Computations of Multiphase Flows by a Finite Difference/Front Tracking Method. I. Multi-Fluid Flows

G. Tryggvason,
B. Bunner, O. Ebrat, and W. Tauber
Department of Mechanical Engineering and Applied Mechanics
University of Michigan

Abstract

A numerical method for direct simulations of multiphase flows is presented. The method is based on writing one set of governing equations for the whole computational domain and treating the different phases as one fluid with variable material properties. Interfacial terms are accounted for by adding the appropriate sources as δ-functions at the interface between the phases. The flow is taken to be incompressible. The unsteady Navier-Stokes equations are solved by a conventional finite volume method on a fixed, structured grid and the interface, or front, tracked explicitly by connected marker points. Interfacial source terms such as surface tension are computed on the front and transfered to the fixed grid. Advection of fluid properties such as density is done by following the motion of the front. The method has been implemented for fully three-dimensional flow as well as two-dimensional and axisymmetric ones. This approach has been shown to be reasonably accurate, versatile, and robust.

In part I of these lecture notes, the method is described for the flow of two or more isothermal phases. The representation of the moving interface, and its dynamic restructuring, as well as the transfer of information between the moving front and the fixed grid is discussed. Various validation studies are shown and the relation of the method to other computational techniques discussed. A recent parallelization of the method is also presented.

In part II, several applications are shown. Those include detailed studies of drop collisions, motion of single bubbles and drops in shear flows, and the collective interaction of many bubbles and drops.

Part III describes extensions of the methodology to problems with more complicated physics such as variable surface tension, solidification, and boiling.

1 Introduction

Although efforts to compute the flow of multiphase flows are as old as computational fluid dynamics, the difficulty of solving the full Navier-Stokes equations in the presence of a deforming phase boundary has proven to be considerable. Progress was therefore slow and simulations of finite Reynolds number flows were, for a long time, limited to very simple problems. In the last few years, however, major progress has been achieved. Here, we describe a method that has been particularly successful for a wide range of multifluid and multiphase flows. Before we start discussing our technique, we will briefly review other techniques.

The oldest and still the most popular approach to compute multifluid and multiphase flows is to capture the interface directly on a regular, stationary grid. The MAC method, where marker particles are advected for each fluid, and the VOF method, where a marker function is advected, are the best known examples. In their original implementation, stress conditions at the fluid interface were only satisfied in a very approximate way, but a number of recent developments, including a technique to include surface tension developed by Brackbill, Kothe, and Zemach (1992), and the use of "level sets" (see, e.g. Sussman, Smereka, and Osher, 1994) to mark the fluid interface has increased the accuracy and therefore the applicability of this approach. A recent application of the VOF method can be found, for example, in Richards, Lenhoff, and Beris (1994) who simulate the axisymmetric breakup of a jet of one liquid in another liquid and in Lafaurie *et al* (1994) who examined the three-dimensional of collision of two drops. An application of the level set method to the motion of bubbles can be found in Sussman, Smereka, and Osher (1994). Recent additions to the collection of methods that capture fluid interfaces on a fixed grid include the CIP method of Yabe (1997) and the phase field method of Jacqmin (1996). Traditionally, the main difficulty in the use of these methods has been the maintenance of a sharp boundary between the different fluids and the computation of the surface tension.

The second class, and the one that offers potentially highest accuracy, uses separate, boundary fitted grids for each phase. The steady rise of buoyant, deformable, axisymmetric bubbles were simulated by Ryskin and Leal (1984) using this method in a landmark paper that had a major impact on subsequent development. Dandy and Leal (1989) subsequently examined the steady motion of deformable axisymmetric drops and Kang and Leal (1987) extended the methodology to axisymmetric, unsteady motion. Several other two-dimensional and axisymmetric computations of the unsteady motion of one or two bubbles or drops have appeared recently. This method is best suited for relatively simple geometries, and applications to complex fully three-dimensional problems with unsteady deforming phase boundaries are very rare. The simulation of a single unsteady three-dimensional bubble by Takagi and Matsumoto (1996) is, perhaps, the most impressive example.

The third class, is Lagrangian methods where the grid follows the fluid. Recent examples include two dimensional computations of the breakup of a drop by Oran and Boris (1987); examination of the initial deformation of a buoyant bubble by Shopov et al (1990); simulations of the unsteady two-dimensional motion of several particles by Feng, Hu, and Joseph (1994); and axisymmetric computations of the collision of a single drop with a wall by Fukai et al. (1994). While this appears to be a fairly complex approach, Tezduyar (1998) and Hu (1998) have recently produced very impressive results for the three-dimensional unsteady motion of many spherical particles.

The fourth category is front tracking where a separate front marks the interface but a fixed grid, only modified near the front to make a grid line follow the interface, is used for the fluid within each phase. This technique has been extensively developed by Glimm and collaborators (see, e.g. Glimm 1991). In addition to these methods that are, in principle, applicable to the full Navier Stokes equations, boundary integral methods have been used for both inviscid and Stokes flows. For three-dimensional applications see Kennedy, Posrikidis, and Skalak (1994), Manga and Stone (1993), and Lowenberg and Hinch (1996) for drops in a Stokes flow, and Chahine (1994) for inviscid bubbles.

The method described here is properly described as a hybrid between a front capturing and a front tracking technique. A stationary regular grid is used for the fluid flow, but the interface is tracked by a separate grid of lower dimension. This grid is usually called the front. However, unlike front tracking methods where each phase is treated separately, we follow front capturing methods and treat all phases by a single set of governing equations whole flow field. Although the idea of using only one set of equations for many co-flowing phases is an old one, the method described here is a direct descendant of a Vortex-In-Cell technique for invisicd multifluid flows described in Tryggvason and Aref (1983) and Tryggvason (1988) and the immersed boundary method of Peskin (1977) developed to put moving boundary into finite Reynold's number homogeneous fluids. The original version of the method and a few sample computations are presented by Unverdi and Tryggvason (1992). Several modifications and improvements are described here.

The technique has been used to examine a number of multiphase flows problems and several examples will be shown in part II of theses lecture notes. For drops we have looked at the collision of two equal sized drops, both by axisymmetric computations for head on collisions (Nobari, Jan, and Tryggvason, 1996) as well as by fully three-dimensional simulations for off-axis collisions (Nobari and Tryggvason, 1996). Primary focus was on when the drops broke up again after initial coalescence. Detailed comparison with experimental data (Quian, Tryggvason, and Law, 1998) have shown excellent agreement. Recently, we have examined the breakup of accelerated drops (Han, 1998). The computations show both "bag" and "shear" breakup and have helped clarify the mechanism when the density difference between the drops and the surrounding fluid is small. Other axisymmetric computations of relatively simple problems include the coalescence of initially stationary drops of a different size (Nobari, 1993) and collisions of vortex rings with a fluid interfaces (Bernal *et al*, 1994). Two-dimensional simulations have been used to examine the dissipation of free surface waves (Yang and Tryggvason, 1998) and the nonlinear evolution of the Kelvin-Helmholtz instability of stratified flows (Unverdi, Tauber, and Tryggvason, 1998).

Several of our investigations have focused on dispersed flows of bubbles and drops surrounded by another continuous fluid. For bubbles we have done a large number of two and three-dimensional computations for periodic domains.The simulations have helped explain how bubbles interact when they move freely, and the importance of accounting for such interactions when modeling multi bubble flows. See Jan (1994), Esmaeeli and Tryggvason (1996,1997, 1998), and Bunner and Tryggvason (1997) for details. Two and three-dimensional simulations of drops in pressure driven channel flows show clustering and shear thinning (Mortazavi, 1995). In addition to simulations of a large number of bubbles and drops, we have examined the motion of a single bubble in a shear flow (Ervin and Tryggvason, 1997) and a single drop in a pressure driven channel flow (Mortazavi, 1995). The results show that a relatively small deformation of a bubble changes not only the value of the lift coefficient, but also its sign. While this has been observed in recent experimental work, our calculations have provided both an explanation and quantitative data. The results for a single drop have shown that for modest Reynolds numbers the drops generally move to a position about half way between the center of the channel and its wall. This results are in excellent agreement with experimental observations.

In addition to these problems where two isothermal fluids have moved together, we have extended the methodology to handle situations where other physical effects must be accounted for. Real bubbles, particularly in water, are rarely clean, and we have investigated the effect of contaminants on deformable bubbles at finite Reynolds numbers (Jan, 1994). While the presence of contaminants usually slows the bubbles down, we have also found that contamination generally reduces deformations and, for very deformable bubbles at high Reynolds number, the reduction in pressure drag can offset the increased frictional drag. We have also done several computations of the migration of drops and bubbles due to a temperature dependent surface tension. Those have shown, for example, that for a fairly wide range of parameters the drops line up in rows perpendicular to the temperature gradient (Nas, 1995; Nas and Tryggvason, 1993). In Yu, Ceccio, and Tryggvason (1995) we used a simple model of cavitation, where we ignore all thermal effects and simply enforce a constant pressure inside the bubble, to examine the effect of fluid shear on the growth and collapse of vapor bubbles. A more realistic model of phase changes is presented in Juric and Tryggvason (1996), where solidification of a pure material is simulated by writing one energy equation for both phases. Heat release due to phase change is included as a source term at the phase boundary. For full multiphase flow problems, the energy equation must be coupled with the Navier-Stokes equations. Juric and Tryggvason (1997) present a method to do this and show simulations of the growth of two-dimensional vapor layer near a hot wall. The extensions of the basic methodology to problems where additional physics must be accounted for is discussed in part III of these lecture notes.

2 Description of the Method

The key to our method as well as several other recently proposed methods to simulate multiphase flow is the use of a single set of conservation equations for the whole flow field. The equations must therefore account for both the differences in the material properties of the different phases as well as surface tension at the phase boundary. While this idea goes back to the early days of CFD at Los Alamos, it has recently been revived very successfully by a number of researchers. Here we will write down the general equations for the case when all phases are incompressible and then examine in detail one particular implementation.

2.1 "One-Field" formulation of the Navier-Stokes equations for multiphase flows.

The representation of the simultaneous flow of different immiscible fluids with one set of conservation laws requires us to account for interfacial phenomena such as surface tension by adding the appropriate interface terms to the governing equations. Since these terms are concentrated at the boundary between the different fluids, they are represented by δ-functions and when the equations are discretized, the δ-functions must be approximated along with the rest of the equations. Since the material properties and the flow field are, in general, discontinuous across the interface, the differential form of the governing equations must be interpreted as a weak form, satisfied only in an integral sense, or all variables must be interpreted in terms of generalized functions. We take the latter approach here.

2.1.1 Preliminaries

Before we write down the equations governing multiphase flow it is useful to discuss a few elementary aspect of the representation of a discontinuous function by generalized functions.

The various fluids can be identified by a step (Heaviside) function H which is 1 where one particular fluid is and 0 elsewhere. The interface itself is marked by a non-zero value of the gradient of the step function and to relate the gradient to the δ-function marking the interface, it most convenient to express H in terms of an integral over the product of one-dimensional δ-functions:

$$H(x,y,t) = \int_{A(t)} \delta(x-x')\delta(y-y')da'. \tag{1}$$

The integral is over an area A bounded by a contour S. H is obviously 1 if (x,y) is within S and 0 otherwise. Here, we have assumed a two-dimensional flow, the extension to three-dimensions is obvious. To find the gradient of H we note first that since the gradient is with respect to the unprimed variables, the gradient operator can be put under the integral sign. Since the δ-functions are anti-symmetric with respect to the primed and unprimed variables, the gradient with respect to the unprimed variables can be replaced by the gradient with respect to the primed variables. The resulting area (or volume in three-dimensions) integral can the be transformed into a line (surface) integral by a variation of the divergence theorem for gradients. Symbolically:

$$\nabla H = \int_A \nabla\left[\delta(x-x')\delta(y-y')\right]da' = -\int_A \nabla'\left[\delta(x-x')\delta(y-y')\right]da' = \tag{2}$$

$$-\oint_S \delta(x-x')\delta(y-y')\mathbf{n}ds'$$

where the prime on the gradient symbol denotes the gradient with respect to the primed variables. Although we have assumed that the area occupied by the marked fluid is finite so that S is a closed contour, the contribution of most of the integral is zero so we can replace it by one over a part of the contour and drop the circle on the integral:

$$\nabla H = -\int_S \delta(x-x')\delta(y-y')\mathbf{n}ds' \tag{3}$$

The density as well as any other material property, can be written in terms of the constant densities on either side of the interface and the Heaviside function:

$$\rho(x,y,t) = \rho_i H(x,y,t) + \rho_o(1 - H(x,y,t)). \tag{4}$$

Here, ρ_i is the density where $H = 1$ and ρ_o is the density where $H = 0$. The gradient of the density is given by

$$\nabla\rho = \rho_i \nabla H - \rho_o \nabla H = (\rho_i - \rho_o)\nabla H = \Delta\rho \int \delta(x - x')\delta(y - y')\mathbf{n}ds' \tag{5}$$

where we have put $\Delta\rho = \rho_o - \rho_i$.

The time derivative of the density is:

$$\frac{\partial\rho}{\partial t} = \rho_i \frac{\partial H}{\partial t} - \rho_o \frac{\partial H}{\partial t} = (\rho_i - \rho_o)\frac{\partial H}{\partial t}. \tag{6}$$

To find the time derivative of H, note that since H is either 1 or 0, its evolution is governed by

$$\frac{\partial H}{\partial t} + \mathbf{V} \cdot \nabla H = 0 \tag{7}$$

where $\mathbf{V}$ is a smooth velocity field that matches the interface velocity at the interface (if there is no expansion at the interface and the fluids are incompressible, $\mathbf{V}$ could be the fluid velocity). The velocity field is smooth and a function of the unprimed variables, so we can bring it under the integral sign, resulting in

$$\frac{\partial H}{\partial t} = -\mathbf{V} \cdot \nabla H = -\oint_S \delta(x - x')\delta(y - y')V_n ds' \tag{8}$$

where $V_n = \mathbf{V} \cdot \mathbf{n}$. Since the density jump is constant, this leads to:

$$\frac{\partial\rho}{\partial t} = \Delta\rho \oint_S \delta(x - x')\delta(y - y')V_n ds'. \tag{9}$$

This equation will be used in Part III of these lecture notes when we introduce a method for flows with phase changes.

2.1.2 Conservation equations

The fluid motion is assumed to be governed by the Navier-Stokes equations. For variable viscosity, the full deformation rate tensor must be included and we will use the conservative form for the advection terms:

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla \cdot \rho\mathbf{u}\mathbf{u} = -\nabla P + \rho\mathbf{f} + \nabla \cdot \mu(\nabla\mathbf{u} + \nabla^T\mathbf{u}) + \int \sigma\kappa'\mathbf{n}'\delta^\beta(\mathbf{x} - \mathbf{x}')ds' \tag{10}$$

This equation is valid for the whole flow field, even if the density field, ρ, and the viscosity field, μ, change discontinuously. Here $\mathbf{u}$ is the velocity field, P is the pressure, and $\mathbf{f}$ is a body force. Surface forces are added at the interface. δ^β is a two or three-dimensional δ-function constructed by repeated multiplication of one-dimensional δ-functions. The dimension is denoted by $\beta = 2$ or 3. κ is the curvature for two-dimensional flow and twice the mean curvature for three-dimensional flows. $\mathbf{n}$ is a unit vector normal to the front. $\mathbf{x}$ is the point at which the equation is evaluated and $\mathbf{x}'$ is a point on the front. Formally, the integral is over the entire front, thereby adding the delta functions together to create a force that is concentrated at the interface, but smooth along the interface. Since the δ-function has a finite support, integrating over the entire front for every point in the flow is neither practical nor necessary. Just as the advection terms can be written in many different forms, it is possible to rewrite the surface tension term in other ways. In the numerical implementation we use a different, but equivalent expression for the surface tension as discussed shortly.

Mass conservation is given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0. \tag{11}$$

In almost all the problems that we have considered so far, the fluids are taken to be incompressible so that the density of a fluid particle remains constant:

$$\frac{D\rho}{Dt} = 0. \tag{12}$$

This reduces the mass conservation equation to

$$\nabla \cdot \mathbf{u} = 0. \tag{13}$$

Usually, we also take the viscosity in each fluid to be constant as well:

$$\frac{D\mu}{Dt} = 0. \tag{14}$$

2.1.3 Interfacial conditions

While the formulation of the Navier-Stokes equation presented in the last section has been around for a long time, it is not as familiar as writing down separate equations and matching them at the phase boundary. It is therefore useful to demonstrate explicitly that this formulation implicitly contains the same conditions at the interface as found in standard references. To do so, we move to a frame moving with the interface and integrate the equations over a small volume enclosing the interface. As we shrink the volume to zero, most of the terms go to zero and only gradient terms survive. Integrating the normal component yields

$$\left[-P + \mu(\nabla \mathbf{u} + \nabla^T \mathbf{u})\right] \mathbf{n} = \sigma \kappa \mathbf{n} \tag{15}$$

where the brackets denote the jump across the interface. This is, of course, the usual statement of continuity of stresses at a fluid boundary, showing that the normal stresses are balanced by surface tension. Integrating the tangential component shows that the tangential stresses are continuous and integrating the mass conservation equation (13) across the interface shows that the normal velocities are also continuous.

If the governing equations are solved separately in the region occupied by the different phases, those conditions along with the continuity of the velocities must be used to match the velocities and the pressures at the interface.

2.2 Numerical Implementation

The formulation described above allows multiphase flow to be treated along the lines usually used for homogeneous flows. Any standard algorithm based on fixed grids can, in general, be used to integrate the Navier-Stokes equations in time. If the density differences are small so the Boussinesq approximation can be used and the viscosities of all the fluids are the same it is even possible to use the streamfunction vorticity formulation (Tryggvason and Unverdi, 1990). In general, however, it is easier to work with the primitive form of the Navier-Stokes equations.

To carry out the actual computation it is necessary to determine:

- how the density (and viscosity) is advected,
- how surface tension is computed,
- how the velocity field is integrated in time,
- how the boundary between the fluids is advected,
- how the advection and the viscous terms are discretized,
- how the pressure equation is solved.

The first two items are what distinguish the various multiphase flow methods based on the weak formulation from each other. In the Volume-Of-Fluid (VOF), the level set, and the Cubic Interpolation (CIP) methods a marker function is used to mark the regions occupied by the various phases. In our method, we explicitly track the phase boundary by connected marker points, forming a "front." Knowing the location of the front allows us to set the density (and other material properties) and to compute the surface tension. Using a front does, however, bring in a number of additional issues. Those are:

- how the front is represented (data structure),
- how information is transferred between the front and the fixed grid,
- how the front is advanced in time,
- how front resolution is maintained as the front deforms,
- how the density and other material properties are determined from the location of the front,
- how surface tension is computed,
- how topology changes are accomplished.

In this section, we will describe how these various tasks can be accomplished. While we focus on one particular implementation we will give references to alternate possibilities (many of which we have also explored). Figure 1 summarizes the approach: A fixed grid is used for the conservation equations but a moving grid of lower dimension marks the boundary between the various phases.

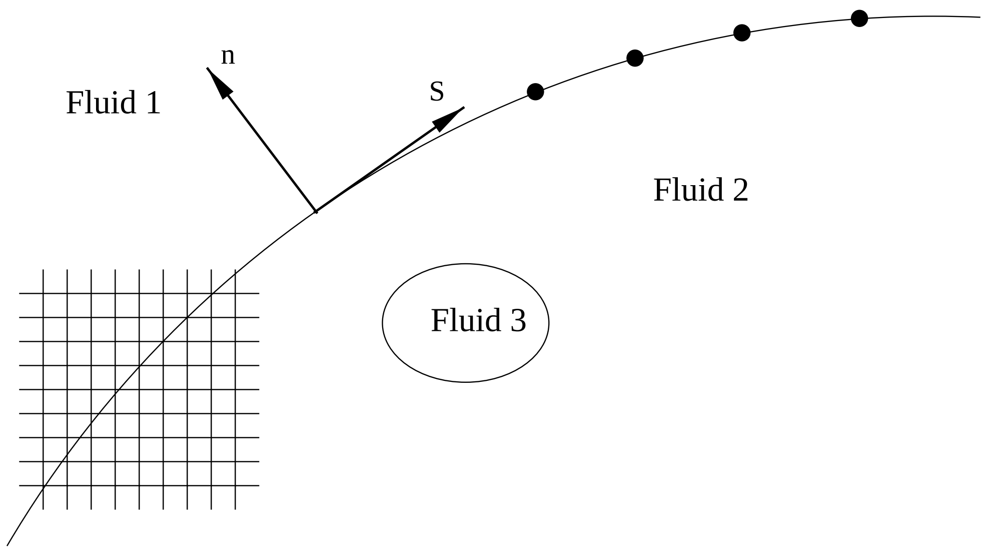

Figure 1: A flow field containing more than one phase. The governing equations are solved on a fixed grid but the phase boundary is represented by a moving "front."

2.2.1 A Projection method for the integration of the Navier-Stokes equations

The equations presented in the last section are usually solved by some form of projection method. Although most of our studies employ a second order time integration technique, we will outline here a very conventional first order scheme: First the density can be updated using the velocity at the current time. We will denote that by:

$$\rho^{n+1} = f(\rho^n, \mathbf{u}^n, \Delta t). \tag{16}$$

This is accomplished by first moving the front and then constructing a grid-density field to match the location of the front. Both these operations will be described in later sections and this equation is merely a symbolic way of expressing what is done. The advection of the density field is one of the critical steps in simulations of multiphase flows. Here, n denotes the old time level and $n+1$ the new one. Once the density has been updated, the velocity field can be computed. The standard way is to split the update into two parts: The first is a projection step where the effects of pressure are ignored:

$$\frac{\rho^{n+1}\mathbf{u}^* - \rho^n \mathbf{u}^n}{\Delta t} = -\nabla_h \cdot \rho^n \mathbf{u}^n \mathbf{u}^n + \nabla_h \cdot \mu^n (\nabla_h \mathbf{u}^n + \nabla_h^T \mathbf{u}^n) + F_s \tag{17}$$

and then a correction step, where the pressure gradient is added:

$$\frac{\rho^{n+1}\mathbf{u}^{n+1} - \rho^{n+1}\mathbf{u}^*}{\Delta t} = -\nabla_h P. \tag{18}$$

The pressure is determined such that the velocity at the new time step is divergence free:

$$\nabla_h \cdot \mathbf{u}^{n+1} = 0. \tag{19}$$

Here we integrate the momentum equations in the conservative form. If the non-conservative form is used, all densities are evaluated at time n and the density can be

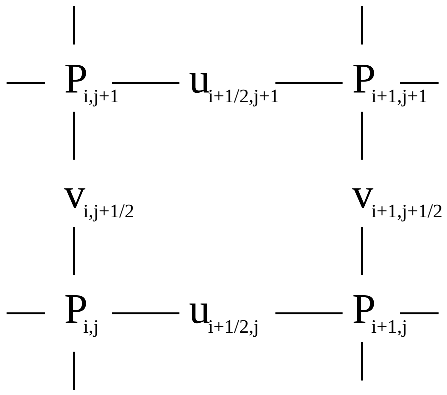

Figure 2: The notation used for a standard staggered MAC mesh.

advected at the end of the step. That is presumably also possible by replacing $\rho^{n+1}\mathbf{u}^*$ by $\rho^n\mathbf{u}^*$. To find the pressure, we use equation (19) to eliminate $\mathbf{u}^{n+1}$ from equation (18), resulting in

$$\nabla_h \frac{1}{\rho^{n+1}} \cdot \nabla_h P = \frac{1}{\Delta t} \nabla_h \cdot \mathbf{u}^* \tag{20}$$

Since the density is variable, this equation can not be solved by traditional fast Poisson solvers designed for separable elliptic equations. The solution of the pressure equation will be discussed in a separate section.

Since we advect the interface at the beginning of the time step, we have the option of computing the surface tension term, F_s, either at the old configuration or the new one. While most investigators using VOF and similar methods appear to have opted to use a completely explicit treatment of the surface tension, we generally find that the implicit treatment is more robust. In actual computations it is, of course, also possible to find those forces using both the new and the old position of the interface in which case the resulting approximation is second order in time.

To compute the momentum advection, the pressure term, and the viscous forces, any number of standard discretization schemes can be used. In most of our computations we use a fixed, regular, staggered MAC grid and discretize the momentum equations using a conservative, second order centered difference scheme for the spatial variables and an explicit second order time integration method. Since we will be making a reference to the layout of the grid, we show the standard notation for the MAC mesh in Figure 2. In the original MAC method, centered differencing was used for all spatial variables, using simple averaging for points where the variables are not defined, and the time integration was done by the simple explicit first order projection method described above. Later implementations, including the VOF method, used first order upwind or the so-called donor-cell method for the advection terms. Recent authors have taken advantage of the various high order upwind schemes developed for compressible flows. Sussman *et al* (1994) used the ENO upwind scheme in their level set method and Pilliod and Puckett (1997) used an unsplit Godunov method, for example. We have used central differences in most of our work to obtain the highest possible accuracy. We have been looking at problems where we are interested in fully resolving the flow and central differences are generally

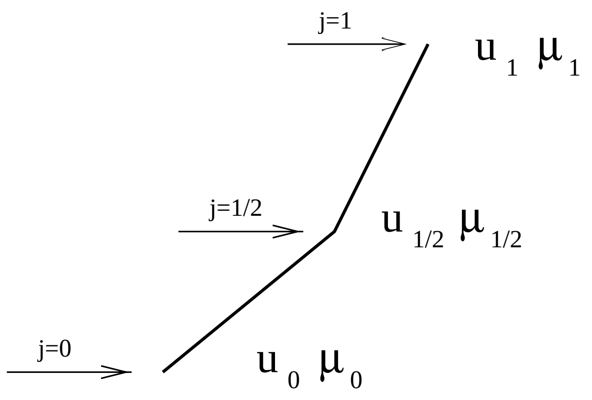

Figure 3: Continuity of the viscous fluxes on a staggered grid

more accurate than upwind methods. For flows where we are more interested in robustness and where we are willing to accept poorly resolved boundary layers, upwind schemes are, of course, preferable.

For the viscous terms we use standard second order centered differences with simple averages for viscosity at points where it is not defined. This is not, however, the only possibility and we have also experimented with using the geometric average as suggested by Patankar several years ago (see Patankar, 1980, for an accessible discussion). Since the use of this method is not as common as one might expect, we discuss it briefly here. Assume that the boundary between a viscous fuid and a less viscous one falls exactly between two grid points (Figure 3). The velocity gradient there will generally be different at $j = 0$ and $j = 1$ since the viscosities are different. The viscous flux, $F_\mu = \mu(\partial u/\partial y)$ is constant, however. The viscosity at node $j = 0$ is different than at node $j = 1$ and we take the viscosity to be μ_0 for $j < 1/2$ and μ_1 for $j > 1/2$. If we knew the velocity at $j = 1/2$, call it $u_{1/2}$, then we must have $\mu_0(u_{1/2} - u_0) = \mu_1(u_1 - u_{1/2})$ since the viscous flux is constant. This should also be equal to $\mu_{1/2}(u_1 - u_0)$ if $\mu_{1/2}$ is correctly defined. Obviously, taking

$$\frac{1}{\mu_{1/2}} = \left(\frac{1}{\mu_0} + \frac{1}{\mu_1}\right) \tag{21}$$

satisfies this requirement. In higher dimensions the formula is extended in an obvious way. While we have experimented with this expression, we have conducted most of our calculations using a simple average. The reason is that the error, when the simple averaging is used, is mostly on the less viscous side of the interface and if we are simulating flows where the less viscous fluid is expected to have a small effect, such as for bubbles or drops in free motion, this is exactly where we prefer the error to be. In other cases, such as when viscous drops are suspended in an other liquid, the accurate resolution of the motion in the less viscous fluid is critical. For a recent use of Patankar's formula, see Coward *et al* (1997).

To achieve a second order accuracy in time, we have used either an Adams-Bashford integration scheme or a simple predictor-corrector scheme where the first order solution at $n+1$ serves as a predictor that is then correct by a trapezoidal rule. The latter method is particularly simple to implement since we can simply take two first order Euler steps

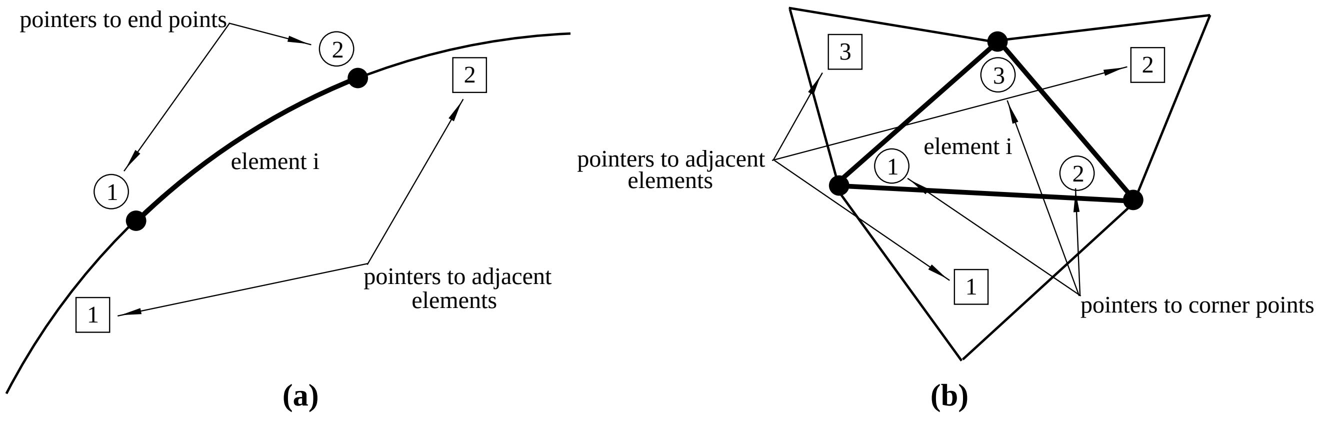
pointers to end points
2
2
1
element i
1
pointers to adjacent elements
1
(a)
3
2
3
pointers to adjacent elements
element i
1
2
1
pointers to corner points
(b)

Figure 4: Structure of the front. (a) Two-dimension. (b) Three-dimension

and then average the final solution and the one at the beginning of the time step.

2.2.2 The structure of the front

For most of our simulations we use a front structure that consists of points connected by elements. Both the points and the elements (the front objects) are stored in linked lists that contain a pointer to the previous object and the next object in the list. The order in the list is completely arbitrary and has no connection to the actual order on the interface. The use of a linked list makes the addition and removal of objects particularly simple. For each point, the only information stored is its coordinates. The elements, on the other hand, contain most of the front information. Each element knows about the points that it is connected to; the elements that are connected to the same endpoints; the surface tension coefficient; the jump in density across the element; and any other quantities that are needed for a particular simulation. The elements are given a direction and for a given front all elements must have the same direction. Figure 4(a) shows the key variables that are stored for a two-dimensional front.

Three-dimensional fronts are build in the same way, except that three points are now connected by a triangular element. The points, again, only know about their coordinates but the elements know about their corner points and the elements that share their edges. Each element has an "outside" and an "inside" and all elements on a given front must be oriented in the same way. The corners of each element and the edges are numbered counter-clockwise when viewing the element from the "outside." Figure 4(b) shows the key variables that are stored for a three-dimensional front.

Since our implementation of the method is in FORTRAN, linked lists are constucted by using several arrays. For each front, an array is set aside for each variable and one object is denoted as the first object in the front. The total number of objects in the front is stored and any operation involving the front is done by starting with the first object and then using its pointer to the next object in the linked list to move to the next object and so on, until all objects have been visited. The unused objects in the array are linked and a pointer to the first unused object is also stored. This makes it relatively simple to add and delete objects.

When information is transferred between the front and the fixed grid it is important to always go from the front to the grid and not the other way around. Since the fixed grid is structured and regular, it is very simple to determine the point on the fixed grid that is closest to a given front position. If we denote the total number of grid points in one direction by nx, the total length by L_x, and we assume that $i = 0$ corresponds to $x = 0$, then the grid point to the left is given by:

$$i = \mathrm{int}(x \cdot nx/L_x) \tag{22}$$

in FORTRAN. If the left hand side of the grid is denoted by $i = 1$, instead of 0, or if the grids are staggered, small modifications are obviously needed. Finding the front point closest to a given grid point is a much more complex operation and we can easily avoid it completely.

In many cases we wish to simulate periodic domains where the front can move out of the domain on one side and reappear in through the other side. This can be done in a very simple way by recognizing that there is no need for the front to occupy the same period as the fixed grid. All that is needed is to correctly identify the grid point that corresponds to a given front position. A slight modification of the operation above accomplishes this:

$$i = \mathrm{int}\left(\mathrm{amod}(x, L_x) \cdot nx/L_x\right) \tag{23}$$

For closed fronts, such as those representing the surface of a bubble or a drop, nothing else needs to be changed. For periodic fronts the end point in one period is connected to the first point in the next period, but only one period is actually computed. When computing the length of such elements, or a line is fitted through the end points, it is therefore necessary correct for the positions of the points.

If a front intersects a boundary, we usually put a point at the wall and a ghost point outside the wall. The ghost point is connected to the point on the wall by a ghost element. These ghost objects are not included in the front that describes the phase boundary, but the front element connected to the wall point treats the ghost element as its neighbor. The ghost objects form a (usually small) linked list that allows us to access them in the same way as the regular front elements. The position of the ghost point is adjusted in such a way that the front tangent at the wall point has the desired value. For a full slip wall or symmetry boundary we usually assume that the front tangent is normal to the boundary. We have not simulated any problems that involve a moving contact line, but in that case the angle between the front and the wall would usually be a function of the velocity of that point.

The strategy for identifying the fixed grid point closest to a given front point works also on irregular grids, as long as they are logically rectangular and can be mapped into a rectangle. In those cases we simply store the mapped coordinate of each front point and use those when we need to communicate between the fixed grid and the moving front.

2.2.3 Restructuring the front

As the front moves, it deforms and usually some parts become crowded with front elements while the resolution of other parts becomes inadequate. To maintain accuracy, additional

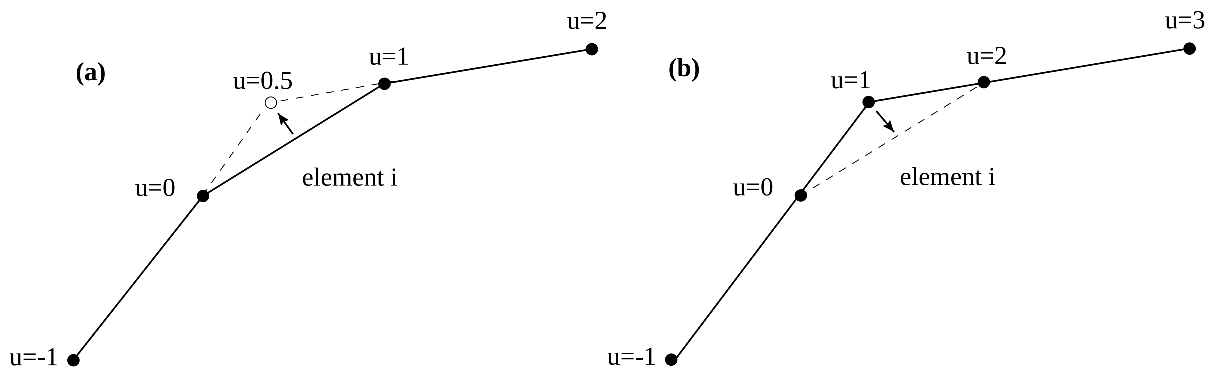

Figure 5: Adding and deleting front elements in two-dimensions.

elements must either be added when the separation of the points becomes too large or the points must be redistributed to maintain adequate resolution. It is generally also desirable to remove small elements. In addition to reducing the total number of elements used to represent the front, element removal usually also prevents the formation of "wiggles" much smaller than the grid size.

While the restructuring of the front makes codes that use explicit tracking more complex than front capturing codes, many of the necessary operations can be made relatively straight forward by the use of a suitable data structure. This is, in particular, true for two-dimensional flows. In some of our early computations, we adjusted the position of all the interface points either at every time step or at every few time step to maintain nearly uniform spacing. Currently we do not do this but add and remove points where needed.

Figure 5 shows the restructuring operations schematically for a two-dimensional front. In (a) we split a large element by insert a point and in (b) we delete an element. Although we sometimes put a new point simply at the mid point between the old end-points of an element, using linear interpolation, we usually account for the curvature of the front by using a higher order interpolation. This is particularly important when surface tension is large and non-smooth parts of the front lead to large pressure fluctuations. We have found that simple Legendre interpolation works well. By numbering the points as shown in the figure, any point is given by:

$$p(u) = \frac{1}{3}(1+u)\left\{\frac{1}{2}u[(2-u)p_1 + (u-1)p_2] + \frac{1}{2}(2-u)[(1-u)p_0 + up_1]\right\} + \tag{24}$$
$$\frac{1}{3}(2-u)\left\{\frac{1}{2}(1+u)[(1-u)p_0 + up_{-1}] + \frac{1}{2}(1-u)[(1+u)p_0 - up_{-1}]\right\}$$

where p is either the x or the y coordinate and u is an interpolation parameter that takes the values shown in the figure. When an element is split by adding a new point, it is simply inserted at $u = 0.5$, resulting in

$$p_{new} = \frac{1}{16}(-p_{-1} + 9(p_0 + p_1) - p_2) \tag{25}$$

When an element is deleted, we delete one of its end points and usually move the other end point to the position given by equation (25). The connectivity of the elements and the points is adjusted and the deleted points and elements removed from the list.

For three-dimensional flows, these operations are more complicated. Not only are there several different ways to add and delete points, but other aspects of the front, such as the shape and the connectivity of the elements must also be considered. The restructuring of the surface grid can be accomplished by adding points and removing points. In some cases, reconnecting the points to define "better shaped" elements is also necessary.

Adding and deleting elements can be done in a variety of ways. Figure 6 shows the strategy that we usually adopt. If an element is too large, we add elements by splitting its longest edge into two and creating two new elements. Similarly, elements are deleted two at a time by collapsing the shortest edge into a point. Sometimes we also reconnect the points by swapping edges to make the elements better shaped.

To interpolate new points, we use barycentric coordinates (u, v, w) defined in figure 7. Notice that the coordinates are not independent and we must have

$$u + v + w = 1 \tag{26}$$

Any interpolated quantity is given by:

$$\begin{aligned} p(u,v,w) = {} & \frac{1}{2}(1-u)[-up_5 + (1-v)p_3 + (1-w)p_2] \\ & +\frac{1}{2}(1-v)[(1-u)p_3 - vp_6 + (1-w)p_1] \\ & +\frac{1}{2}(1-w)[(1-u)p_2 + (1-v)p_1 - wp_4] \end{aligned} \tag{27}$$

The centroid is at approximately $u = v = w = 1/3$, and the point half way between corner points 1 and 2 is given by $u = v = 1/2$ and $w = 0$ or

$$p(\frac{1}{2}, \frac{1}{2}, 0) = \frac{1}{2}(p_1 + p_2) + \frac{1}{4}p_3 - \frac{1}{8}(p_5 + p_6) \tag{28}$$

To determine when it is necessary to add or delete an element, we usually define a minimum and a maximum element size and take action if the size of an element exceeds these limits. These limits are selected such that the resolution of the front is comparable to the fixed grid. In two-dimension we have found that 2-4 elements per grid mesh is a good rule of thumb. For three-dimensional elements we usually examine the lengths of the edges of each element as well as its "aspect" ratio. The aspect ratio is defined as the length of the perimeter squared divided by the area of the element, normalized by the corresponding ratio for an equilateral triangle. If the length of the longest edge is longer than a preset value we split it and add two new elements and if the shortest edge is shorter than a minimum value, we collapse it and eliminate two elements. Elements are also added if the aspect ratio of an element is larger than a minimum value and its shortest edge is larger than the minimum size. We have found that a minimum edge length of $h/3$, a maximum length of h, and a maximum aspect ratio of 1.5 usually works well. Additional checks, such as making sure that a deletion or addition does not result in poorly shaped or connected elements, are usually also necessary. We do, for example, not allow restructuring that results in elements adjacent to a given element having more than one common point (the one shared by the original element). In many simulations we

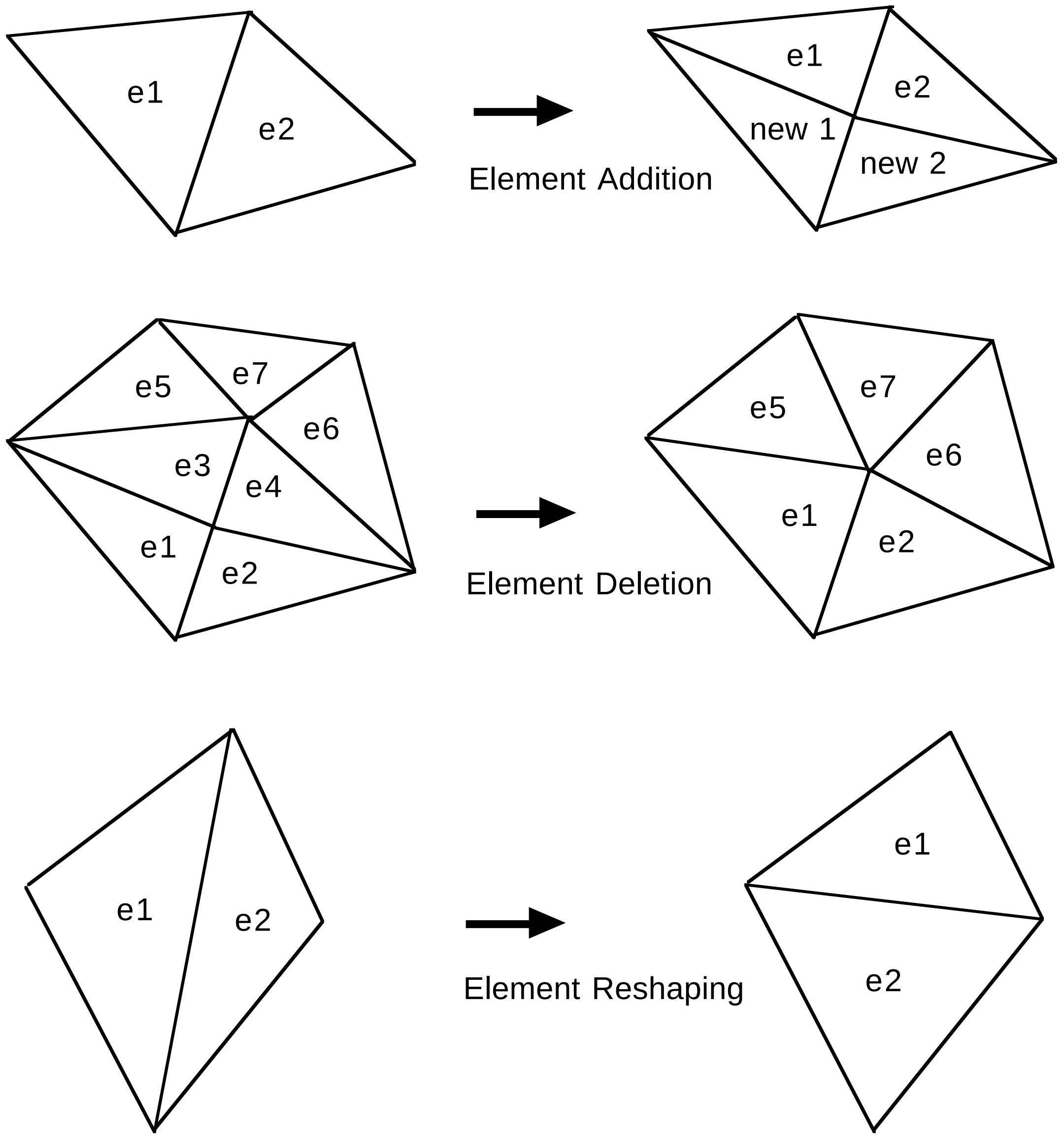

Figure 6: The restructuring of a three-dimensional front.

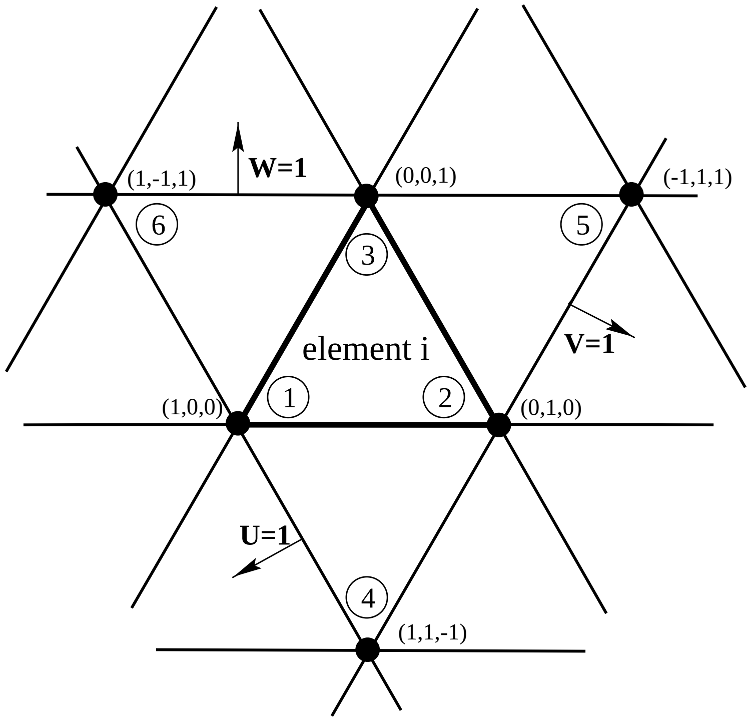

Figure 7: Barycentric coordinates for a three-dimensional element

do not carry out edge swapping since it is possible to show that a combination of element insertion and deletion will have approximately the same effect. In other cases we swap edges if that improves the aspect ratio of the elements.

2.2.4 Smoothing the front properties onto the fixed grid

Since the Navier-Stokes equations are solved on a fixed grid but surface tension is found on the front, it is necessary to convert a quantity that exists at the front to a grid value. When the density gradient is used to reconstruct the property fields on the fixed grid, it must also be transferred to the grid. Since the front represents a δ-function, the transfer corresponds to the construction of an approximation to this δ-function on the fixed grid. This "smoothing" can be done in several different ways, but it is always necessary to ensure that the quantity transferred is conserved. The interface quantities, ϕ_f, are usually expressed in units per unit area (or length in two-dimensions), but the grid value, ϕ_g, should be given in terms of units per unit volume. To ensure that total value is conserved in the smoothing, we must therefore require that:

$$\int_{\Delta s} \phi_f(s) ds = \int_{\Delta v} \phi_g(\mathbf{x}) dv. \tag{29}$$

This is accomplished by writing

$$\phi_{ijk} = \sum_l \phi_l w_{ijk} \frac{\Delta s_l}{h^3} \tag{30}$$

for a three-dimensional interpolation. Here ϕ_l is is a discrete approximation to the front value and ϕ_{ijk} is an approximation to the grid value. Δs_l is the area of element l. The weights must satisfy

$$\sum_{ijk} w_{ijk} = 1, \tag{31}$$

but can be selected in different ways. The number of grid points used in the interpolation depends on the particular weighting function selected. Since the weights have a finite support, there is a relatively small number of front elements that contribute to the value at each point of the fixed grid. In the actual implementation of the transfer of quantities from the front to the grid, we loop over the interface elements and add the quantity to the grid points that are near the front.

We usually write the weighting functions as a product of one-dimensional functions. In three-dimension, for example, the weight for the grid point (i, j, k) for interpolation to $\mathbf{x}_p = (x_p, y_p, z_p)$ is written as

$$w_{ijk}(\mathbf{x}_p) = d(x_p - ih)\; d(y_p - jh)\; d(z_p - kh) \tag{32}$$

where h is the grid spacing. For two-dimensional interpolation, the third term is set to unity. $d(r)$ can be constructed in different ways. The simplest interpolation is the area (volume) weighting:

$$d(r) = \begin{cases} (r - ih)/h, & 0 < r < h, \\ (h - (r - ih))/h, & -h < r < 0, \\ 0, & |r| \geq h\ , \end{cases} \tag{33}$$

Peskin (1977) suggested:

$$d(r) = \begin{cases} (1/4h)(1 + cos(\pi r/2h)), & |r| < 2h, \\ 0, & |r| \geq 2h. \end{cases} \tag{34}$$

A newer version of this function was used by Juric and Tryggvason (1997).

While it is, in principle, desirable to have a grid approximation that is as compact as possible, a very narrow support, obtained by using only a few grid points close to the front, usually results in increased grid effect. Nevertheless, we have found the area weighting to work very well in most cases although the functions proposed by Peskin are obviously smoother. Since area weighting involves only two grid points in each direction it is much more efficient in three-dimensions where it requires values from 8 grid points versus 27 for the Peskin interpolation functions. It also allows for simpler treatment of boundaries.

2.2.5 Updating the material properties

The fluid properties, such as the density, are not advected directly; instead the boundary between the different fluids is moved. It is therefore necessary to reset these quantities at every time step. The simplest methods is, of course, to loop over the interface points and set the density on the fixed grid as a function of the shortest normal distance from the interface. Since the interface is usually restricted to move less than the size of one fixed grid mesh, this update can be limited to the grid points in the immediate neighborhood of the interface. This straight forward approach (used, for example by Udaykumar *et al*), 1997) does have one major draw back: When two interfaces are very close to each other, or when an interface folds back on itself, such that two front segments are between the same two fixed grid points, then the property value on the fixed grid depends on which interface segment is being considered. Since this situation is fairly common, a more general method is necessary.

To construct a method that sets the density correctly even when two interfaces lay close to each other, we use the fact that the front marks the jump in the density and that this jump is translated into a steep gradient on the fixed grid. If two interfaces are close to each other, the grid gradients simply cancel. The gradient can be expressed as:

$$\nabla \rho = \int \Delta \rho \mathbf{n} \delta(\mathbf{x} - \mathbf{x}_f) ds. \tag{35}$$

and the discrete version is:

$$\nabla_h \rho_{ijk} = \sum_l \Delta \rho w_{ijk}^l \mathbf{n}_e \Delta l_e \tag{36}$$

where Δl_e is the area (length in two-dimension) of the element and w_{ijk}^l is the weight of grid point ijk with respect to element l.

Once the grid-gradient field has been constructed, the density field must be recovered. Again, this can be done in several ways. The simplest approach is to integrate the density gradient directly from a point where the density is known. If we distribute the gradient on a staggered grid (in two-dimension the x-gradient goes to the u-velocity points $(i+1/2, j)$ and so on, see Figure 2), and the density at (i, j) is known, then

$$\rho_{i+1,j} = \rho_{i,j} + h \nabla \rho_{i+1/2,j} \tag{37}$$

for example. Obviously, this operation only has to be performed at points near the front, since the density away from the front has not changed. This approach can, however, produce a density field that depends slightly on the direction in which the integration is done and also allows errors in the density gradient to propagate away from the interface. In most implementation of the method we use the following procedure: Taking the numerical divergence of the grid-density results in a numerical approximation to the Laplacian which should be equal to the Laplacian of the density field.

$$\nabla^2 \rho = \nabla_h \cdot \nabla \rho_{ij}. \tag{38}$$

The left hand side is approximated by the standard centered difference approximation for the Laplacian and solving the resulting Poisson equation with the appropriate boundary conditions yields the density field everywhere. Specifying the boundary density, when possible, usually results in higher accuracy. For intermediate density ratios this is a fairly robust procedure. Two types of error are possible. The density away from the interface may not be exactly equal to what it should be and small over and under shoots are occasionally found near the interface. To solve the first problem we often solve the Poisson equation by iterating only on points around the interface and thus leaving points away from the interface unchanged. The second problem can be dealt with by simple filtering. Small variations in density away from the interface can lead to unphysical buoyancy currents and undershoots can lead to negative densities that cause problems in the pressure solver.

Figure 8 shows the construction of the density field in a domain containing a circular drop. In the top row we show first the x and then the y-component of the grid density smoothed on a 32^2 grid by the Peskin distribution function (Equation 34). The grid density is then recovered by solving a Poisson equation (top row, on the right). The bottom row shows the density constructed by using area weighting on a 32^2 grid; by integrating the density gradient in the top row directly, first in the x and then the y direction and then averaging; and finally by solving a Poisson equation on a 64^2 grid. Obviously, the distribution function and the grid control how steep the transition zone is.

2.2.6 Computing surface tension

The accurate computation of the surface tension is perhaps one of the most critical elements of any method designed to follow the motion of the boundary between immiscible fluids for a long time. In our approach the front is explicitly represented by discrete points and elements and while this makes the surface tension computations much more straight forward than reconstructing it from a marker function, there are several alternative ways to proceed, some which are much better than others. In most of our simulations, we need the force on a front element but not the curvature directly. This simplifies the computations considerably. We will first describe the two-dimensional case and then the extension to three-dimension.

The force on a short segment of the front is given by:

$$\delta F_e = \int_{\Delta s} \sigma \kappa \mathbf{n} ds \tag{39}$$

Using the definition of the curvature of a two-dimensional line, $\kappa \mathbf{n} = \partial \mathbf{s} / \partial s$, we can write this as

$$\delta F_e = \sigma \int_{\Delta s} \frac{\partial \mathbf{s}}{\partial s} ds = \sigma (\mathbf{s}_2 - \mathbf{s}_1). \tag{40}$$

Therefore, instead of having to find the curvature, we only need to find the tangents of the end points. In addition to simplifying the computation, this ensures that the total force on any closed surface is zero, since the force on the end of one front element is exactly the same as the force on the end of the adjacent element. This conservation property is particularly important for long time computation where even a small error in the surface tension computation can lead to an unphysical net force on a front that can accumulate over time. This formulation also makes the extension to variable surface tension almost trivial. A simple expansion of the partial derivative shows that

$$\frac{\partial \sigma \mathbf{s}}{\partial s} = \sigma \frac{\partial \mathbf{s}}{\partial s} + \frac{\partial \sigma}{\partial s} \mathbf{s} = \sigma \kappa \mathbf{n} + \frac{\partial \sigma}{\partial s} \mathbf{s} \tag{41}$$

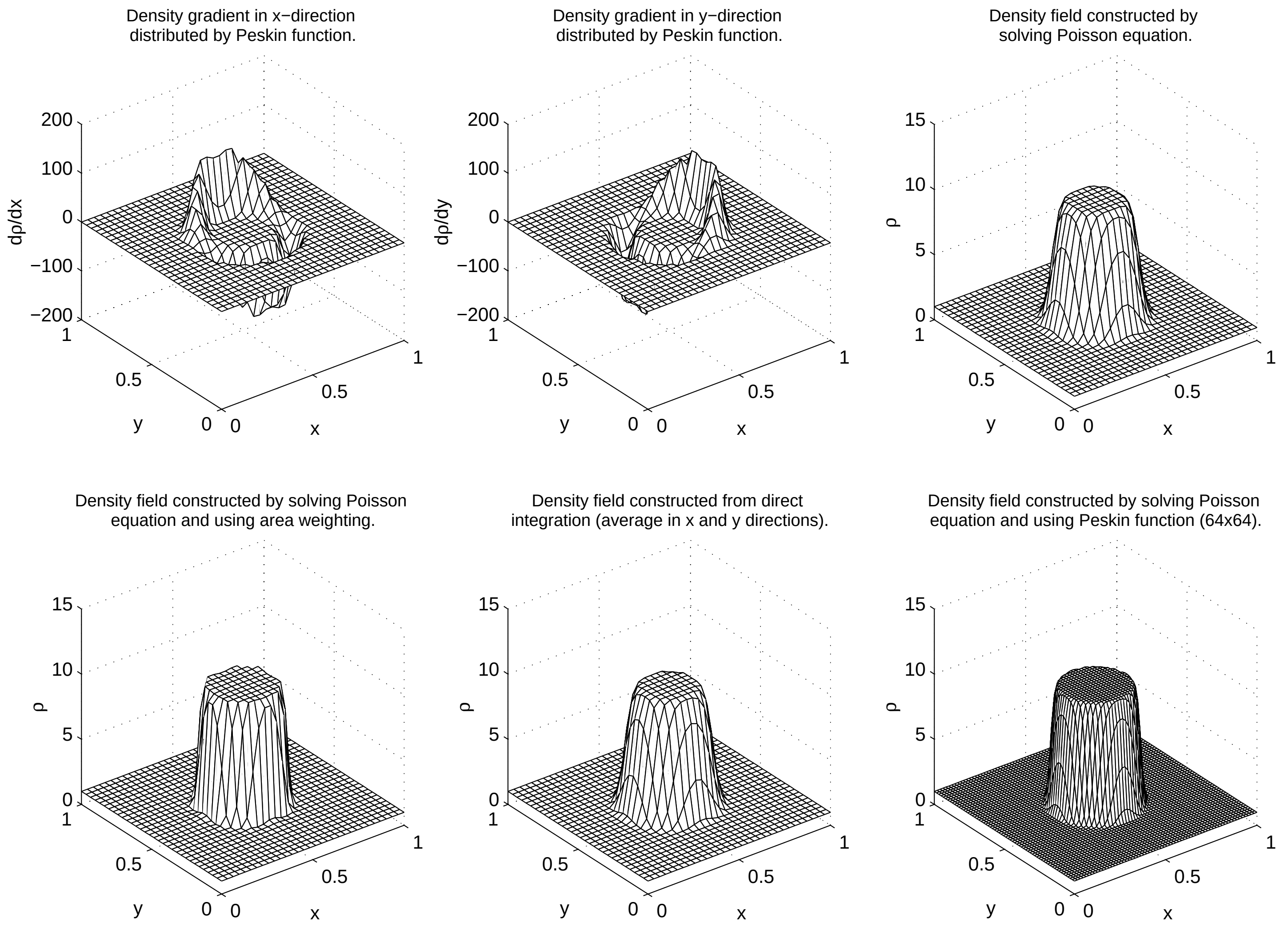

Figure 8: The generation of the density field from the density gradient.

which is the usual expression accounting for both the normal and the tangential force. Since

$$\delta F_e = \int_{\Delta s} \frac{\partial \sigma \mathbf{s}}{\partial s} ds = (\sigma \mathbf{s})_2 - (\sigma \mathbf{s})_1 \tag{42}$$

the force on each element is computed by simply subtracting the product of the surface tension coefficients and the tangents at the end points of each elements for both constant and variable surface tension.

The accuracy and efficiency of the computations depends on how we find the tangent vectors. In most of our two-dimensional computations we compute the tangents directly from a Legendre polynomial fit through the end points of the elements and the end points of the adjacent elements. Since this four point fit is not the same for two elements that share a common end point, we average the tangents computed for each element. The tangent to the curve is given by

$$\mathbf{t} = \frac{\partial \mathbf{x}}{\partial u} / || \frac{\partial \mathbf{x}}{\partial u} || \tag{43}$$

and using the polynomial written down in section (2.2.3), we find that the derivatives at point $u = 0$ is given by

$$\frac{\partial p}{\partial u} = \frac{1}{6}[-2p_{-1} - 3p_0 + p_1 - p_2] \tag{44}$$

for $u = 0$ and

$$\frac{\partial p}{\partial u} = \frac{1}{6}[p_{-1} - 6p_0 + 3p_1 + 2p_2] \tag{45}$$

for $u = 1$. In the actual code, we compute the tangents at the end points for each element and then average them when we compute the force. To test the accuracy of this approach, we have computed the curvature of a circle using unevenly spaced points. Since it is the integral of the curvature over each element that is actually computed, we divide by the exact arclength to obtain the curvature. To test the accuracy, we have put fourty points unevenly on a circle and computed the curvature in this way. Figure 9 shows the results. The points are shown on the left and the curvature as a function of arclength is shown on the right. Results for 80 points, distributed in the same manner, are shown by a dashed line. Obviously, the results are already quite accurate for 40 points and increasing the number of points improves the results even further.

For three-dimensional problems, we use the fact that the mean curvature of a surface can be written as

$$\kappa = (\mathbf{n} \times \nabla) \times \mathbf{n}. \tag{46}$$

The force on a surface element is therefore,

$$\delta F_e = \sigma \int_{\delta A} \kappa \mathbf{n} dA = \sigma \int_{\delta A} (\mathbf{n} \times \nabla) \times \mathbf{n} dA = \sigma \oint_s \mathbf{t} \times \mathbf{n} ds \tag{47}$$

where we have used the Stokes theorem to convert the area integral into a line integral along the edges of the element. Here, $\mathbf{t}$ is a vector tangent to the edge of the element and $\mathbf{n}$ is a normal vector to the surface. The cross product is a vector that is in the surface and is normal to the edge of the element. The surface tension coefficient times this vector gives the "pull" on the edge and the net "pull" is obtained by integrating around the edges. If the element is flat, the net force is zero but if the element is curved, the net force

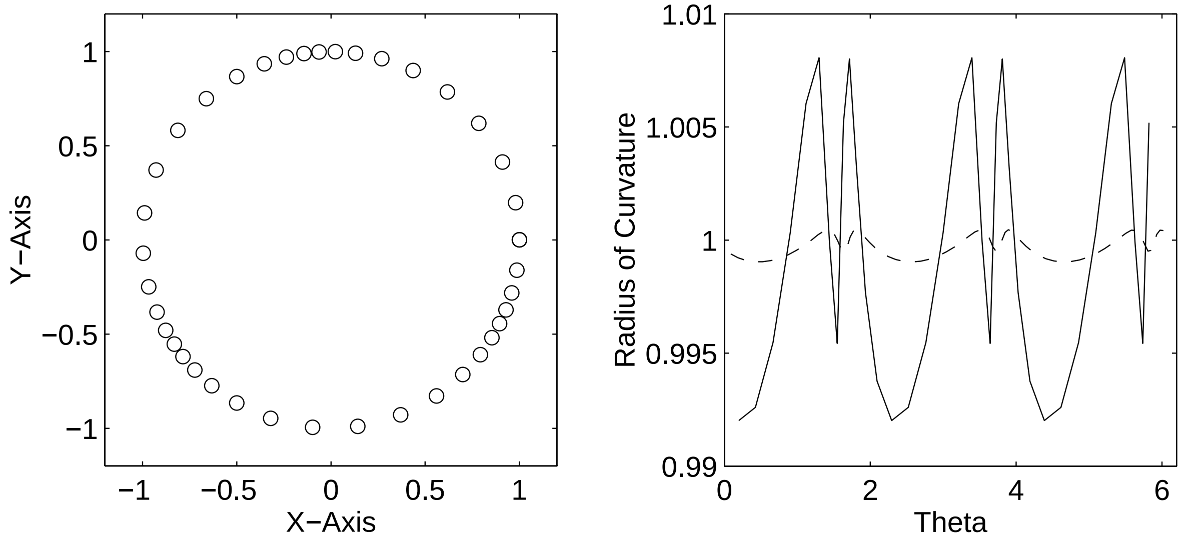

Figure 9: The accuracy of two-dimensional curvature calculations.

is normal to it when the surface tension coefficient is constant. As in two-dimensions, this formulation ensures that the net force on a closed surface is zero, as long as the force on the common edge of two elements is the same. This formulation also extends to variable surface tension in an obvious way:

$$\delta F_e = \int_{\Delta s} \sigma \mathbf{t} \times \mathbf{n} ds \tag{48}$$

For three-dimensional surfaces the computations are more involved than the two-dimensional version and although our current procedure works, it is not nearly as elegant as the two-dimensional one. At the moment, we explicitly fit a quadratic surface to the corner points of each element and the points of the elements that have a common edge to the element that we are working with. After transforming an element into a coordinate system where its corner points are in the $x' - y'$ plane with one point at $(0,0)$, we find the coefficients a, b, c, d, e such that the surface

$$z = ax + by + cxy + dx^2 + ey^2 \tag{49}$$

goes through the corner points of the elements that share a common edge with the element that we are working with. Once this surface is found, we compute the normal by

$$\mathbf{n} = (-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, -1)/H \tag{50}$$

where

$$H = \left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2 + 1 \tag{51}$$

and the tangent is simply computed by subtracting the end points of the edge. The edge is then divided into four segments and the integral evaluated by the midpoint rule. To ensure that the surface tension is conserved, this integral is averaged between two elements that share an edge when we find the force on each element (as is done in two dimensions). For the fit to work, it is important that the points are all distinct and this is the reason that we do not allow the elements adjacent to a particular element to share any other corner points. The approximation is obviously of a lower order than used in the two-dimensional simulations. Nevertheless, we have found the results to be quite accurate. For a sphere resolved by 200 elements, the maximum and the minimum curvature are within 3% of this value and when the sphere is resolved by 400 elements, the variations are about 1%.

Although we have not done so, it should be possible to write the vectors directly in terms of the barycentric coordinates and to evaluate the force without going through the explicit fitting of a continuous surface to the surface points.

We note that for closed surfaces it is possible to explicitly account for the pressure increase due to the average curvature and to work only with the perturbation curvature (Jan, 1994). The pressure and the curvature term are written as:

$$-\nabla P + \int \sigma\kappa\mathbf{n}ds = -\nabla P_o - \nabla P' + \int \sigma(\kappa_o + \kappa')\mathbf{n}ds \tag{52}$$

and since we assume that

$$\nabla P_o = \int \sigma\kappa_o\mathbf{n}ds \tag{53}$$

we are only left with the perturbation pressure and the perturbation curvature in equation (10). Since this is limited to closed surfaces, we usually do not do this.

2.2.7 Solving the pressure equation

For any incompressible flow, it is necessary to solve an elliptic equation reflecting the fact that pressure is always in equilibrium. For the velocity-pressure formulation this is a Poisson equation (Equation 20) for the pressure. If the density is constant, this equation can be solved by a large number of specialized techniques, such as those found in FISHPACK, but when the density depends on the spatial coordinates and the equation is non-separable, the choice of method is more limited. We use iterative techniques in all cases. During code development and for preliminary runs, we usually solve the pressure equation by a simple Successive Over Relaxation (SOR) iteration method. For production runs we use multigrid methods. We have used several packages, including MUDPACK Adams (1989), but most recently we have used a code written in-house (Bunner and Tryggvason, 1997). The reason for doing so was primarily the need to generate a parallel version of the solver. The solution of the pressure equation is usually the most time consuming part of the computation and must therefore be done efficiently.

In addition to changes in the pressure due to the flow, the pressure changes across a curved fluid interface when the surface tension coefficient is not zero. The pressure rise due to this effect can often be considerable. In Figure 10 we plot the pressure in a two-dimensional domain containing a single rising bubble. The various parameters are noted in the caption and we show the results for two different resolutions. The lower resolution result on the left are computed on a 32×64 grid and the higher resolution result on the right on a 64×128 grid. The transition between the outside and the inside of the bubble

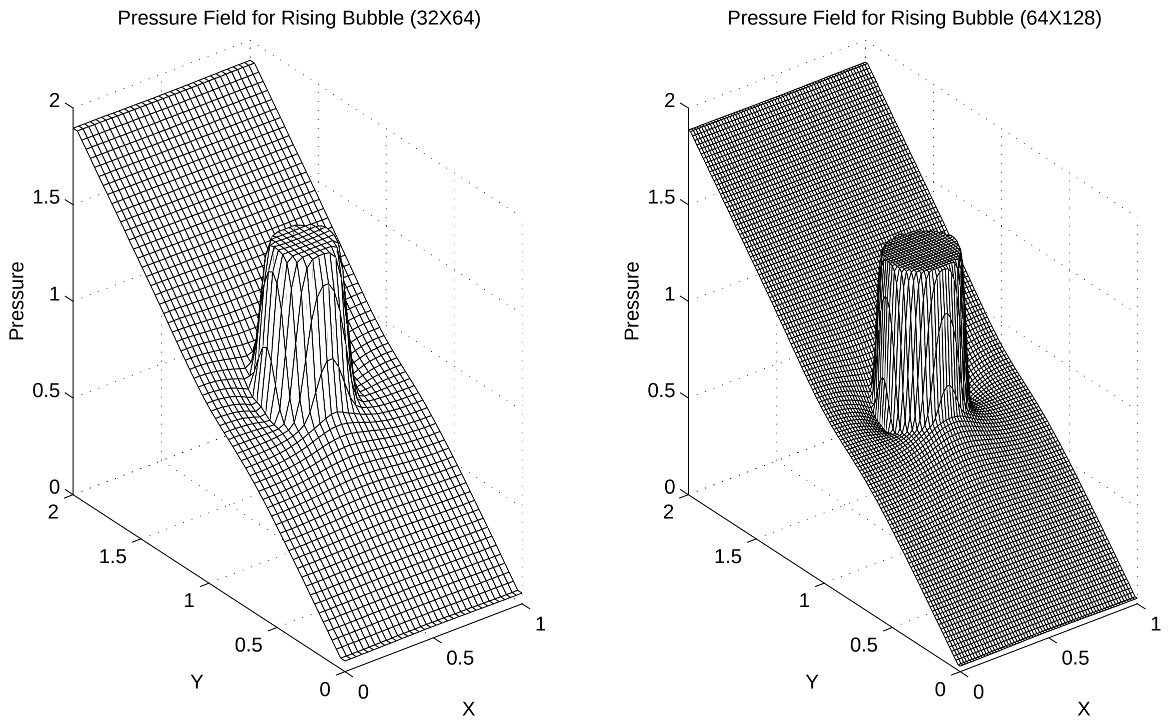

Figure 10: The pressure field for a bubble rising in a channel. The pressure rises due to hydrostatics toward the bottom. Results for a 32×64 grid are shown on the left and results for a 64×128 grid are shown on the right.

takes place over 2-3 grid points in both cases. It is important to note that the pressure is a computed quantity and the pressure jump results from a surface tension smoothed onto the grid using a Peskin distribution function.

The ease by which the pressure equation is solved depends generally on the density ratio. For large density ratios, small errors can lead to negative densities that usually cause convergence difficulties. These problems are, however, eliminated relatively easily by minor filtering. The more serious difficulty is that while SOR with a low overrelaxation parameter always allows us to solve the pressure equation, more efficient methods can fail to converge. For most practical purposes, this makes long computations with large density ratios impractical. In many cases we simply use density ratios that are relatively modest.

In addition to the convergence difficulties sometimes encountered at high density ratios, the pressure solution can cause other difficulties. If the surface tension coefficient is high, and if the representation of the surface forces on the grid has any significant anisotropy, unphysical velocities can be generated. These velocities, sometimes called "parasitic currents," are usually small. In Figure 11 we show the stream function for an initially cylindrical two-dimensional drop. The drop should remain exactly stationary and the velocity of the fluid should be exactly zero. Because of small pressure fluctuations in the surrounding fluid near the drop, slight recirculation is seen. These currents depends strongly on the grid resolution, smoothing function used, fluid viscosity, and surface tension coefficient. We have done a number of test and generally find that these currents are

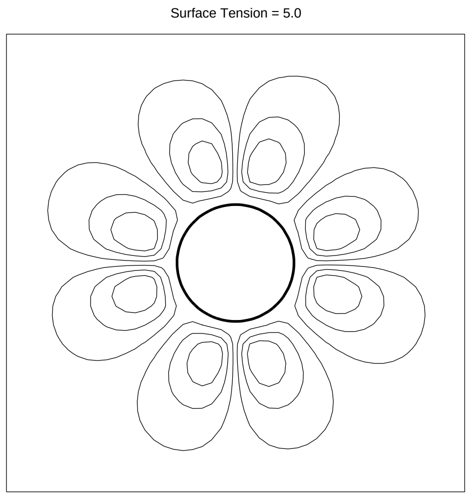

Figure 11: The parasitic current generated by a cylindrical drop with high surface tension. The computational domain is resolved by a 50^2 grid and the nondimensional velocities (defined in the text) are $O(10^{-5})$.

insignificant for well resolved problems with reasonably large surface tension. In particular, if the surface tension coefficient is such that a bubble or drop is slightly deformable, the parasitic currents do not appear to influence the solution. We have made an attempt to compare the magnitude of these currents with those observed by Lafaurie *et al* (1994) in VOF simulations. They concluded that because of this effect they were limited to drops smaller than about $10R_\nu$, where R_ν is a capillary-viscous length defined by $R_\nu = \rho\nu^2/\sigma$, and that the maximum velocity due to parasitic currents where about $0.01\sigma/\mu$. Here, ν is the kinematic viscosity. For a drop of diameter $R = 0.25$ in a 1×1 domain resolved by a 25^2 grid with $R/R_\nu = 125$ we find that the maximum velocity is $O(10^{-4})$. Increasing the viscosity or the resolution significantly decreases the current. Using area weighting increases the parasitic current by nearly an order of magnitude but not to the levels seen by Lafaurie *et al* (1994).

While the pressure gradient on a staggered grid is usually approximated simply by subtracting the pressure at point $i, j-1/2$ from the pressure at $i, j+1/2$, Zaleski (1997) has pointed out that if a curved sharp interface crosses a cell boundary and there is a significant difference in pressure across the interface, then the net pressure force may be poorly represented by the pressure at the midpoints. To approximate the pressure force more accurately, Zaleski has proposed to account for the contributions on either side of the interface explicitly. For two-dimensional situations this appears to be relatively straight forward, but in three-dimensions the complexities are likely to be considerably greater.

2.2.8 Advancing the front

Since the fluid velocities are computed on the fixed grid and the front moves with the fluid velocities, the velocity of the interface points must be found by interpolating from the fixed grid. The interpolation starts by identifying the closest grid point to the left and below the front point in the way described in section 2.2.2. The grid value is then interpolated by:

$$\phi_f = \sum_{ijk} w_{ijk}\phi_{ijk} \tag{54}$$

where the summation is over the points on the fixed grid that are close to the front point and ϕ stands for one of the velocity components. It is generally desirable that the interpolated front value be bounded by the grid values and that the front value be the same as the grid value if a front point coincides with a grid point. Although it is not necessary to do so, we usually use the same weighting functions to interpolate values to the front from the fixed grid as to smooth front values onto the fixed grid.

Once the velocity of each front point has been found, its new position can be found by integration. We usually use the same integration rule used for the integration of the momentum equation to advance the points, although that is not necessary. Thus, if a simple first order explicit Euler integration is used:

$$\mathbf{x}_f^{n+1} = \mathbf{x}_f^n + \mathbf{v}_f^n \Delta t \tag{55}$$

where $\mathbf{x}_f$ is the front position, $\mathbf{v}_f$ is the front velocity, and Δt is the time step.

While the momentum equations are usually solved in the conservative form, the advection of the front is not conservative. Unlike the VOF method, for example, errors are likely to results in changes in the total mass. Accurate advection of the front points minimizes this error and we have done a large number of simulations of bubbles, for example, where the change in mass remains within 1-2% during a time when the bubbles move about 100 diameters. In some cases, particularly for very long runs with many bubbles or drops where the resolution of each particle is relatively low, we have encountered changes in mass that are unacceptably high. In these case we correct the size of the particles every few time steps. Since the correction is very small at each time, the effect on the result is negligible. The inaccuracy in the advection of the front is due to errors coming from the interpolation of the velocities and the integration scheme. Increasing the accuracy of the front advection by using a higher order time stepping method is straight forward. The error due to the interpolation comes from the fact that although the discrete velocity field may be divergence free (for incompressible flows), the interpolated velocity field is not necessarily divergence free. An interpolation scheme that produces a divergence free velocity at the front points has been developed by Peskin and Printz (1993). The result is, however, a more complex pressure equation and we have not implemented this technique. Interpolation errors appear primarily to be due to poor resolution and should therefore generally be small. A test of the accuracy of the time integration has been done by Juric (1997) who advected an initially circular blob of fluid by a prescribed velocity field that deformed the blob into a long ligament. Mass was conserved very well during the simulation and when the velocity was reversed, the circle was recovered nearly perfectly. This test has been used for several other methods that either track or capture interfaces and it is generally found that tracking produces superior results.

For the basic method, the velocities of the front are found in this way, but it is sometimes also necessary to obtain front values for other quantities, such as temperature, that are available on the fixed grid. Those values are interpolated in exactly the same way.

2.2.9 Changes in the Front Topology

In general, numerical simulations of multiphase flow must account for topology changes of the phase boundary when, for example, drops or bubbles coalesce. When the interface is explicitly tracked by connected marker points, such changes must be accounted for by modifying the front in the appropriate way. The complexity of this operation is often cited as the greatest disadvantages of front tracking methods. In methods that follow the phase boundary by a marker function, topology changes take place whenever two interfaces, or different parts of the same interface, come closer than about one grid spacing. While automatic coalescence can be very convenient in some cases, particularly if the topology change does not need to be treated accurately, this is also a serious weakness of such methods. Coalescence is usually strongly dependent on how quickly the fluid between the coalescing parts drains and simply connecting parts of the interface that are close may give the incorrect solution.

Topology changes in multifluid flows can be divided into two broad classes:

- Films that rupture. If a large drop approaches an other drop or a flat surface, the fluid in between must be "squeezed" out before the drops are sufficiently close so that the film becomes unstable to attractive forces that can rupture it.

- Threads that break. A long and thin cylinder of one fluid will generally break by Rayleigh instability where one part of the the cylinder becomes sufficiently thin so that surface tension "pinches" it in two.

We note that exact mechanism of how threads snap and films break is still an active area of investigation. There are, however, good reasons to believe that threads can become infinitely thin in a finite time and that their breaking is "almost" described by the Navier-Stokes equations. Films, on the other hand, are generally believed to rupture due to short range attractive forces once they are a few hundred Angstrom thick. These forces are usually not included in the continuum description. To account for the draining of films prior to rupture requires the resolution of very small length scales and this is unlikely to be practical in most cases. It may also be unnecessary. We have examined the collision of two drops in detail (Nobari, Jan, and Tryggvason, 1996; Nobari and Tryggvason, 1996; Qian, Tryggvason, and Law, 1998) and generally find that the details of the collision is not sensitive to the resolution of the film between the drops. Indeed, a series of simulations where we examined the evolution of the film by using nonevenly spaced grids to resolve the film showed that the shape of the film was well predicted even when it was very poorly resolved. Drops with high surface tension, for example, produce a film of a small area that drains quickly whereas more deformable drops trap a large amount of fluid in a film with large area. Although the actual thickness was not predicted as well, this suggests that useful results can be obtained even on grids that are much coarser than the film. The reason is—most likely—that the flow in the film is essentially a plug flow of the same magnitude as the flow outside the film. It is therefore likely that the same

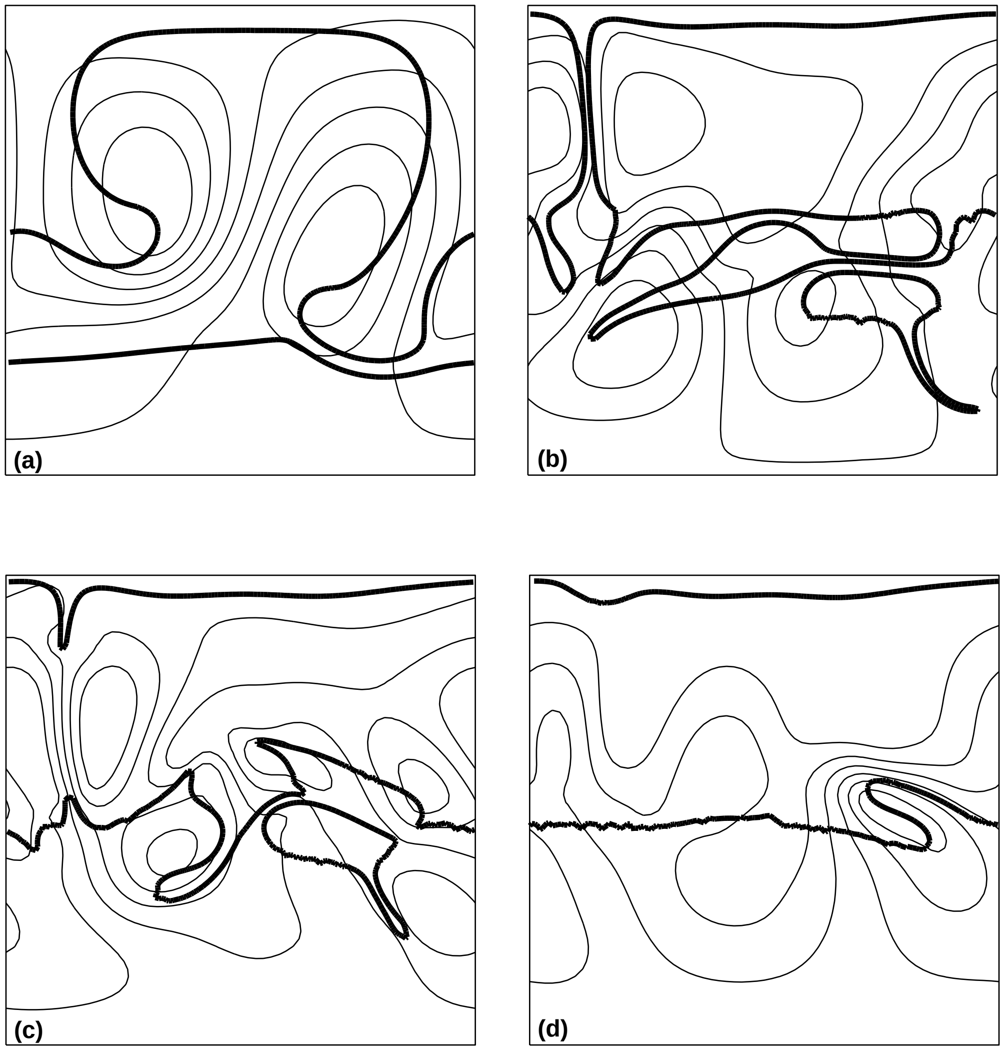

Figure 12: A simulation of the draining of a layer of fluid from the top of the computational domain. Here the topology of the front is changed if two fronts come close together. The front and a few streamlines are shown in each frame

conclusion will not hold for more complex interfacial conditions such as when surfactants or thermocapillary effects are present. The simplicity of the flow does, however, suggest that more detailed predictions could be accomplished by combining simple lubrication models for the draining with numerical simulations of the motion of the drop. In our simulations, so far, we have used very simple criteria for coalescence based either on a given time or a specified thickness of the film. Specifying the time of rupture is obviously the less general approach since it requires a previous knowledge of what the solution looks like, but results based on a given rupture time usually show little dependency on the grid resolution whereas results based on specifying the minimum thickness do. We note that while we would generally expect films to rupture due to attractive forces not included in the usual continuum description, there are recent evidence that films may become infinitely thin in a finite time under certain circumstances (Hou *et al*, 1996; Unverdi, Tauber, and Tryggvason, 1998)

Accomplishing topology changes in a front tracking code is a two step process. First, the part of the front that should undergo topology change must be identified and then the actual change must be done. For simple problems, the region where a change should take place is often obvious and no special search technique is needed. In general, however, rupture or coalescence can take place anywhere and it is necessary to search the whole front to find where two fronts or two parts of the same front are close to each other. The simplest, but least efficient, way to conduct this search is to compute the distance between the centroids of every front element. This is an $O(N^2)$ operation, but by dividing the computational domain into small subregions and looking only at the elements within each region, the efficiency of the search can be increased considerably.

In two-dimensional simulations, topology changes are rather simple. Figure 12 shows a simulation of the motion of an initially layered fluid. The fluid on the top and the bottom is heavier than the fluid in the middle and the top layer falls down and merges with the bottom layer. The computation was done on a 64^2 grid and the density ratio was 10. Two interfaces are merged if their separation is less than one grid spacing. In three-dimensions we have used a similar technique for colliding drops in Nobari and Tryggvason (1996) but the algorithm has not been generalized to the same degree as the two-dimensional one.

2.3 Parallelization

For large problems, it is necessary to use parallel computers. For most practical purposes, this means grids that are larger than about 64^3. The method described in the previous sections has been implemented for distributed memory parallel computers using the Message-Passing Interface, or MPI, library (Gropp *et al.*, 1995). The rectangular, three-dimensional domain is partitioned into even-sized subdomains by a simple domain decomposition and each subdomain is computed on a different processor. For the front, we take advantage of the physics of our problem, i.e., the presence of a large number of rather small bubbles. Each bubble is represented by its own data structure, which is communicated to the neighboring subdomains, or processes, as the bubble crosses the boundaries between the subdomains. Data coherence is maintained by a master-slave type approach. Each bubble has a master process which centralizes the advected front point positions from the slave processes, performs front restructuring and curvature cal-

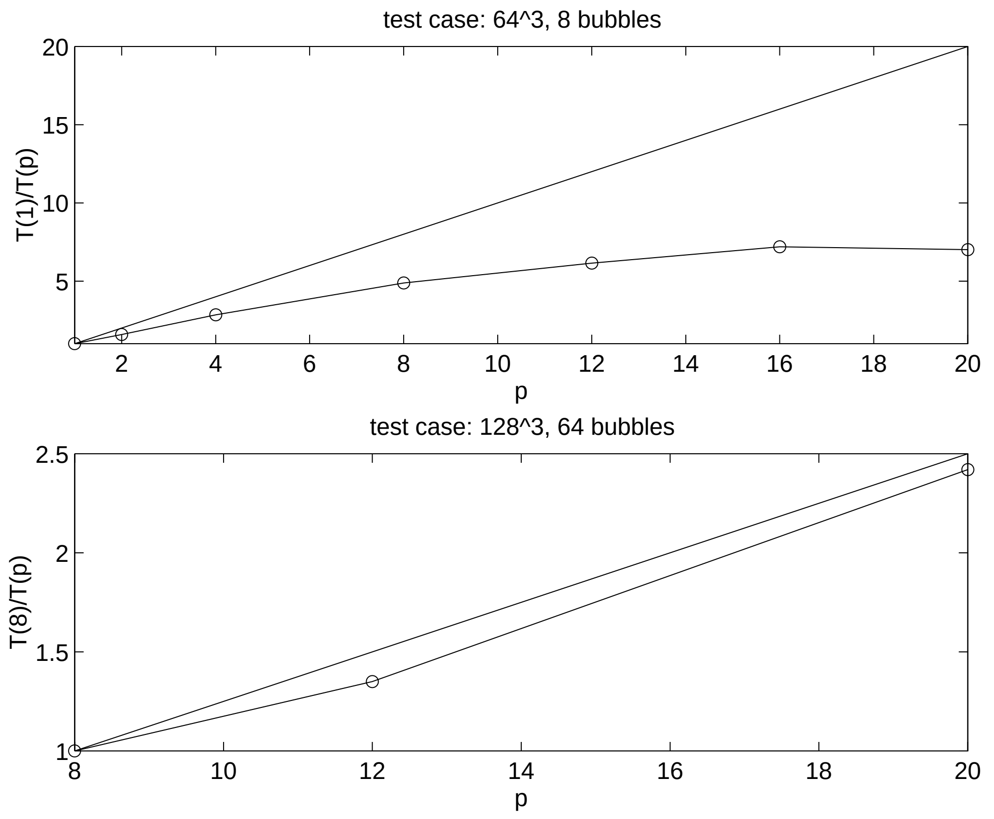

Figure 13: test of parellelization

culation, and then redistributes the data to the slave processes. A more general, and more complex, parallelization technique for the front is described by Glimm *et al.* (1998) who use domain depomposition to distribute the front to different processors.

An important criterion of the performance of a parallel code is its scalability. Scalability describes the ability of the code to achieve performance proportional to the number of processors (Kumar *et al.*, 1994) and one measure of the scalability is the speedup, usually defined as the time spent to solve a problem on one processor divided by the time spent to solve the same problem on p processors. The speedup of our code is plotted in the top half of Figure 13 for a test case with a 64^3 domain containing 8 bubbles. The code runs on a 48-node IBM-SP2 and this test case represents about the largest resolution that can be supported on a single node of that platform. For an ideal algorithm, the speedup is linear. For the 64^3 grid the speedup is less. The speedup for a larger test case, with a 128^3 domain and 64 bubbles, is represented in the bottom half of Figure 13. The first value is at $p = 8$ because it is the minimum number of processors that this problem requires. In this case the speedup is closer to the linear, ideal case. This indicates that the algorithm is better suited to a coarse-grain than to a fine-grain architecture and that optimum performance is achieved by using as few processors as possible. The main reason is the parallel multigrid solver for the elliptic equations for the density and pressure. When using simple SOR solvers, the code achieves nearly linear and in some cases superlinear speedups, but at the cost of prohibitively long runtimes. For 64^3 grid

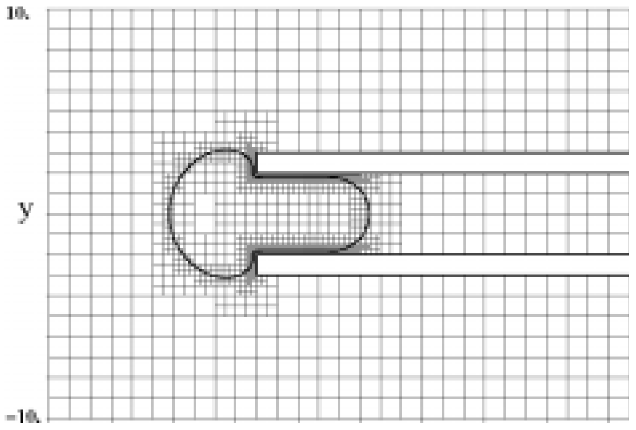

Figure 14: Adaptive grid refinement. From Agresar *et al.*

test case with 8 bubbles, a comparison between the serial code and the parallel code on one processor indicates a penalty of about 20% due to the parallelization. We have also examined the performance of the code when the number of bubbles is changed. For a 128^3 resolution and 8 processors, the time needed to solve for the flow around 8, 16, 32, 48, and 64 bubbles shows that doubling the number of bubbles increases the run time by less than 30%. This shows that the method is relatively insensitive to the number of bubbles, and that the total time required is mostly a function of the size of the fixed grid, not the front.

2.4 Adaptive Grid Refinement

With relatively few exceptions, fluid systems consist of "active" regions of concentrated vorticity (boundary and shear layers) and more "passive" regions where the flow is more uniform (often potential flow). Frequently, the "active" regions are only a small part of the whole flow field and using the fine grid required for the "active" regions everywhere results in excessive and unnecessarily fine grid in the passive regions. For large scale simulations, considerable savings can be realized using adaptive grids. Indeed, adaptive grids allow simulations of large systems that could not be simulated by a uniform grid.

Although nearly all of our simulations have been done on uniform grids, we have experimented with two types of adaptive gridding: one-dimensional stretching and locally refined Cartesian grids. The first approach is the simplest one. All grid lines are straight, but are allowed to be unevenly spaced. While this approach is very useful for simple problems where it is clear what the solution will look like and where high resolution is needed, it is not very general. The locally refined Cartesian grid refinement strategy is, however, more versatile. In this approach, the grid cells are rectangular but each cell can be refined by splitting it up into four (two-dimension) or eight (three-dimension) cells. The various levels of refinement are organized in a tree structure that allows each level to be accessed efficiently. This technique has been developed by several groups. See,

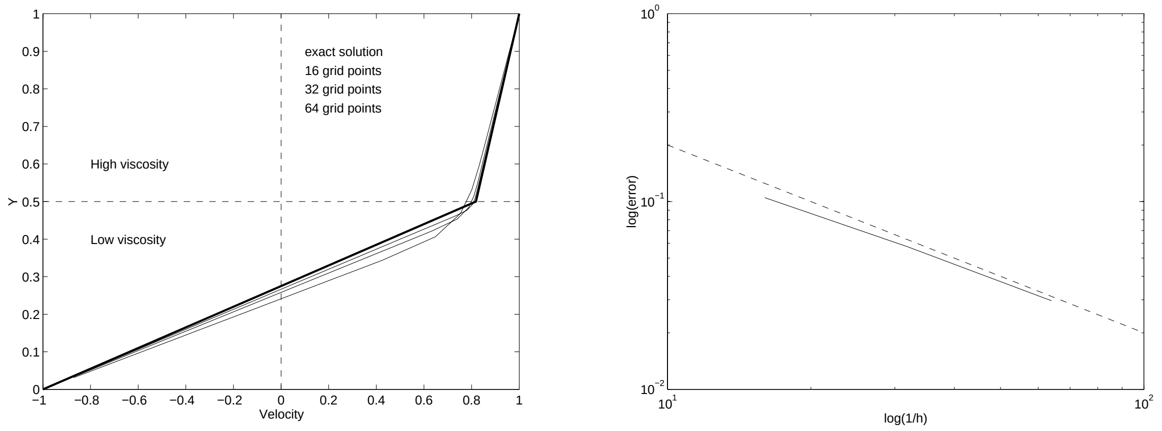

Figure 15: Stratified shear flow in a periodic channel. The viscosity of the fluid on the top is ten times the viscosity on the bottom. The velocity profile is shown on the left and the error in the total through flow on the right.

for example, Berger and Colella (1987), and Powell (1997). Sussman *et al* (1997) has used it for computations with the level set method. In Agresar *et al* (1998), our front tracking approach was combined with the grid refinement strategy to simulate the motion of two-dimensional and axisymmetric biological cells. Figure 14 shows one frame from an axisymmetric simulation of a cell being sucked up into a pipette.

3 Validation

While the "one fluid" formulation is a completely rigourous rewrite of the Navier-Stokes equations, the accuracy of a numerical scheme based on this reformulation must be established. It is indeed fair to say that accuracy is perhaps the concern most often raised by critics of the one fluid approach. While we will show later in this section that the method described here is capable of producing accurate results, it has to be admitted that the critisism is not completely without merit. Early implementations of the "one fluid" idea often lacked detailed convergence studies or relied on demonstrations that were not entirely convincing. Some of the early simulations using MAC and VOF by the Los Alamos group were done on very coarse grids by todays standards but nevertheless were very demanding on the computational resources available at the time. The resolution used in these computations was generally near the minimum needed for reasonable results and doubling the resolution was simply out of the question. Although both Daly and Pracht (1968) and Daly (1969), for example, examined the influence of several physical parameters on their results, no grid refinement appears to have been done.

Some aspects of methods for multiphase flows can be tested by very simple one dimensional model problems. Figure 15 shows one such test where we compute the steady state velocity profile in a simple shear flow. The top wall is moving to the right and the bottom wall is moving to the left. The bottom half of the channel contains one fluid and the top half another one. If the viscosity of both fluids is the same, a simple linear velocity profile would be obtained at steady state. If, however, the top fluid has much higher viscosity than the bottom one, then the shear in the more viscous fluid is smaller than in the less viscous one and it will move with the top wall, resulting in a finite velocity at the center line and a net flow to the right. An analytic expression for the centerline velocity and the net volume flux is easily found:

$$U_c = \frac{\mu_t - \mu_b}{\mu_t + \mu_b}; \qquad Q = 0.5\frac{\mu_t - \mu_b}{\mu_t + \mu_b} \tag{56}$$

In Figure 15(a) we show the velocity profile at steady state computed using several different resolutions. The thick straight lines are the exact solution. Two conclusions can be drawn from the figure. The first is that even on coarse grids the results are reasonably good (and excellent on the finer grids) and the second is that most of the error is in the less viscous fluid. The second observations provides some guidelines for the resolution needed for a given problem. If the motion is determined by the more viscous fluid, we expect more rapid convergence than if it is the less viscous fluid that matters most. Thus, for example, we see more rapid convergence for the motion of bubbles in a more viscous fluid than drops in a less viscous fluid. Similarly, we need fewer grid points to simulate the deformations of colliding drops (where the outer fluid has small effect) than their breakup (where the ambient fluid matters). By using Patankar's formulation for the viscous fluxes, the error is more evenly distributed. We emphasize, however, that in all cases it is possible to obtain accurate solutions on fine enough grids. The rate of convergence is examined in Figure 15(b) where we plot the error in the total flow rate as a function of the resolution used on a log-log graph. The dashed line indicates linear convergence. Although the method used here employs second order difference formulas for the spatial derivatives, we can not hope for more than a linear convergence since the sharp discontinuity is resolved on a fixed grid. The objective, however, is accuracy—and not order—and we note that this behavior is also found in modern highly accurate shock capturing schemes used in aeronautical computations of flows with shocks (Roe, 1986). Although Figure 15 only shows results for the steady state, we have examined the transient response and find very comparable errors.

The next level of complexity is to use simple analytical solutions for multidimensional problems as test cases. Such solutions are nearly all limited to linear oscillations around simple steady state solutions, usually assuming viscous effects to be negligible. The oscillation frequency of an inviscid drop can be found in most standard references, see Lamb (1932), for example:

$$\omega_n^2 = \frac{n(n+1)(n-1)(n+2)\sigma}{[(n+1)\rho_d + n\rho_o]R^3} \tag{57}$$

and n is the mode number. $n = 1$ corresponds to volume oscillations so the $n = 2$ is the lowest one for an incompressible drop. Assuming viscous effects to be small, Lamb also found that the amplitude would decay as

$$a_n(t) = a_o e^{-t/\tau} \qquad \text{where} \qquad \tau = \frac{R}{(n-1)(2n+1)\nu} \tag{58}$$

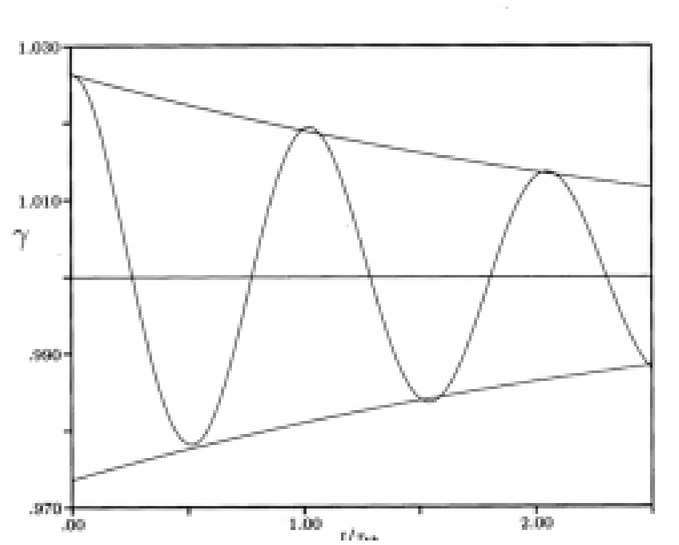

Figure 16: The amplitude of an oscillating drop

if the effect of the surrounding fluid is neglected. Here ν is the kinematic viscosity of the drop. Figure 16 shows the amplitude versus time for a drop with an initial amplitude perturbation equal to 2.5% of its radius. The drop radius is 1, the size of the computed domain is 5 and a 64×128 uniform grid is used to resolve the domain. The drop has a density equal to 100 times the density of the the surrounding fluid and the kinematic viscosity is 350 times higher. The time is nondimensionalized by the theoretical period for the lowest ($n = 2$) mode. The theoretical prediction for the amplitude versus time is also shown. Obviously, both the oscillation frequency as well as the amplitude are in good agreement with the theoretical prediction.

In addition to examining the oscillations of an axisymmetric drop, we have also computed the oscillation frequency of two-dimensional drops (Esmaeeli and Tryggvason, 1997) and compared it with an analytical expression for the frequency (See Lamb, 1932, for the single fluid case and Fyfe, Oran, and Fritts, 1988, for the two fluid case):

$$\omega_n^2 = \frac{(b^3 - n)\sigma}{(\rho_d + \rho_o)R^3} \tag{59}$$

For a two-dimensional drop of diameter 0.4 in a 1 by 1 computational domain, with its density 20 times larger than the ambient fluid, we find that the oscillation period is 9.3% longer than the theoretical value on a 32^2 grid, 3.4% longer on a 64^2 grid, and 1.9% longer on a 128^2 grid, when the initial amplitude is 5% of the drop radius. Comparisons of the growth rate of a nearly flat interface subject to a shear (Unverdi, Tauber, and Tryggvason, 1998) and the propagation of a linear waves (Yang and Tryggvason, 1998) result in a similar agreement. In addition to comparisons with linear perturbation solutions in the inviscid limits, we have made comparisons between high Reynolds number transient motions and inviscid solutions computed by a vortex method. For short times, the agreement is excellent (Han, 1998).

Analytical solutions also exists for several problems in the zero Reynolds number limit (Stokes Flow). We have compared the rise velocity of regular arrays of low Reynolds

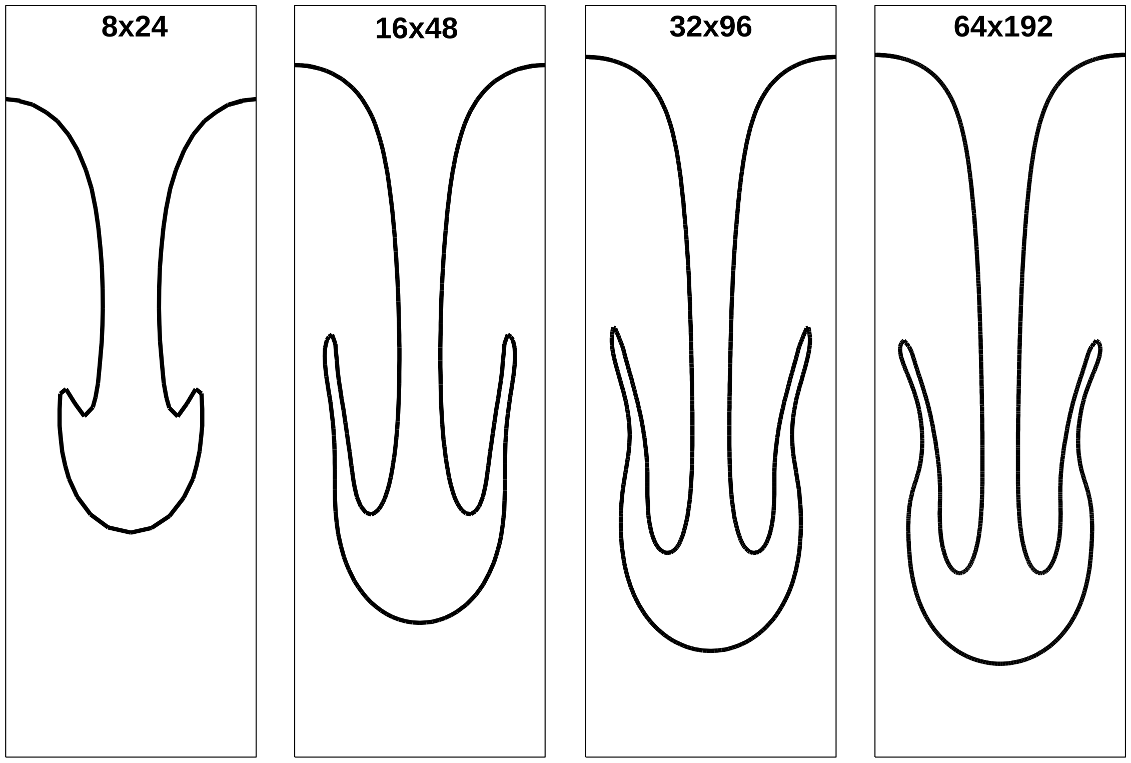

Figure 17: Grid refinement test

number viscous drops with the result of Sangani (1988) for drops in a Stokes flow (Esmaeeli and Tryggvason, 1998). For density and viscosity ratios of 1/10 and a void fraction of $\alpha = 0.0335$, Sangani's Reynolds number is 0.0599 and our Reynolds number is 0.051 for a grid resolution of 34^3 and 0.06158 for a 66^3 grid. This gives a relative error of 8.9% and 2.57%, respectively. At $\alpha = 0.1256$, Sangani's Reynolds number is 0.0394 and our Reynolds number is 0.037 at grid resolution of 34^3 and 0.0395 at grid resolution of 66^3. This gives a relative error of 4.99% and 0.445%, respectively.

To test for the fully nonlinear aspect of the method at finite Reynolds numbers, we must resort to grid resolution studies, other numerical solutions, and experiments. While all are subject to considerable uncertainty, we have done a number of such tests. For a comparison between computational results and experimental data see Qian, Tryggvason, and Law (1998), where binary collisions of equal size hydrocarbon drops are studied in high pressure environments. Using an axisymmetric version of our code we have compared our results with one case computed by Ryskin and Leal (1984). For $Re = 20$ and $We = 12$, they found $C_d = 0.33$ on the grid that they used for most of their computations. We found Eo and M from these values and followed the motion of a bubble, using a very large domain and about 25 grid points per bubble radius, until it reached steady state velocity. This velocity was within 2% of Ryskin and Leal's prediction (see Jan and Tryggvason, 1995). Comparison with other cases computed by Dandy and Leal (1989) show similar agreement.

In Figure 17, we show a grid refinement test for the Rayleigh-Taylor instability. This is a common test problem for methods intended to simulate multifluid flows. A heavy fluid is initially placed above a lighter one and the boundary given a small perturbation. The heavy fluid falls down in a relatively narrow "spike" while the lighter fluid rises upward as a large "bubble." For finite density ratios, the spike forms a mushroom shaped end. In Figure 17 the large amplitude stage is shown for four different resolutions, noted at the top of each frame. The density and viscosity of the lower fluid are 1 and 0.01, respectively. the top fluid is 5 times heavier and 10 times more viscous. The surface tension coefficient is 0.015 and the wavelength is 1. Even when there are only 8 grid points per wave, the results show the general behaviour of the solution, although the details are far from being converged. As the resolution is increased, the bubble shape converges quickly, but since the surface tension coefficient here is low, so the solution can generate relatively small features, slight differences are seen in the "spike" between the finest two grids. Other grid refinement studies can be found in Unverdi and Tryggvason (1992), Esmaeeli and Tryggvason (1997), and Han (1998).

We end this section by two examples that while not being validation studies, show the capability of the method. Figure 18 contains two frames from a simulation of four bubbles on a 512^2 grid. The governing parameters correspond to an air bubble in water except the density ratio is 20. In addition to the front marking the bubble surface, a few vorticity contours are also plotted. The average rise Reynolds number is plotted in Figure 19, showing that the bubble rise Reynolds number reaches a value of about 800 which is roughly what one would expect from experimental studies (where the bubbles are fully three-dimensional).

As discussed in the section on the solution of the pressure equation, multigrid methods sometime fail when the density difference between the fluid are large. A SOR iteration will, however, always converge as long as the density field is well behaved. In Figure 20 we show one frame from a simulation of the rise of two bubbles with a density ratio of a 100 (left) and another frame from a simulation with a density ratio of a 1000 (right). The density of the continuous fluid is 1, its viscosity is 0.001, the surface tension coefficient is 0.25, and gravity acceleration is 2. The kinematic viscosity in both fluids is the same and the bubble radius is 0.2. The computational domain is a periodic unit square resolved by a 64^2 grid. The bubble surface and a few streamlines with respect to a stationary frame of reference are plotted at the same time for both runs. The bubbles have risen about six diameters and we can see that the bubbles on the right are very slightly ahead, as we expect. The average Reynolds number is about 28. Both simulations were run using a simple SOR iteration method to solve the pressure equation. For the high density case, the solver sometimes required more than $10,000$ iterations with an over relaxation parameter of 1.2. A detailed comparison of the figure shows that the fluid motion inside the bubble is not as smooth for the high density ratio, suggesting that better resolution may be needed.

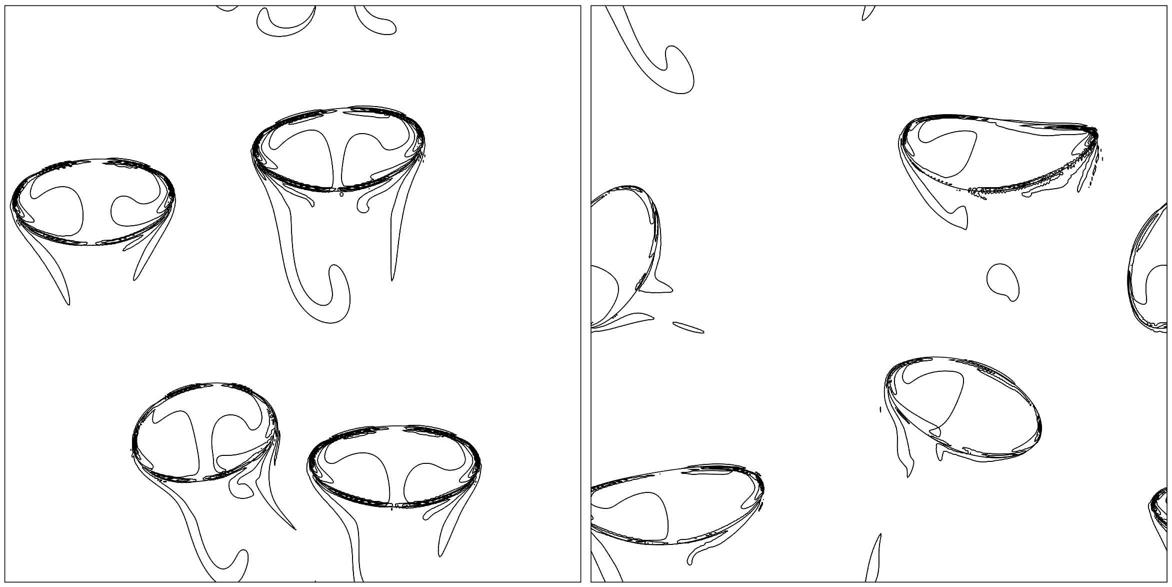

Figure 18: High Reynolds number bubbles at two times. The bubble surface and a few vorticity contours are plotted in each frame.

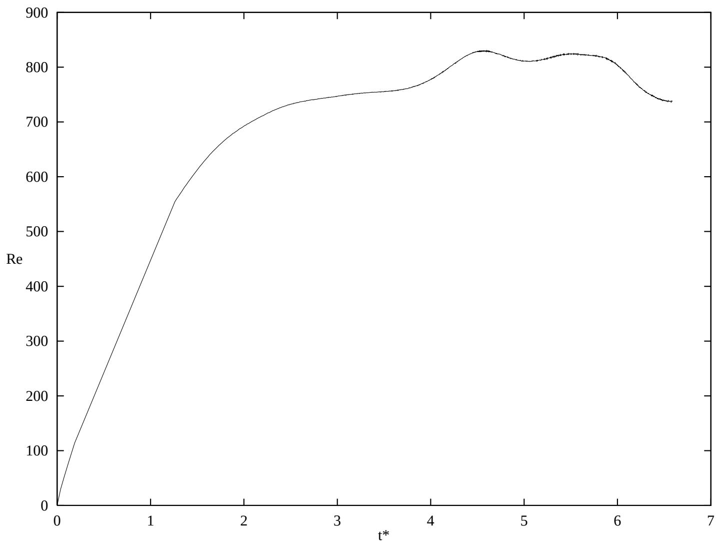

Figure 19: Reynolds number versus time for the bubbles in Figure 18.

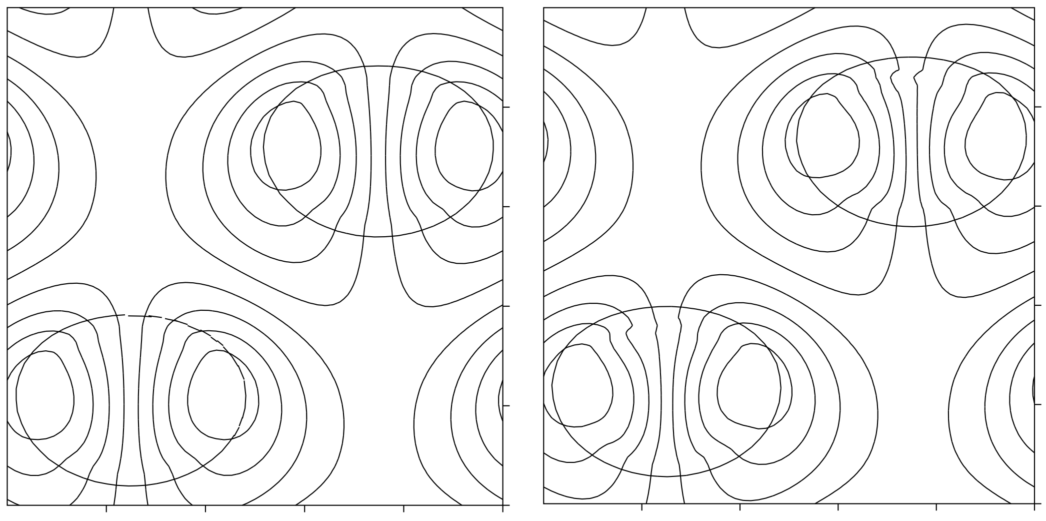

Figure 20: Simulation of the motion of two, two-dimensional bubbles with a high density ratio. Left: $\rho_o/\rho_b = 100$. Right: $\rho_o/\rho_b = 1000$.

4 Conclusion

Attempts to simulate multiphase flows go back to early days of computational fluid dynamics at Los Alamos. While a few successful simulations can be found in the literature, major progress has been made in the last few years. The "one field" formulation is the key to much of this progress. While the method described here is one of the most successful implementation of the "one field" formulation, impressive results have been obtained by improved VOF methods, Level Set methods, phase field methods, and the new CIP method. The key difference between these methods and the technique described here is our use of a separate "front" to mark the phase boundary, instead of a marker function. While explicit front tracking is generally more complex that the advection of a marker function, we believe that the increased accuracy and robustness is well worth the effort. The explicit tracking of the interface not only eliminates errors associated with the advection of a marker function and surface tension computations but we believe also that the flexibility inherent in the explicit tracking approach is important for the application of the method to problems where complex interface physics must be accounted for. In addition to several different implementations of the method described here in our group, it has been implemented independently by several other investigators such as Sheth and Pozrikidis (1995) who used it to examine the deformation of a two-dimensional drop in a shear flow, and Udaykumar *et al* (1997) who studied the motion of two-dimensional drops in pipes with variable cross section.

The method could be improved on in many ways. We mention only two issues here, one

numerical and another physical: The development of efficient methods for the solution of the pressure equation for very large density ratios would increase the efficiency not only of our method, but of all other methods based on the "one field" formulation. The draining of thin films and their eventual rupture is unlikely to be fully resolved by computations where the goal is, for example, to follow accurately the motion of many bubbles or drops. Such computations would greatly benefit from subgrid models that would determine the minimum film thickness and the probable time of coalescence.
Acknowledgment. The developement of the method described here has been supported by the National Science Foundation.

References

[1] J. Adams, "MUDPACK: Multigrid FORTRAN Software for the Efficient Solution of Linear Elliptic Partial Differential Equations," *Applied Math. and Comput.* **34**, p. 113, (1989).

[2] G. Agresar, J. J. Linderman, G. Tryggvason and K.G. Powell. "An Adaptive, Cartesian, Front-Tracking Method for the Motion, Deformation and Adhesion of Circulating Cells," *J. Comput. Phys.*, to appear.

[3] M. J. Berger and P. Colella, "Local Adaptive Mesh Refinement for Shock Hydrodynamics," Technical Report UCRL-97196, Lawrence Livermore National Laboratory, (1987).

[4] L.P. Bernal, P. Maksimovic, F. Tounsi and G. Tryggvason, "An Experimental and Numerical Investigation of Drop Formation by Vortical Flows in Microgravity," AIAA 94-0244. Presented at the 32th AIAA Aerospace Sciences Meeting, Reno, NV, Jan. 10-13, (1994).

[5] J. U. Brackbill, D. B. Kothe and C. Zemach, "A Continuum Method for Modeling Surface Tension," *J. Comput. Phys.* **100**, p. 335-354, (1992).

[6] B. Bunner and G. Tryggvason, "Simulations of Large Bubble Systems," *ASME Fluids Engineering Division Summer Meeting*, Vancouver, Canada, June 22-26, (1997).

[7] G. L. Chahine, "Strong Interactions Bubble/Bubble and Bubble/Flow," In: J.R. Blake (editor), Proc. *IUTAM* Conference on Bubble Dynamics and Interfacial Phenomena. Kluwer, (1994).

[8] A. V. Coward, Y. Y. Renardy, M. Renardy and J. R. Richards, *J. Comp. Phys.,* **132**, p. 346-361, (1997).

[9] B. J. Daly, "Numerical Study of the Effect of Surface Tension on Interface Instability," *Phys. Fluids*, **12**, p. 1340-1354, (1969).

[10] B. J. Daly, and W. E. Pracht, "Numerical Study of Density-Current Surges," *Phys. Fluids*, **11**, p. 15-30, (1968).

[11] D.S. Dandy and G.L. Leal, "Buoyancy-Driven Motion of a Deformable Drop Through a Quiescent Liquid at Intermediate Reynolds Numbers," *J. Fluid Mech.*, **208**, p. 161-192, (1989).

[12] E.A. Ervin and G. Tryggvason, "The Rise of Bubbles in a Vertical Shear Flow." *ASME J. Fluid Engineering 119*, p. 443-449, (1997).

[13] A. Esmaeeli, "Numerical Simulations of Bubbly Flows," *Ph.D. Dissertation*, The University of Michigan, (1995).

[14] A. Esmaeeli and G. Tryggvason, "An Inverse Energy Cascade in Two-Dimensional, Low Reynolds Number Bubbly Flows," *J. Fluid Mech.*, **314**, p. 315-330, (1996).

[15] A. Esmaeeli and G. Tryggvason, "Direct Numerical Simulations of Bubbly Flows. Part I-Low Reynolds Number Arrays," (1997), submitted for publication.

[16] A. Esmaeeli and G. Tryggvason, "Direct Numerical Simulations of Bubbly Flows. Part II-Moderate Reynolds Number Arrays," (1998), submitted for publication.

[17] J. Feng, H. H. Hu and D. D. Joseph, "Direct Simulation of Initial Value Problems for the Motion of Solid Bodies in a Newtonian Fluid, Part 1. Sedimation," *J. Fluid Mech.* **261**, p. 95-134, (1994).

[18] J. Feng, H. H. Hu and D. D. Joseph, "Direct Simulation of Initial Value Problems for the Motion of Solid Bodies in a Newtonian Fluid, Part 2. Couette and Poiseuilli Flows," *J. Fluid Mech.* **277**, p. 271-301, (1995).

[19] J. Fukai, Y. Shiiba, T. Yamamoto, O. Miyatake, D. Poulikakos, C. M. Megaridis and Z. Zhao, "Wetting Effects on the Spreading of a Liquid Droplet Colliding with a Flat Surface: Experiment and Modeling," *Phys. Fluids* **7**(2), p. 236-247, (1995).

[20] D. E. Fyfe, E. S. Oran and M. J. Fritts, "Surface tension and viscosity with Lagrangian Hydrodynamics on a Triangular Mesh," *J. Comput. Phys.* **76**, p. 349-384, (1988).

[21] J. Glimm, "Nonlinear and Stochastic Phenomena: The Grand Challenge for Partial Differential Equations," *SIAM Review* **33**, p. 625-643, (1991).

[22] W. Gropp, E. Lusk and A. Skjellum, "Using MPI: Portable Parallel Programming with the Message-Passing Interface," The MIT Press (1995).

[23] J. Han, "Numerical Studies of Drop and Motion in Axisymmetric Geometry," *Ph.D. Dissertation*, The University of Michigan, (1998).

[24] H. H. Hu, "Direct Simulation of Flows of Solid-Liquid Mixtures," *Int. J. Multiphase Flow*, **22**, p. 335-352, (1996).

[25] H. H. Hu, *http://www.lrsm.upenn.edu/ howard/homepage.html.*

[26] T. Y. Hou, J. S. Lowengrub and M. J. Shelley, "The Long-Time Motion of Vortex Sheets with Surface Tension," *Phys. Fluids* **9**(7), p. 1933, (1997).

[27] D. Jacqmin, "An Energy Approach to the Continuum Surface Tension Method," Technical Report AIAA 96-0858, (1996).

[28] Y. J. Jan, "Computational Studies of Bubble Dynamics," *Ph.D. Dissertation*, The University of Michigan, (1994).

[29] D. Juric, *http://www.lanl.gov/home/djuric.*

[30] D. Juric, "Computations of Phase Change," *Ph.D. Dissertation*, The University of Michigan, (1996).

[31] D. Juric and G. Tryggvason, "A Front Tracking Method for Dentritic Solidification," *J. Comput. Phys.* **123**, p. 127-148, (1996).

[32] D. Juric and G. Tryggvason. "Computations of Boiling Flows," to appear in *Int'l. J. Multiphase Flow.*

[33] I. S. Kang and L. G. Leal, "Numerical Solution of Axisymmetric, Unsteady Free-Boundary Problems at Finite Reynolds Number. I. Finite-Difference Scheme and its Applications to the Deformation of a Bubble in a Uniaxial Straining Flow," *Phys. Fluids*, **30**, p. 1929-1940, (1987).

[34] M. R. Kennedy, C. Pozrikidis and R. Skalak, "Motion and Deformation of Liquid Drops, and the Rheology of Dilute Emulsions in Simple Shear Flows," *Computers Fluids* **23**, p. 251-278, (1994).

[35] V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing*, Benjamin/Cummings, (1994).

[36] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski and G. Zanetti, "Modelling Merging and Fragmentation in Multiphase Flows with SURFER," *J. Comp. Phys.* **113**, p. 134-147, (1994).

[37] H. Lamb, *Hydrodynamics,* Dover, New York, (1932).

[38] M. Lowenberg and E. J. Hinch, "Numerical Simulation of a Concentrated Emulsion in Shear Flow," *J. Fluid Mech.*, In press, (1996).

[39] M. Manga and H. A. Stone, "Buoyancy-Driven Interactions between Deformable Drops at Low Reynolds Numbers," *J. Fluid Mech.* **256**, p. 647-683, (1993).

[40] S. Mortazavi, "Computational Investigation of Particulate Two-Phase Flows," *Ph.D. Dissertation*, The University of Michigan, (1995).

[41] S. Nas, "Computational Investigation of Thermocapillary Migration of Bubbles and Drops in Zero Gravity," *Ph.D. Dissertation,* The University of Michigan, (1995).

[42] S. Nas and G. Tryggvason, "Computational Investigation of the Thermal Migration of Bubbles and Drops," in AMD 174/FED 175 Fluid Mechanics Phenomena in Microgravity, Ed. Siginer, Thompson and Trefethen. p. 71-83, Presented at the ASME 1993 Winter Annual Meeting, ASME (1993).

[43] M. R. H. Nobari, "Numerical Simulations of Drop Collisions and Coalescence," *Ph.D. Thesis*, The University of Michigan, (1993).

[44] M. R. Nobari, Y.-J. Jan and G. Tryggvason. "Head-on Collision of Drops–A Numerical Investigation," *Phys. Fluids* **8**, p. 29-42, (1996).

[45] M. R. Nobari, and G. Tryggvason, "Numerical Simulations of Three-Dimensional Drop Collisions," *AIAA Journal* **34**, p. 750-755, (1996).

[46] E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow,* Elsevier, New York, (1987).

[47] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, (1980).

[48] C. S. Peskin, "Numerical Analysis of Blood Flow in the Heart," *J. Comput. Phys.* **25**, p. 220, (1977).

[49] C. S. Peskin and B. F. Printz, "Improved Volume Conservation in the Computation of Flows with Immersed Boundaries," *J. Comput. Phys.* **105**, p. 33-46, (1993).

[50] J. E. Pilliod and E. G. Puckett, "Second-Order Accurate Volume-of-Fluid Algorithms for Tracking Material Interfaces," Preprint, (1997).

[51] K. Powell, "Solution of the Euler Equations on Solution-Adaptive Cartesian Grids," to appear in *Annual Reviews in Computational Fluid Dynamics,* M. Hafez, editor, (1997).

[52] J. Quian, G. Tryggvason and C.K. Law, "An Experimental and Computational Study of Bounching and Deforming Droplet Collision," submitted to *Phys. Fluids.*

[53] J. R. Richards, A. M. Lenhoff and A. N. Beris, "Dynamic Breakup of Liquid-Liquid Jets," *Phys. Fluids* **6**, p. 2640-2655, (1994).

[54] P. L. Roe, "Characteristic-Based Schemes for the Euler Equations," *Ann. Rev. Fluid Mech.*, p. 337-365, (1986).

[55] G. Ryskin and L. G. Leal, "Numerical Solution of Free-Boundary Problems in Fluid Mechanics, Part 2. Buoyancy-Driven Motion of a Gas Bubble Through a Quiescent Liquid," *J. Fluid Mech.* **148**, p. 19-35, (1984).

[56] A. S. Sangani, "Sedimentation in Ordered Emulsions of Drops at Low Renolds Number," *J. of Appl. Math. and Physics, ZAMP* **38**, p. 542-555, (1988).

[57] K. S. Sheth and C. Pozrikidis, "Effects of Inertia on the Deformation of Liquid Drops in Simple Shear Flow," *Computers and Fluids* **24**, p. 101-119, (1995).

[58] P. J. Shopov, P. D. Minev, I. B. Bazhekov and Z. D. Zapryanov, "Interaction of a Deformable Bubble with a Rigid Wall at Moderate Reynolds Numbers," *J. Fluid Mech.* **219**, p. 241-271, (1990).

[59] M. Sussman, P. Smereka and S. Osher, "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flows," *J. Comput. Phys.* **114**, p. 146-159, (1994).

[60] M. Sussman, *Private Communication.*

[61] S. Takagi and Y. Matsumoto, "Force Acting on a Rising Bubble in a Quiescent Liquid," *ASME Fluids Engineering Division Conference.* FED-126, p. 575-580, (1996).

[62] T. Tezduyar, "Large-Scale Fluid-Particle Interactions," *http://www.arc.umn.edu/research /tezduyar/101sphere.html*, (1996).

[63] G. Tryggvason and H. Aref, "Numerical Experiments on Hele Shaw Flow with a Sharp Interface," *J. Fluid Mech.* **136**, p. 1-30, (1983).

[64] G. Tryggvason, "Numerical Simulation of the Rayleigh-Taylor Instability," *J. Comput Phys.* **75**, p. 253-282, (1988).

[65] G. Tryggvason and S. O. Unverdi, "Computations of Three-Dimensional Rayleigh-Taylor Instability," *Phys. Fluids A* **2**, p. 656-659, (1990).

[66] S. O. Unverdi and G. Tryggvason, "A Front-Tracking Method for Viscous, Incompressible, Multi-Fluid Flows," *J. Comput Phys.* **100**, p. 25-37, (1992).

[67] S. O. Unverdi and G. Tryggvason, "Computations of Multi-Fluid Flows," *Physica D* **60**, p. 70-83, (1992).

[68] S. O. Unverdi, W. Tauber and G. Tryggvason, *In Preparation.*

[69] H. S. Udaykumar, H. C. Kan, W. Shyy and R. Tran-Son-Tay, "Multiphase Dynamics in Arbitrary Geometries on Fixed Cartesian Grids," *J. Comput. Phys.* **137**, p. 366-405, (1997).

[70] T. Yabe, "Unified Solver CIP for Solid, Liquid and Gas," *Computational Fluid Dynamics Review,* (1997), to appear.

[71] Y. Yang and G. Tryggvason, "Dissipation of Energy by Finite Amplitude Surface Waves," *Computers and Fluids,* to appear.

[72] P.-W. Yu, S.L. Ceccio and G. Tryggvason, "The Collapse of a Cavitation Bubble in Shear Flows-A Numerical Study," *Phys. Fluids* **7**, p. 2608-2616, (1995).

[73] S. Zaleski, *Personal Communication.*