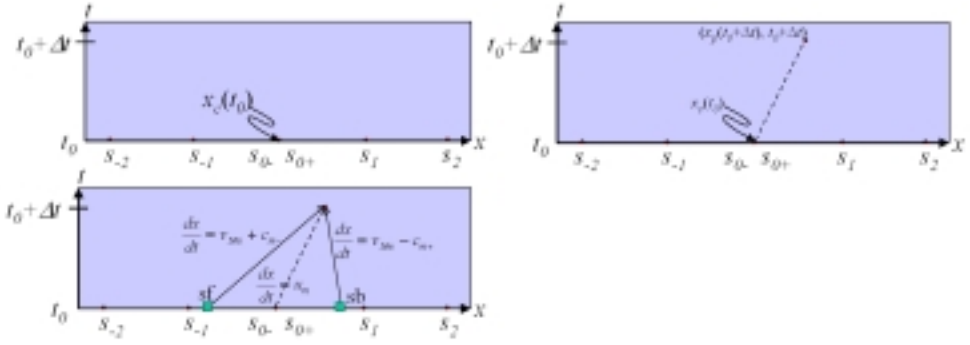


FIG. 4.3. *In normal point propagate the flow state is projected onto a line normal to the interface point. A one dimensional Cauchy problem is solved to compute the updated front position and state.*

The correction step is based on the method of characteristics. As seen in the lower left frame of Figure 4.3, we trace back the incoming characteristics from the predicted new front position using the velocity and sound speeds of the midstates computed from the Riemann problem solution to approximate the characteristic slopes. We use (4.5) to compute states at the feet of the traced back characteristics. For a contact discontinuity we obtain two states, $s_f$ and $s_b$, on the left and right of the original interface at the feet of the tracked back characteristics. Together with the states on the front, we then solve a discrete version of the characteristic system (4.4) to obtain the time updated states on the front and a time updated velocity of the front, $w_1$, at time $t + \Delta t$. The net interface velocity is then found by time averaging $w = (w_0 + w_1)/2$, and the new position of $\mathbf{p}$ is given by

$$\mathbf{p}(t + \Delta t) = \mathbf{p} + w\mathbf{N}\Delta t. \tag{4.7}$$

Integrating system (4.4) along the incoming characteristics at the contact discontinuity, and using the Rankine–Hugoniot conditions, $P_- = P_+$ and $v_{N-} = v_{N+} = v_N$, across the contact we obtain an integral equation system for the state values at time

$t + \Delta t$:

(4.8)
$$
\begin{aligned}
v_N - v_{fN} + \int_t^{t+\Delta t} \frac{dP}{\rho c}\bigg|_{\frac{dx}{dt}=v_N+c} + \int_t^{t+\Delta t} \frac{\alpha N_0}{x} c v_N dt\bigg|_{\frac{dx}{dt}=v_N+c} &= \int_t^{t+\Delta t} g_N dt\bigg|_{\frac{dx}{dt}=v_N+c}, \\
v_N - v_{bN} - \int_t^{t+\Delta t} \frac{dP}{\rho c}\bigg|_{\frac{dx}{dt}=v_N-c} - \int_t^{t+\Delta t} \frac{\alpha N_0}{x} c v_N dt\bigg|_{\frac{dx}{dt}=v_N-c} &= \int_t^{t+\Delta t} g_N dt\bigg|_{\frac{dx}{dt}=v_N-c}, \\
\mathbf{v}_{T-} &= \mathbf{v}_{0-T}, \\
\mathbf{v}_{T+} &= \mathbf{v}_{0+T}, \\
e_- - e_{0-} + \int_t^{t+\Delta t} P dV\bigg|_{\frac{dx}{dt}=v_N} &= 0, \\
e_+ - e_{0+} + \int_t^{t+\Delta t} P dV\bigg|_{\frac{dx}{dt}=v_N} &= 0.
\end{aligned}
$$

The last two integrals in system (4.8) are taken along the left or right side of the contact, respectively.

We will now describe three different techniques for approximating the solution to system (4.8). The first method, which we call MOC+RH (for method of characteristic plus Rankine–Hugoniot), uses central differences in time to approximate the above integrals. This is equivalent to approximating the integrals using a two point trapezoidal rule quadrature. For simplicity we assume constant gravity. The resulting system then becomes

(4.9)
$$
\begin{aligned}
v_N - v_{fN} + \frac{1/(\rho_f c_f) + 1/(\rho_- c_-)}{2}(P - P_f) + \alpha N_0 \frac{c_f v_{fN}/x_f + c_- v_N/x}{2}\Delta t &= g_N \Delta t, \\
v_N - v_{bN} - \frac{1/(\rho_b c_b) + 1/(\rho_+ c_+)}{2}(P - P_b) - \alpha N_0 \frac{c_b v_{bN}/x_b + c_+ v_N/x}{2}\Delta t &= g_N \Delta t, \\
\mathbf{v}_{T-} &= \mathbf{v}_{0-T}, \\
\mathbf{v}_{T+} &= \mathbf{v}_{0+T}, \\
e_- - e_{0-} + \frac{P_0 + P}{2}(V_- - V_{0-}) &= 0, \\
e_+ - e_{0+} + \frac{P_0 + P}{2}(V_+ - V_{0+}) &= 0.
\end{aligned}
$$

After the trivial elimination of the tangential velocity components the result is a system of four nonlinear equations in the eight unknowns $\rho_-$, $e_-$, $c_-$, $\rho_+$, $e_+$, $c_+$, $P$, and $v_N$ at the updated contact position. The remaining four equations are given by the equations of state on each side of contact:

(4.10)
$$
\begin{aligned}
P &= P_-(\rho_-, e_-), \\
P &= P_+(\rho_+, e_+), \\
c_- &= c_-(\rho_-, e_-), \\
c_+ &= c_+(\rho_+, e_+).
\end{aligned}
$$

The combined systems (4.9) and (4.10) can be solved iteratively to yield the approximate solution to system (4.8). The value of $x$ used in above system is that obtained

in the prediction step; more generally we could recompute $x$ for each iteration using (4.7) to get the value of $x$ from the last iteration.

In view of the various first order approximations already in place at the interface, there would seem to be little advantage in using the formally second order approximation to system (4.8) given by the systems (4.9) and (4.10). Thus a second integration method using only first order approximations to the integrals in system (4.8) is provided by a backward difference quadrature. We call this option CHEAP MOC+RH. The resulting system becomes linear in $e_-$, $V_-$, $e_+$, $V_+$, $P$ and $v_N$:

$$\begin{aligned} v_N - v_{fN} + \frac{1}{\rho_f c_f}(P - P_f) + \alpha N_0 \frac{c_f v_{fN}}{x_f}\Delta t &= g_N \Delta t, \\ v_N - v_{bN} - \frac{1}{\rho_b c_b}(P - P_b) - \alpha N_0 \frac{c_b v_{bN}}{x_b}\Delta t &= g_N \Delta t, \\ e_- - e_{0-} + P_0(V_- - V_{0-}) &= 0, \\ e_+ - e_{0+} + P_0(V_+ - V_{0+}) &= 0. \end{aligned} \tag{4.11}$$

As in the MOC+RH case the system is closed by the equations of state relations between pressure, energy, and density on either side of the interface.

The third algorithm, which we call RIEMANN, exploits the close relation between the solution of a Riemann problem and the method of characteristics. If we ignore source terms, the one dimensional Riemann problem solution with data $\rho_-$, $P_-$, and $v_-$ on the left and $\rho_+$, $P_+$, and $v_+$ on the right is given by the solution to system

$$\begin{aligned} u - u_- + \frac{P - P_-}{m_-} &= 0, \\ u - u_+ - \frac{P - P_+}{m_+} &= 0, \end{aligned} \tag{4.12}$$

where

$$m_\pm = \begin{cases} \int_P^{P_\pm} \frac{dP}{\rho c}\Big|_{S=S_\pm} & \text{if } P < P_\pm, \\ \rho_\pm c_\pm & \text{if } P = P_\pm, \\ \sqrt{\frac{P-P_\pm}{V_\pm - V}} & \text{if } P > P_\pm \end{cases} \tag{4.13}$$

is the mass flux across the wave in the Riemann problem solution. When $P < P_\pm$ the wave is a centered rarefaction, and the integral in (4.13) is taken at constant specific entropy. If $P > P_\pm$, the wave is a shock and $V_\pm$ is computed as the solution to the Hugoniot equation

$$e - e_\pm + \frac{P(\rho, e) + P_\pm}{2}(V - V_\pm) = 0. \tag{4.14}$$

It is well known that the change in entropy across a weak wave is third order in the pressure jump across the wave, so the two expressions for the mass flux in (4.13) agree to third order in $\Delta P$ for weak waves. Comparing the form of the rarefaction wave solution for a Riemann problem and the characteristic equation (4.8) we see that the equations are identical, provided we ignore entropy changes across the incoming characteristic. Indeed, for a Riemann problem solution with a rarefaction wave the corresponding midstate is found by integrating along the characteristic of the opposite family that crosses the wave. This suggests the following algorithm for approximating the solution to system (4.8), provided the flow gradients on either side of the contact are sufficiently small.

1. Solve the Riemann problem with left state $s_f$ and right state $s_{0-}$ to obtain an approximation $s_{ll}$ to the left state on the contact at time $t + \Delta t$. That is, $s_{ll}$ is the right midstate from the Riemann problem solution using data $s_f$ and $s_{0-}$.
2. Solve the Riemann problem with left state $s_{0+}$ and right state $s_b$ to obtain an approximation $s_{rr}$ to the right state on the contact at time $t + \Delta t$. In this case $s_{rr}$ is the left midstate from this Riemann problem solution.
3. Solve the Riemann problem with left state $s_{ll}$ and right state $s_{rr}$ to obtain an approximation to the states $s_l$ and $s_r$ on the contact at the new time.

The third step above is designed to enforce the proper jump conditions at the wave. This scheme can also be described dynamically. The solution to the Riemann problems between the off front and on front states describe a pair of incoming waves incident on the contact. The third Riemann problem then describes the interaction between these two incoming waves.

If source terms exist, either due to body forces or gravity, the computation is completed using operator splitting to incorporate the source. From system (4.3) we see that this can be done by adding $g_N \Delta t$ to the velocity of the two midstates and subtracting $\frac{\alpha N_0}{x} \rho v_N$ from the density on either side of the contact. Here we simply use the original position of the contact as the value of $x$.

*Tangential point propagate.* The previous discussion described the operations used to compute the normal motion of front points. The tangential component is computed by solving a pair of finite difference equations corresponding to the projection of the Euler equations onto the tangent space at the front. More specifically, the tangential projection of system (4.1) is given by

$$\begin{aligned} \rho_t + \nabla_T \cdot (\rho \mathbf{v}_T) + \alpha \mathbf{v}_T \cdot \mathbf{e}_x \frac{\rho}{x} &= 0, \\ (\rho \mathbf{v})_t + \nabla_T \cdot (\rho \mathbf{v} \otimes \mathbf{v}) + \nabla_T P + \alpha \mathbf{v}_T \cdot \mathbf{e}_x \frac{\rho \mathbf{v}_T}{x} &= \rho \mathbf{g}_T, \\ (\rho E)_t + \nabla_T \cdot \mathbf{v}(\rho E + P) + \alpha \mathbf{v}_T \cdot \mathbf{e}_x \frac{\rho E + P}{x} &= \rho \mathbf{v}_T \cdot \mathbf{g}_T, \end{aligned} \tag{4.15}$$

where $\nabla_T$ is that spatial gradient in the tangent plane and $\mathbf{e}_x$ is the unit vector in the $x$ direction. The data for system (4.15) is found by projecting the states obtained from the normal sweep onto the updated tangent plane at $\mathbf{p}$. The case for two dimensions is illustrated in Figure 4.2(b). Using the value

$$\Delta s = \left( \sum_{i=1}^{d} \left( \frac{T_i}{\Delta x_i} \right)^2 \right)^{-\frac{1}{2}}, \tag{4.16}$$

where $\mathbf{T}$ is the tangent to the curve at the given point $\mathbf{p}$, we choose pairs of states $s_{lri}$ for $i = -m, \ldots, m$ by interpolating the states on either side of the front at distances $\Delta s$ when measured along the arc length of the curve. As before, $m$ is the stencil radius of the finite difference method being used (2 for the case illustrated). The situation for three space dimensions is slightly more complicated. We wish to sample states $s_{lrij}$ $i, j = -m, \ldots, m$ on the surface that map onto a lattice on the tangent space. Given a choice of two orthogonal tangent vectors $\mathbf{T}_0$ and $\mathbf{T}_1$ we interpolate along the surface in the two directions as in the two dimensional case. The choice of the tangent vectors is not unique for three dimensional flows. For contact discontinuities we choose $\mathbf{T}_0$ to be the direction of the velocity shear across the interface (which

must always be tangent to the contact surface); otherwise $\mathbf{T}_0$ is obtained by selecting $Mi$ and $mi$ such that $|N_{Mi}| = \max(|N_i|)$ and $|N_{mi}| = \min(|N_i|)$, setting the third coordinate of $\mathbf{N}$ to zero and rotating the resulting vector in the $Mi - mi$ plane by 90 degrees. In either case, $\mathbf{T}_1 = \mathbf{N} \times \mathbf{T}_0$. Using the data from the normal sweep we solve system (4.15) twice on either side of the front to compute the contribution of tangential flow gradients to the states on the front. Since the tangential sweep reduces to a regular finite difference stencil any of a variety of standard solvers may be used to compute the solution. In contrast to the normal sweep, no additional motion is accounted for in the tangential sweep.

**4.1.2. Untangle and redistribution.** Front tracking contains a feature in common with all Lagrangian type methods; due to the discretization it is possible for the interface grid to become tangled as it moves. This situation must be detected and resolved before the computation continues. At the end of the point propagation steps described above, the entire interface is tested for intersections.

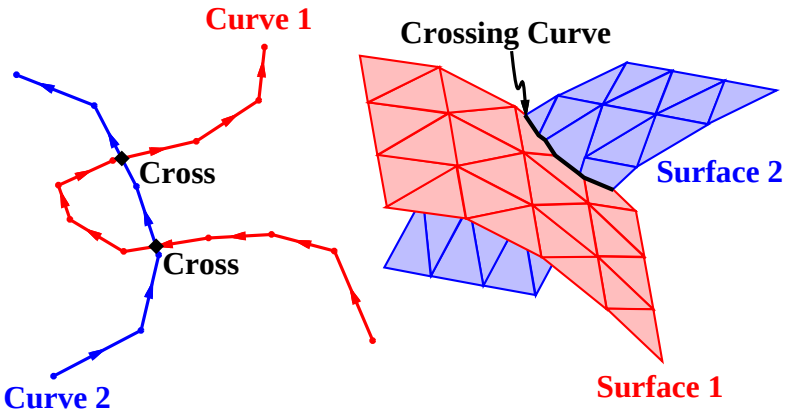



FIG. 4.4. *Tangled fronts.*

Figure 4.4 illustrates the tangling of two curves and two surfaces. The intersection algorithm tests each pair of interface elements (bonds or triangles, respectively) for a possible intersection. The results of this calculation are assembled into a set of data structures describing the intersection. The crossing data includes the addresses of the hypersurfaces that intersect and the location of the crosses. In three dimensions the line segments forming the cross between two surfaces are assembled into a curve describing the intersection.

Intersections between fronts correspond to wave interactions. A detailed discussion of the techniques for resolving such interactions is beyond the scope of this article. See [30, 33, 34] for a description of untangle algorithms for the interaction of shock waves and material interfaces. For contact discontinuities we proceed using a simple mechanical description of the interaction; basically such tangles correspond to either the pinch-off or merger of separate sections of fluids. If the two interacting materials are the same, the interaction is modeled by simply deleting the overlapping segments. Several possibilities for two dimensional interactions are shown in Figure 4.5. The basic algorithm for resolving two dimensional tangles between contacts can be found in [21].

The resolution of three dimensional tangles is more complicated. The algorithm

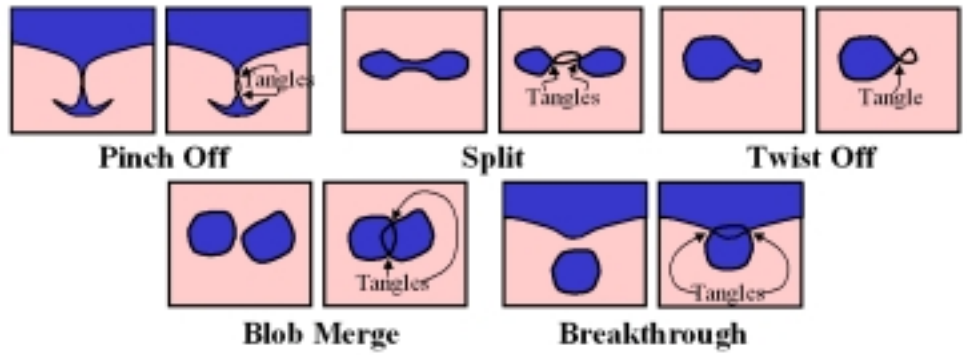



FIG. 4.5. *Types of two dimensional tangles.*

described in [21] has been extended to three dimensions, but the geometric issues are more complex. An alternative method proposed by X. L. Li and described in [20] and [19] uses a form of interface reconstruction similar to that used in level set type methods [46] to simultaneously resolve interface interactions for two fluid flows as well as redistribute the interface points. This method, which we call grid based interface reconstruction, uses a mapping of the components numbers for the two fluids onto a rectangular lattice, followed by a cell by cell rebuilding of the interface sections. The rebuilding uses the component numbers of the cell corners and the interface crossing along the cell edges to reconstruct the surface sections contained in each cell. The method has proven to be quite robust in practice and can be combined with both the direct untangle referred to above as well as the redistribution methods described below to produce a hybrid method that combines the desirable features from both procedures. An illustration of the type of complex fronts that can be handled is shown in Figure 4.6.

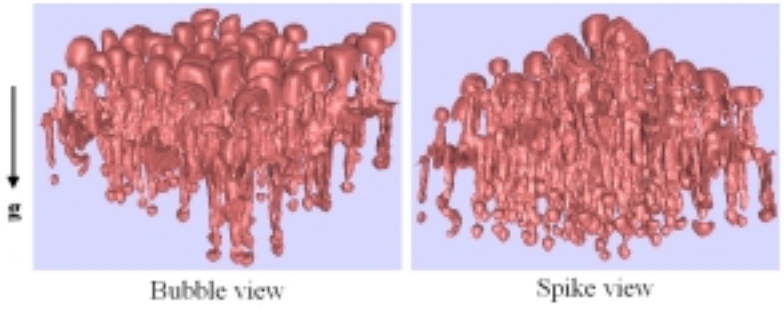



FIG. 4.6. *A three dimensional RT unstable interface. The calculation began as a plane perturbed connected interface. By the time shown in the figure, bifurcations have detached numerous blobs of material from the original interface.*

Another operation that is performed at the end of the propagation of the front is the redistribution of the points describing the interface. As front sections expand or contract, the finite resolution of the interface leads to a poor distribution of interfacial element sizes, which can in turn lead to the overdevelopment of interfacial instabilities. Therefore we periodically reinterpolate the front points, creating a new set of elements with a more equitable size distribution. The algorithm in two space dimensions is

relatively simple. Starting from a node of a curve, we add new points at approximately equal distances when measured along the arclength of the curve and then delete the original curve points between the newly inserted points. The metric used is the same as given by (4.16), where the tangent direction evolves along the curve. The specific value of the spacing is a user defined quantity. Typically, a value of $\Delta s$ of about 75% of the computational mesh size is used to govern the redistribution. Figure 4.7 shows a schematic of the redistribution of a single curve.

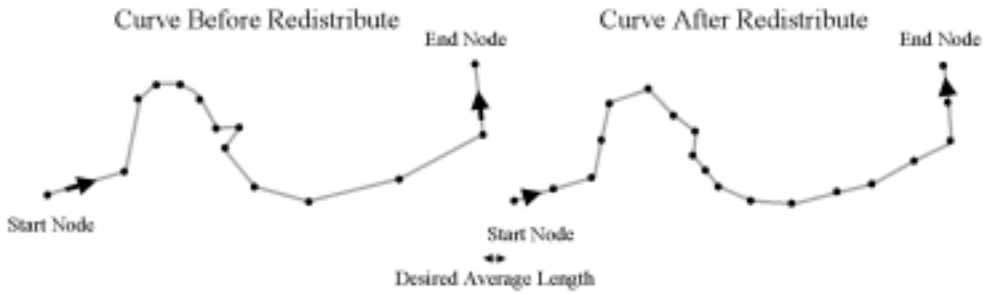



FIG. 4.7. *The two dimensional redistribution of a curve.*

As in the untangle algorithm, the redistribution of surfaces for three dimensional flows is considerably more complex. Here we adopt a different strategy from the two dimensional case that uses local operations on the interface elements. The triangles that make up a given surface are sorted according to their size and aspect ratios. Small triangles are merged with neighbors, while large triangles are subdivided. Bad aspect ratio triangles are handled by a couple of techniques. Long thin triangles can be merged with neighbors, or we may flip the diagonals of a pair of triangles to improve the aspect ratios. Figure 4.8 shows a schematic of some of the elementary operations on triangles, while Figure 4.9 shows an interface before and after redistribution.

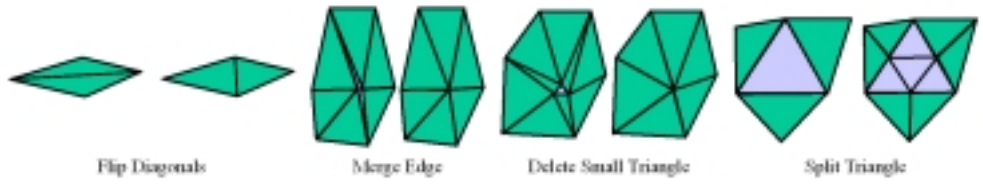



FIG. 4.8. *Elementary triangle redistributions.*

**4.1.3. Parallel communication of the front.** *FronTier* is implemented to use spatial domain decomposition parallelism. The implementation assumes a distributed memory computer and the low level communications use the message passing interface (MPI) standard. The domain decomposition is based on a user defined rectangular partition of the computational domain into geometric subdomains. Each subdomain is extended by a ghost region of some specified width. The width of the ghost regions depends on the particular finite difference or shock capturing method being used and must at a minimum be such that an entire time step update of the interior (i.e., excluding ghost regions) of the subdomain can be performed without any further data from the adjacent regions. An additional constraint from the front requires that all geometric information (e.g., normal vectors) for front points in the interior of a

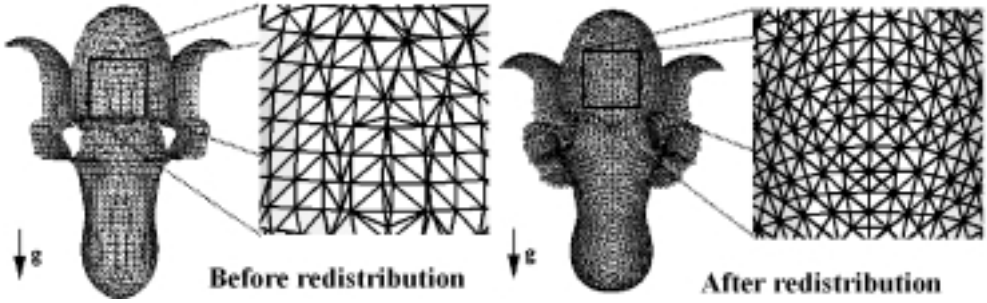



FIG. 4.9. *An RT unstable interface before and after redistribution.*

subdomain can be performed using only data within the union of the subdomain and its ghost region. For the solvers currently implemented in *FronTier* typical ghost regions widths are 4–5 mesh blocks. At the beginning of a time step it is assumed that all ghost regions are synchronized with their adjacent neighbors; i.e., the data in the ghost region is a copy of the corresponding data from the adjacent subdomain.

The final step in the propagation of the front is communication of the propagated front information in the interior of a subdomain to its neighbors. Four basic operations are provided to accomplish the communication.

1. *Clip interface.* Given a interface on some region, construct a copy of that interface that is clipped to a specified subregion.
2. *Join interfaces.* Given two interfaces on adjacent regions, construct a new interface by joining the two interfaces across the common boundary of the two regions.
3. *Send interface.* Broadcast an interface data structure from one subdomain to another.
4. *Receive interface.* Receive an interface that has been broadcast from another subdomain.

An important aspect of the send/receive operation for interfaces involves the reconstruction of pointer information from the interface data structure. This is accomplished by replacing the actual machine addresses in the sending blocks by relative addresses with respect to the message block. From this information it is then possible to reconstruct new addresses for the pointers in the reconstructed interface.

The basic front communication proceeds as follows. First each subdomain replaces its interface by a version clipped to the interior of the subdomain. The subdomain then clips copies of its interface onto slices at the subdomain boundaries that have widths equal to the ghost region width in the $x$ direction and extend to the width of the interior of the subdomain in the other coordinate directions. These slices are then broadcast to the adjacent subdomains which then attach the received sections onto their corresponding boundaries. The process is then repeated in the remaining coordinate directions except that in the subsequent passes the slices are extended to include the ghost regions that have been filled in by the previous communications. A significant advantage of this procedure is that corner regions are handled implicitly and do not require special treatment.

**4.2. Grid construction.** As mentioned above, *FronTier* uses a hybrid grid in which the flow state is represented on the union of a spatial grid together with a codimension one grid that describes the front. Constructing the data structures needed

to couple these two grids is the next step in time update process.

**4.2.1. Grid crossings.** The first data structure that describes the coupling between the spatial grid and the front contains a list of all crossings of the front with the dual lattice defined by joining the cell centers of the computational lattice. Given a computational cell center and a coordinate direction, the grid crossing list contains an ordered list of all intersections of the front with the line segment between that cell center and its neighbor in the specified direction. At each intersection a point is interpolated at the crossing location together with a pair of states corresponding to the two sides of the front at the crossing point. This crossing list also includes the component numbers on either side of each crossing.

The crossing list is constructed by looping over all interface elements (bonds in two dimensions and triangles in three dimensions) and computing the dual lattice line segments that intersect each element. Each crossing is then recorded in a list corresponding to the segment where it occurs.

While mathematically straightforward, the grid crossing algorithm is computationally expensive and occupies a significant portion of the overhead for a time step.

**4.2.2. Interpolation grid.** As we saw from the point propagate algorithm described in section 4.1.1 a critical component in the algorithm is the evaluation of the flow field at arbitrary locations. This is accomplished by constructing a constrained interpolant using the data on both the spatial and interfacial grids.

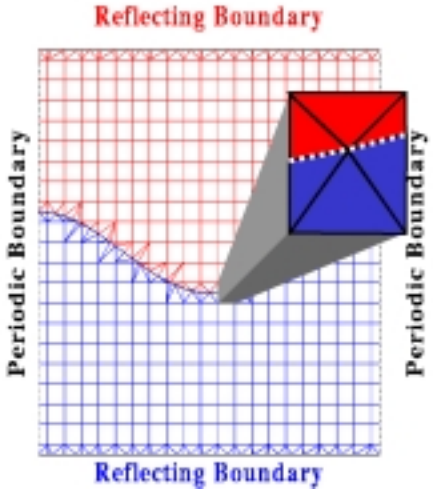



FIG. 4.10. *A triangulated grid.*

In two space dimensions this interpolant is constructed using a triangulated grid based on the dual lattice, the front, and the crossing list described above. Figure 4.10 shows an example of such a triangulated grid for the early stages of an RT calculation. The grid consists of two parts: rectangles joining four adjacent computational cell centers that are disjoint from the front and local triangulations of dual lattice cells crossed by the front. Each triangulation is constrained so that no triangle is formed using an edge that crosses the front. It is beyond the scope of this article to describe the triangulation algorithm in detail, but briefly it consists of identifying polygonal sections within a dual lattice cell corresponding to a connected region bounded by either front lines or the cell boundary. These subregions are then triangulated using

a divide and conquer type method that seeks to split the polygon by a line segment from the vertex with largest interior angle to another vertex that is visible to the given vertex from within the polygon. Note that no assumption regarding the convexity of this polygon can be made.

Once the triangulated grid has been constructed, the interpolant is defined by linear interpolation on triangles and bilinear interpolation on the rectangles.

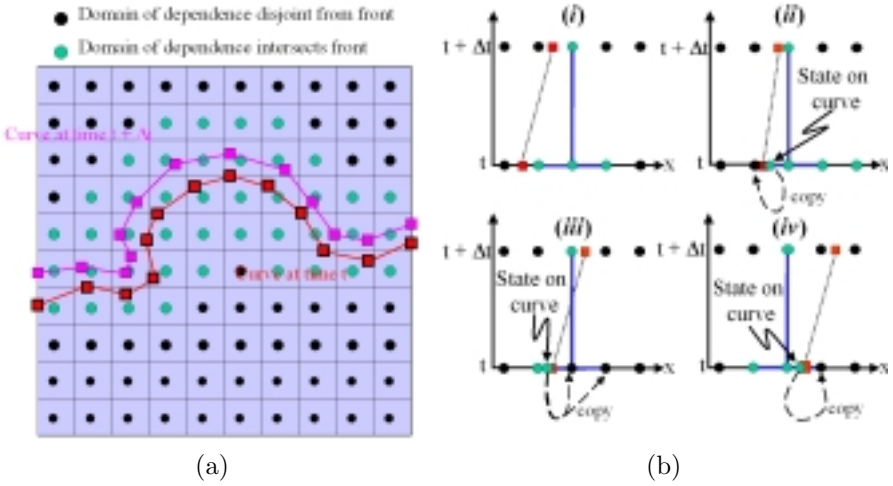
The complexity of constructing constrained tetrahedrazations has led to a choice of a different interpolation algorithm in three space dimensions. Here we do not construct the tetrahedra in advance but instead construct elements on the fly that contain the interpolation location. As in the two dimensional case we first construct a list of dual lattice cells that do not overlap the front, upon which trilinear interpolation will be used exactly as before. However, for the front intersecting cells we instead simply record the point locations (cell corners, front points, and interface dual lattice crossings) together with the component numbers associated with those points. Note that the component numbers are bivalued at front points and interface lattice crossings. The interpolation algorithm then proceeds as follows. When a location with a specified component is found to lie within a specified dual lattice cell, the list of points within the closure of that cell that have the specified component is processed to find an optimal set of four points that contain the desired location in their convex hull (i.e., the tetrahedron formed using the four points as vertices). The optimization is based on the distances of the vertices from the interpolation location so that tetrahedra formed by points closer to the interpolation location are selected preferentially over those formed by more distant points. Linear interpolation is then used on the optimal tetrahedron to compute the state value at the desired location. Since the component numbers of the four points in the interpolation tetrahedron agree, the interpolation uses only data from the appropriate side of the interface. In the event that no tetrahedra are found that both contain the interpolation location and satisfy the component constraint we project the interpolation location onto the nearest front location with the required component and use linear interpolation along the front element containing the projected position.

**4.3. Finite difference update.** The final phase of the time step update consists of computing the time updated states on the spatial grid. In the current implementation this spatial grid is simply a rectangular lattice, and the finite difference scheme uses directional splitting into one dimensional sweeps in the coordinate directions. Strang splitting [48, 49] is used to provide higher order accuracy. By this we simply mean that the order of the one dimensional sweeps is alternated on each time step to cover all possible permutations of the sweep directions.

Several different shock capturing methods are currently implemented in *FronTier* that can be used for the one directional sweeps. These include the Lax–Wendroff method [38, 45], the Colella piecewise linear method [10], and a second order monotone upwind scheme for conservation laws (MUSCL) scheme developed by I-L. Chern. The details for this MUSCL scheme are unpublished, but it is basically similar to those described for the piecewise parabolic method [11] except that it is simplified to use a linear reconstruction. A variety of Riemann solvers are provided for use with the upwind schemes. These include exact Riemann solvers using either a fixed point type iteration as described in [9] or a secant type iteration to compute the intersection of the wave curves. Several approximate Riemann solvers are also provided such as the BCT solver [1], the Dukowicz solver [14], and a simple linearized solver. All Riemann solvers are implemented to use a general equation of state. The code structure in

*FronTier* makes it relatively easy to incorporate additional finite difference operators or Riemann solvers.

For reasons of efficiency the finite difference update on the spatial lattice is divided into two parts. In the first pass, which we call the regular sweep, all interface information is ignored, and the cells are updated using shock capturing on the rectangular lattice. A second pass is then used to recompute the state values for those cells whose domain of dependence overlaps the interface. Figure 4.11(a) shows an illustrative example of the division of a two dimension computation into cells whose domains of dependence are disjoint from the front (*regular cells*) and those whose domains of dependence intersects the front (*irregular cells*). The determination of the irregular cells uses the interface crossing list described in the previous section. A cell is irregular if at least one of the line segments connecting two adjacent points in its domain of dependence stencil contains an interface crossing. For example, in the illustration of Figure 4.11(a) we assume a three point directionally split explicit solver. This means that the domain of dependence of a cell is the nine point lattice formed by that cell and its neighbors. Using the crossing list we check whether there are interface crossings on any of the line segments joining a pair of cell centers in this sublattice. If no crossings are found, then the updated value for the state at the cell center as determined by the first pass sweep is correct. Otherwise we proceed to construct an alternative finite difference stencil that uses only state information from the same side of the interface as the position of the cell center being updated with respect to the propagated interface.




(a) (b)

FIG. 4.11. *Update of cells with irregular stencils.* (a) *shows the division of cells into regular and irregular cells depending on whether their domain of dependence intersects the front.* (b) *illustrates the setting of the pseudostencil states by copying nearby front states. The dotted line in* (b) *shows the space time propagation of the front point near the cell being updated.*

The algorithm we use to construct the modified stencil is based on a simple extrapolate by constant state algorithm [25]. At each irregular cell we construct a pseudostencil state structure that contains the information needed to update that single location. Initially, this structure contains copies of the state data from the cell centers making up the particular cell's domain of dependence. We then proceed to replace any state information in this pseudostencil that would correspond to differencing across a front. This is illustrated for one space dimension in Figure 4.11(b). If the front enters

the stencil during the time step but does not cross the cell center being updated, as shown in Figure 4.11(b)*(i)*, then that point is treated as regular. Figure 4.11(b)*(ii)* shows another case where a front separates the location being updated from the its adjacent cell's center. We copy the state from the grid crossing along this segment that is nearest the point being updated onto the pseudostencil cells that lie on the other side of the front along this line. Figure 4.11(b)*(iii)* illustrates another case in which the cell center being updated is itself crossed by the front. In this case we search for the nearest crossing whose component agrees with the time updated component of that cell (note that at this point we have available both crossing lists from time $t$ and $t + \Delta t$) and copy the corresponding crossing state onto the appropriate locations. In multiple dimensions we construct cells on the axes through the stencil center first and then proceed outward from the axes already computed. In three dimensions we can alternate the order of the spreading onto the remain coordinates to average out any directional bias this algorithm might induce.

Once a properly constructed pseudostencil is generated, it is passed off to the same finite difference or shock capturing solvers as used in regular sweep to recompute the cell state.

Although conceptually simple, this two pass process and the extrapolation by constant state method has proven to be quite effective as we will see from the quality of the solutions discussed in the next section. This algorithm shares some features in common with, but is older than, the ghost cell extrapolation method of Fedkiw et al. [15]; in our case we extrapolate the entire state rather than selected fields.

The final operation in the time step algorithm is the communication of the time updated spatial grid states to the adjacent subdomains. This is done in a similar way to the communication of the interfaces. We extract strips of states from the boundaries of each subdomain and transmit that information to the adjacent subdomain which then uses this data to fill in its ghost cells.

**5. Validation.** The evolution of RT instability has been the focus of investigations for many years due to its wide range of applications and the scientific interest in the complex fluid flows it produces. RT instability occurs at a fluid interface whenever the density gradient is opposed to the acceleration gradient across the interface. The instability is manifested as a fingering behavior at the interface. Relatively broad fingers of the lighter fluid, called bubbles, penetrate into the heavier fluid, while narrower spikes of the heavier fluid are injected into the lighter. This instability appears in many interesting physical situations. Incompressible examples include salt domes that rise on geologic time scales. Thunder cloud systems often display the characteristics of incompressible RT instability when colder and hence heavier air becomes suspended above warmer lighter air. Compressible flow examples of RT instability include the laser implosion of deuterium-tritium fusion targets, electromagnetic implosion of metal liners, and the overturn of the outer portion of the collapsed core of massive stars. For a review article giving examples and phenomenology of the RT instability, see Sharp [47]. In the discussion below we study axisymmetric RT instability in a cylindrical tube.

When a heavier fluid is suspended above a light fluid in the presence of a gravitational field the fluid is in a state of unstable equilibrium. If the fluid is slightly disturbed the two fluids interpenetrate to form bubbles of rising light fluid and spikes of falling heavy fluid. The bubbles and spikes each move toward the opposite fluid with accelerating speed. For the single mode disturbance considered here, this acceleration decreases with time towards zero due to a balance that is established between

buoyancy and form drag forces. This balance leads to a constant terminal velocity. Our goal is the conduct numerical simulations of RT instability that approach this terminal velocity regime.

We assume the fluid is rotationally symmetric about the $z$ axis. For the simulations discussed here we used a domain $r_0 \le r \le r_1$ and $z_0 \le z \le z_1$ with $r_0 = 0$, $r_1 = 0.5$, $z_0 = 0$, and $z_1 = 3$. The interface perturbation was given by

$$z = \bar{z} - a_0 \cos\left[\pi \frac{r - r_0}{r_1 - r_0}\right].$$

Where $\bar{z} = 1.5$ and $a_0 = 0.015$, this perturbation has a wave length $\lambda = 1$. Gravity is given by $\mathbf{g} = g\mathbf{e}_z$, where $g = -0.14$ and $\mathbf{e}_z$ is the unit vector in the positive $z$ direction. We assume both fluids are perfect gases with $\gamma = 1.4$. The sound speed $c$ is then given by $c^2 = \gamma P/\rho$. The strength of the body source is measured by the compressibility

$$M^2 = \frac{|g|\lambda}{c_h^2},$$

where $c_h^2$ is the sound speed of the heavier fluid. Note that for an incompressible fluid $c_h = \infty$ and $M^2 = 0$. The initial density gradient across the interface is measured by the Atwood number

$$A = \frac{\rho_h - \rho_l}{\rho_h + \rho_l},$$

where $\rho_l$ and $\rho_h$ are the densities of the light and heavy fluids, respectively. All of the simulations used an ambient pressure of 5 at the interface with an isothermal stratification

$$\frac{1}{\rho}\frac{dP}{dz}\bigg|_T = g$$

in density and pressure away from the interface.

The boundary conditions on the sides of the computational domain are reflecting; thus we are simulating RT instability in a closed right circular cylinder. The calculations used the Lax–Wendroff method for the interior solver. Each simulation was performed three times using each of the methods described above for integrating the normal point propagate characteristic equations. For the low compressbility runs shown here there was no appreciable difference in the values of the interfacial growth rate or amplitude due to the characteristic integration method. The simulations shown in the figures below all used the RIEMANN option. Figure 5.1 shows the evolution of the contact fluid interface at times $t = 0, 3, 6, 9$. The grid and physical parameters are $\Delta r = 1/480$, $A = 1/3$, $M^2 = 0.04$. The time step $\Delta t$ was chosen so that the CFL condition was satisfied with CFL factor 0.75. The simulations were run on a 128-node parallel computer. The CPU time was about 160 hours for the runs on the finest grid $\Delta r = 1/480$. At $t = 6$, we begin to observe vortex formation and roll up of the interface due to Kelvin–Helmholtz instability created by velocity shear across the interface.

**5.1. Mesh refinement.** Figure 5.2 shows graphs of both the bubble velocity and interface amplitude for a sequence of mesh sizes dr $= 1/60, 1/120, 1/240, 1/480$ for the same parameter regime described above. We see that the difference between

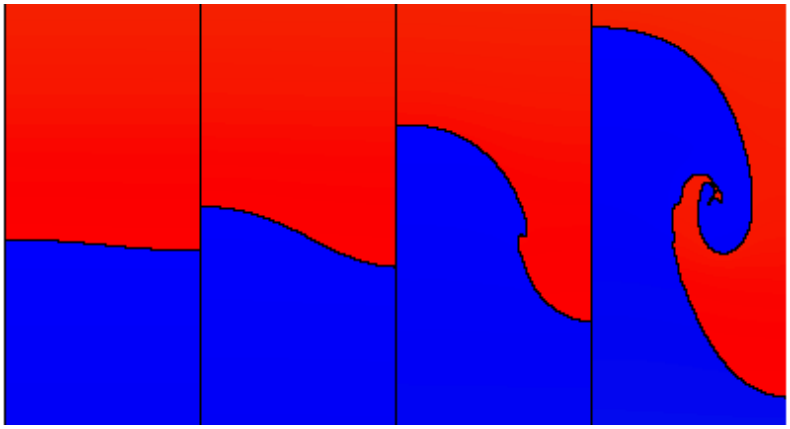


FIG. 5.1. *Front evolution for an RT simulation of an axisymmetric flow with heavy fluid on top of light fluid and gravity pointing down. The computational domain is* $0 \le r \le 0.5$ *and* $0 \le z \le 3$. *The grid size used in the computation is* $\Delta r = \Delta z = 1/480$. *The Atwood ratio of the initial data is* $A = 1/3$, *and the initial compressibility is* $M^2 = 0.04$.

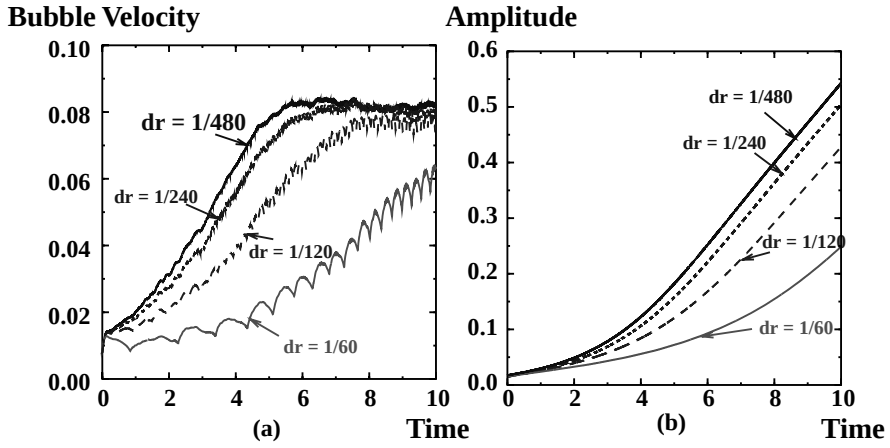



FIG. 5.2. *A convergence test for the simulation of RT instability under mesh refinement:* $\Delta r = \Delta z = 1/60, 1/120, 1/240, 1/480$. *Here* $A = 1/3$. *Left: bubble velocity. Right: amplitude.*

the successive graphs is reduced by half as we move to the next finer grid level in both bubble velocity and amplitude. This is as expected since the code is second order in the interior and first order at the front; hence the motion of the interface is first order accurate. Also we notice that the amplitude is smoother than bubble velocity.

**5.2. Convergence to the plane perturbation case.** As is immediately apparent from the form of the system (4.1) the asymptotic behavior of an axisymmetric flow should approach that of a two dimensional slab-symmetric flow as the minimum radius of the domain approaches infinity. We can validate the axisymmetry implementation of our code by comparing the asymptotic limit $r_0 \to \infty$ with the slab-symmetric plane perturbation case. For this comparison we used physical parameters $A = 1/3$ and $M^2 = 0.04$ with a perturbation wave length of one and a minimum radius $r_0 = 1000$. Both the axisymmetric and slab-symmetric calculations used a square

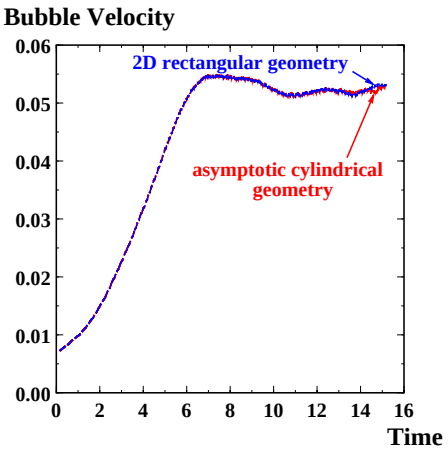



FIG. 5.3. *Comparison of bubble velocities between the asymptotic case $r_0 \to \infty$ and the purely two dimensional planar case. The velocity plots for curved geometry with $r_0 = 1000$ and the one for planar two dimensional geometry are almost identical.*

mesh with $\Delta x = \Delta y = \Delta r = \Delta z = 1/240$. The graphs in Figure 5.3 show bubble velocities computed for both simulations; the results are nearly indistinguishable.

**5.3. The case $r_0 \to 0$.** The geometric singularity at $r = 0$ for axisymmetric flows is problematic for numerical codes. Since the three dimensional form of the code has no such singularity this would suggest that the geometric source terms should have removable singularities at $r = 0$. However, this statement disregards the fact that the axisymmetric formulation forces the flow to remain axisymmetric at the origin, while a three dimensional flow could break axisymmetry. In any case the absolute prohibition on dividing by zero in a numerical code means that the indeterminate form in the geometry source terms must be addressed numerically. One of the easiest ways that people have used to remove the singularity is to simply use a computational domain bounded away from $r = 0$. Thus they compute on a domain with $r_0 > 0$ that is sufficiently small so that its value does not appreciably affect the dynamics of the flow but is still large enough so that no numerical difficulties are cased by dividing by $r_0$. Typical values for $r_0$ are usually on the order of half a mesh block in the $r$ direction or less. Another option is to explicitly enforce the requirement that the geometric source terms have removable singularities at $r = 0$ and that in fact the radial component of velocity should vanish faster than $r$ as $r \to 0$. Indeed, one assumes $v_N/r \to 0$ as $r \to 0$. Thus for small $r$ one simply ignores the geometric terms in the finite difference equations, typically for values of $r$ that are on the order of the machine precision. This approach is implemented in *FronTier*, and we can test its validity by comparing computations with $r_0 = 0$ and relatively small but positive $r_0$.

We conducted a numerical experiment for different values of $\Delta r$ and $r_0/\Delta r$. The corresponding bubble terminal velocities are listed in Table 5.1.

The bubble velocities for $r_0 = 0$, $0.5\Delta r$, $1.5\Delta r$, $3\Delta r$, $6\Delta r$, $12\Delta r$, $24\Delta r$, $\infty$ with fixed $\Delta r = 1/480$ are plotted in Figure 5.4(a). From Figure 5.4(a) and Table 5.1, we see that the terminal velocity converges as $r_0 \to 0$ for the fixed mesh size $\Delta r = 1/480$. From Table 5.1 we also see that for fixed $\Delta r$ the terminal velocity increases as $r_0$ gets smaller. Larger $r_0$ means less curvature, so the bubble evolution occurs in a planar geometry as $r_0 \to \infty$. The bubble is almost fully three dimensional circular

TABLE 5.1
*Bubble terminal velocities for various mesh sizes and values of $r_0$.*

| | $r_0 = 0$ | $r_0 = 0.5\Delta r$ | $r_0 = 3\Delta r$ | $r_0 = 6\Delta r$ | $r_0 = 12\Delta r$ | $r_0 \to \infty$ |
|---|---|---|---|---|---|---|
| $\Delta r = 1/60$ | 0.07 | 0.07 | 0.07 | 0.07 | 0.066 | 0.05 |
| $\Delta r = 1/120$ | 0.078 | 0.078 | 0.078 | 0.077 | 0.073 | 0.052 |
| $\Delta r = 1/240$ | 0.081 | 0.081 | 0.081 | 0.08 | 0.076 | 0.055 |
| $\Delta r = 1/480$ | 0.081 | 0.081 | 0.081 | 0.08 | 0.078 | 0.055 |

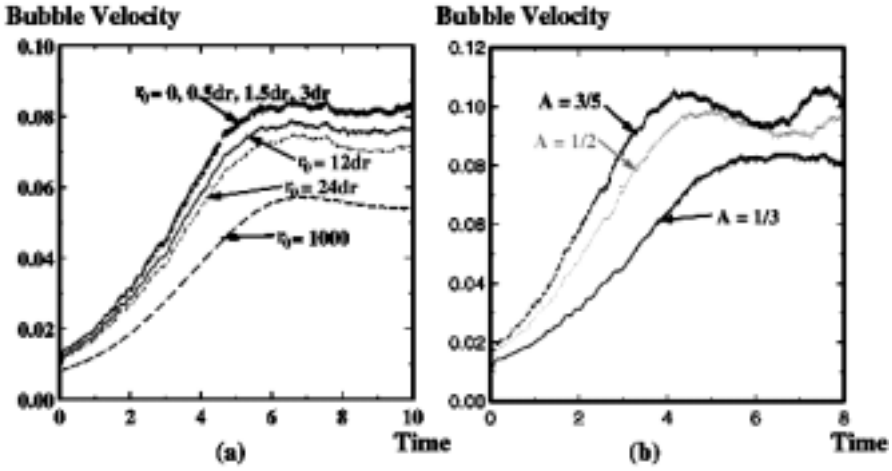



FIG. 5.4. (a) *The convergence of the bubble velocity of RT instability for different $r_0$ at the grid size $\Delta r = 1/480$. The computational domain is $[r_0, r_0 + 0.5] \times [0, 3]$, and $A = 1/3$, $M^2 = 0.04$.* (b) *Comparison of the bubble velocities of RT instability for different Atwood numbers $A = 1/3, 1/2, 3/5$ at the grid $\Delta r = 1/480$. The computational domain is $[0.001, 0.501] \times [0, 3]$, and $M^2 = 0.04$.*

for $r_0$ near zero. The three dimensional geometrical effect increases as $r_0$ decreases. From Figure 5.4(a), we also observe that the rate of approach to the terminal velocity region becomes faster as $r_0$ gets smaller. Finally, the bubble terminal velocities vary modestly with grid size $\Delta r$ for $r_0$ fixed.

We also investigated the effect of the Atwood number on the bubble velocity. For the value $r_0 = 0$ and $\Delta r = 1/480$ we conducted simulations with $A = 1/3$ $M^2 = 0.04$, $A = 1/2$ $M^2 = 0.04$, and $A = 3/5$ $M^2 = 0.04$. The bubble velocity plots are shown in Figure 5.4(b). We see that the bubble terminal velocity and the rate of approach to terminal velocity increases with $A$. We also observe some oscillation in the terminal velocity region about the limiting value. This effect was previously observed by Lin [43].

**5.4. Comparison to experiment and theory.** An experimental and theoretical investigation for a single axisymmetric air bubble rising in water or nitrobenzene in a constant cross section cylindrical tube was considered by Davies and Taylor [12]. Assuming incompressible irrotational flow and an Atwood number of one, they derived an approximate theory that yielded a steady state velocity for the vertex of the bubble as

$$V_B \approx C\sqrt{gR}, \tag{5.1}$$

TABLE 5.2
*Bubble terminal velocities as a function of Atwood number.*

| $A$ | $g$ | $R$ | $M^2$ | $V_B$ | $C$ | $C_A$ | $C'_A$ |
|---|---|---|---|---|---|---|---|
| 1/3 | 0.14 | 1/2 | 0.04 | 0.081 | 0.30 | 0.53 | 0.43 |
| 1/2 | 0.14 | 1/2 | 0.04 | 0.090 | 0.48 | 0.5 | 0.42 |
| 3/5 | 0.14 | 1/2 | 0.04 | 0.095 | 0.36 | 0.46 | 0.41 |

where $R$ is the radius of the tube and $g$ is the acceleration of the gravity. They calculated $C = 0.464$, and their experiments yielded values of $C$ from 0.466 to 0.490.

Subsequently, Layzer [39] considered this same problem. His theory yielded a value of $C = 0.511$. The difference between the Davies–Taylor theory and Layzer's is that the former required that Bernoulli's equation be satisfied on the bubble surface at two distinct points ($r = 0$ and $r = R/2$), while Layzer required that Bernoulli's equation be satisfied in a first order neighborhood of the bubble vertex.

While both Davies–Taylor's formula and Layzer's model assume irrotational incompressible fluid and Atwood number one, they have nevertheless served as useful guides for gaining understanding for more general flows. A generalization of formula (5.1) to Atwood numbers less than one has been proposed by adding an $A$ dependence to $C$ so that $C = C_A\sqrt{A}$ with $C_A$ approximately independent of $A$. Birkoff [5] proposed a further refinement $C = C'_A\sqrt{\frac{2A}{1+A}}$.

Table 5.2 shows a comparison of our numerical simulations with the above three formulas. We see that our numerical coefficients lie approximately within the range reported by Davies–Taylor experimental values and that the value of $C'_A$ is approximately constant in agreement with Birkoff's formula.

The case of two dimensional bubbles is also of interest. This corresponds to a slab-symmetric bubble rising between two parallel walls. Layzer's potential model also applies in this case, and he predicted that formula (5.1) holds for the bubble terminal velocity for two dimensional incompressible flow with Atwood number one if one replaces $R$ by half the perturbation wave length. He obtained $C_{2D} \approx 0.326$. Birkhoff and Carter [7] estimated $C_{2D}$ in the range 0.31–0.35. Some experimental evidence is given by the photographic observations of Duff as reported in [6]. His measured value of $C_{2D} \approx 0.41$ was about 25% greater than the value given by Birkhoff and Carter [7]. Birkhoff and Carter [7] pointed out that this discrepancy might have been largely a side-wall effect. Using a difference-differential equation derived from the free-boundary condition, Garabedian [16] established a lower bound

$$C_{2D} \geq 0.334. \tag{5.2}$$

Our computed value of $C_{2D}^{num}$ for the case $R = 1/2$, $A = 1/3$, $g = 0.14$, $M^2 = 0.04$ is 0.36 as can be calculated from Table 5.1 for the case $r_0 \to \infty$. This value is about 8% higher than Garabedian's lower bound and slightly above the upper limit of the Birkoff and Carter range. Studies in two dimensions [54] indicate that $C_{2D}$ increases with compressibility, so our result for low compressibility flows is generally consistent with the incompressible theory.

**6. Conclusions.** Interface tracking is a method for the resolution of wave fronts in a flow. Here we have provided a general description of the main code modules used in constructing the front tracking code *FronTier*. Specifics, such as the handling of wave interactions or the front motion calculation, may differ between implementations. Other approaches to interface tracking such as level sets and volume of fluids have

also been proposed and share some of the features required by our code, namely some ability to either reconstruct or evaluate local geometry and modified shock capturing methods to account for fronts. Both approaches are compatible with the current code structure in *FronTier*, and hybrid approaches combining features of all three methods are currently being investigated.

In our implementation we have tried to adopt a modular approach to the code structure, carefully separating geometric modules from physics dependencies. This has made it possible to create libraries that can be applied across a range of physical systems including compressible gas dynamics, flow in porous media, and rate dependent elastoplasticity. Current research is also progressing on the incorporation of new physical models such as homogenized models for multiphase mixing.

Our axisymmetric algorithms were validated by comparing the computed RT bubble velocity to experimental and theoretical predictions in both two dimensional and three dimensional axisymmetric cases, and we found good agreement between these values and those computed by *FronTier*.

## REFERENCES


[1] J. Bell, P. Colella, and J. Trangenstein, *Higher order Godunov methods for general systems of hyperbolic conservation laws*, J. Comput. Phys., 82 (1989), pp. 362–397.

[2] D. J. Benson, *An efficient, accurate, simple ALE method for nonlinear finite element programs*, Comput. Methods Appl. Mech. Engrg., 72 (1989), pp. 305–350.

[3] D. J. Benson, *Computational methods in Lagrangian and Eulerian hydrocodes*, Comput. Methods Appl. Mech. Engrg., 99 (1992), pp. 235–394.

[4] D. J. Benson, *Momentum advection on a staggered mesh*, J. Comput. Phys., 100 (1992), pp. 143–162.

[5] G. Birkhoff, *Helmholtz and taylor instability*, in Proceedings of Symposia in Applied Mathematics, AMS, Providence, RI, 1962, pp. 55-76.

[6] G. Birkhoff, *Taylor Instability. Appendices to Report LA*-1862, Tech. Rep. Appendix D of LA-1927, Los Alamos National Laboratory, Los Alamos, NM, 1955.

[7] G. Birkhoff and D. Carter, *Rising plane bubbles*, J. Math. Mech., 6 (1957), pp. 769–779.

[8] I.-L. Chern, J. Glimm, O. McBryan, B. Plohr, and S. Yaniv, *Front tracking for gas dynamics*, J. Comput. Phys., 62 (1986), pp. 83–110.

[9] A. Chorin, *Random choice solutions of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–533.

[10] P. Colella, *A direct Eulerian MUSCL scheme for gas dynamics*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 104–117.

[11] P. Colella and P. Woodward, *The piecewise parabolic method (PPM) for gas-dynamical simulation*, J. Comput. Phys., 54 (1984), pp. 174–201.

[12] R. M. Davies and G. I. Taylor, *The mechanics of large bubbles rising through extended liquids and through liquids in tubes*, Proc. Roy. Soc. London, Ser. A, 200 (1950), pp. 375–390.

[13] J. Donea, S. Giuliani, and J. P. Halleux, *An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions*, Comput. Methods Appl. Mech. Engrg., 33 (1982), pp. 689–723.

[14] J. K. Dukowicz, *A general, non-iterative Riemann solver for Godunov's method*, J. Comput. Phys., 61 (1985), pp. 119–137.

[15] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, J. Comput. Phys., 152 (1999), pp. 457–492.

[16] P. Garabedian, *On steady-state bubbles generated by Taylor instability*, Proc. Roy. Soc. London Ser. A, 241 (1957), pp. 423–431.

[17] C. L. Gardner, J. Glimm, O. McBryan, R. Menikoff, D. Sharp, and Q. Zhang, *The dynamics of bubble growth for Rayleigh-Taylor unstable interfaces*, Phys. Fluids, 31 (1988), pp. 447–465.

[18] J. Glimm, J. Grove, X. L. Li, W. Oh, and D. H. Sharp, *A critical analysis of Rayleigh-Taylor growth rates*, J. Comput. Phys., 169 (2001), pp. 652–677.

[19] J. Glimm, J. W. Grove, X. L. Li, and D. C. Tan, *Robust computational algorithms for dy-*

*namic interface tracking in three dimensions*, SIAM J. Sci. Comput., 21 (2000), pp. 2240–2256.

[20] J. Glimm, J. Grove, X.-L. Li, and N. Zhao, *Simple Front Tracking*, Contemp. Math. 238, G.-Q. Chen and E. DiBebedetto, eds., AMS, Providence, RI, 1999, pp. 133–149.

[21] J. Glimm, J. Grove, B. Lindquist, O. A. McBryan, and G. Tryggvason, *The bifurcation of tracked scalar waves*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 61–79.

[22] J. Glimm, J. W. Grove, X. L. Li, K.-M. Shyue, Y. Zeng, and Q. Zhang, *Three-dimensional front tracking*, SIAM J. Sci. Comput., 19 (1988), pp. 703–727.

[23] J. Glimm, E. Isaacson, D. Marchesin, and O. McBryan, *Front tracking for hyperbolic systems*, Adv. Appl. Math., 2 (1981), pp. 91–119.

[24] J. Glimm, X.-L. Li, R. Menikoff, D. H. Sharp, and Q. Zhang, *A numerical study of bubble interactions in Rayleigh-Taylor instability for compressible fluids*, Phys. Fluids A, 2 (1990), pp. 2046–2054.

[25] J. Glimm, D. Marchesin, and O. McBryan, *Subgrid resolution of fluid discontinuities* II, J. Comput. Phys., 37 (1980), pp. 336–354.

[26] J. Glimm and O. McBryan, *A computational model for interfaces*, Adv. Appl. Math., 6 (1985), pp. 422–435.

[27] J. Glimm and X.-Li, *On the validation of the Sharp-Wheeler bubble merger model from experimental and computational data*, Phys. Fluids, 31 (1988), pp. 2077–2085.

[28] E. Godlewski and P. A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Springer-Verlag, New York, 1991.

[29] J. Grove, *Anomalous waves in shock wave – fluid interface collisions*, in Current Progress in Hyperbolic Systems: Riemann Problems and Computations, Contemp. Math. 100, B. Lindquist, ed., AMS, Providence, RI, 1989, pp. 77–90.

[30] J. Grove, *The interaction of shock waves with fluid interfaces*, Adv. Appl. Math., 10 (1989), pp. 201–227.

[31] J. Grove, *FronTier: A Compressible Hydrodynamics Front Tracking Code; A Short Course in Front Tracking*, LANL Report LA-UR 99-3985, Los Alamos National Laboratory, Los Alamos, NM, 1999.

[32] J. Grove, R. Holmes, D. H. Sharp, Y. Yang, and Q. Zhang, *Quantitative theory of Richtmyer-Meshkov instability*, Phys. Rev. Lett., 71 (1993), pp. 3473–3476.

[33] J. Grove and R. Menikoff, *The anomalous reflection of a shock wave at a material interface*, J. Fluid Mech., 219 (1990), pp. 313–336.

[34] J. W. Grove, *Applications of front tracking to the simulation of shock refractions and unstable mixing*, J. Appl. Numer. Math., 14 (1994), pp. 213–237.

[35] C. W. Hirt, A. A. Amsden, and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253. Reprinted in 135 (1997), pp. 203–216.

[36] R. L. Holmes, B. Fryxell, M. Gittings, J. W. Grove, G. Dimonte, M. Schneider, D. H. Sharp, A. Velikovich, R. P. Weaver, and Q. Zhang, *Richtmyer-Meshkov instability growth: Experiment, simulation, and theory*, J. Fluid Mech., 389 (1999), pp. 55–79.

[37] R. L. Holmes, J. W. Grove, and D. H. Sharp, *Numerical investigation of Richtmyer-Meshkov instability using front tracking*, J. Fluid Mech., 301 (1995), pp. 51–64.

[38] P. Lax and B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math., 13 (1960), pp. 217–237.

[39] D. Layzer, *On the instability of superimposed fluids in a gravitational field*, Astrophys. J., 122 (1955), pp. 1–12.

[40] R. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, Basel, Boston, Berlin, 1992.

[41] X.-L. Li, *A numerical study of* 3*-D bubble merger in the Rayleigh-Taylor instability*, Phys. Fluids, 8 (1996), pp. 322–335.

[42] X.-L. Li, B. X. Jin, and J. Glimm, *Numerical study for the three dimensional Rayleigh-Taylor instability using the TVD/AC scheme and parallel computation*, J. Comput. Phys., 126 (1996), pp. 343–355.

[43] A. Lin, *Nonuniform Approach to Terminal Velocity for Single Mode Rayleigh-Taylor Instability*, Ph.D. thesis, Dept. of Applied Math., SUNY at Stony Brook, Stony Brook, NY, 2000.

[44] L. G. Margolin, *Introduction to "an arbitrary Lagrangian-Eulerian computing method for all flow speeds,"* J. Comput. Phys., 135 (1997), pp. 198–202.

[45] R. Richtmyer and K. Morton, *Difference Methods for Initial Value Problems*, 2nd ed., Interscience, New York, 1967.

[46] J. A. Sethian, *Level Set Methods*, Cambridge University Press, Cambridge, UK, 1996.

[47] D. H. Sharp, *An overview of Rayleigh-Taylor instability*, Phys. D, 12 (1984), pp. 3–18.
[48] G. Strang, *Accurate partial difference methods* II: *Nonlinear problems*, Numer. Math., 6 (1964), pp. 37–46.
[49] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.
[50] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer-Verlag, Berlin, Heidelberg, 1997.
[51] B. van Leer, *Towards the ultimate conservative difference scheme:* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.
[52] Y. Yang, Q. Zhang, and D. H. Sharp, *Small amplitude theory of Richtmyer-Meshkov instability*, Phys. Fluids, 6 (1994), pp. 1856–1873.
[53] Q. Zhang, *Validation of the chaotic mixing renormalization group fixed point*, Phys. Lett. A, 151 (1990), pp. 18–22.
[54] Q. Zhang, *The motion of single bubble or spike in Rayleigh-Taylor unstable interfaces*, Impact Comput. Sci. Engrg., 3 (1991), pp. 277–304.
[55] Q. Zhang and M. J. Graham, *A numerical study of Richtmyer-Meshkov instability driven by cylindrical shocks*, Phys. Fluids, 10 (1998), pp. 974–992.