

The Point-Set Method: Front-Tracking without Connectivity

D. J. Torres and J. U. Brackbill

Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545

E-mail: dtorres@lanl.gov; jub@lanl.gov

Received November 18, 1999; revised September 13, 2000

We describe a point-set method for extracting the normal, curvature, and surface area from unordered data points residing on a surface. This capability relaxes front-tracking's reliance on connectivity between interfacial points and allows front-tracking to model topological changes at an interface naturally. We use this capability in 2D and 3D front-tracking calculations to model coalescence. We also describe a simple projection scheme which allows us to suppress parasitic currents in front-tracking models. © 2000 Academic Press

Key Words: front-tracking; connectivity; surface tension; parasitic currents.

1. INTRODUCTION

Front-tracking allows one to simulate fluid-flow problems in which there is a surface of discontinuity separating two fluids or fluid phases [1, 2]. The method combines the numerical solution of the Navier–Stokes equations with an explicit tracking of the phase boundary. Front-tracking has many advantages, among them its simplicity and its lack of numerical diffusion.

Front-tracking has been applied to a wide range of problems. It is used to study fluid instabilities [3], chemically reacting flows [4] and detonations [5], material failure [6], and phase changes [7]. The solution of immiscible flow problems is described in Unverdi and Tryggvason [1, 8], its application to drop collisions in [9], the thermal migration of drops in [10], and the dynamics of bubbly flows in [11–13]. A method for fluid flows with phase changes is described in papers on dendritic solidification [14] and boiling flows [15].

Current front-tracking methods depend upon connectivity among interfacial points to calculate the geometry of the interface. In a connectivity based scheme, one maintains a list of neighbors for each interfacial point. There is a cost for connectivity, however. Connectivity increases the complexity of the computer model when an interface undergoes topological changes (coalescence or breakup), especially in three dimensions [16]. (A detailed

description of the data structures and implementation issues for three-dimensional front tracking is given in Glimm *et al.* [2].)

However, connectivity is not a necessary requirement for front-tracking. One can determine the unit normal, curvature, and surface area of individual data points without relying on connectivity between data points. Such a capability allows one to redistribute or delete interfacial points on an interface without worrying about reconnecting points. For example, York [17, 18] describes a mass matrix method, which he uses to compute normals from unordered points and to model air bag inflation in two dimensions. York embeds the surface in a grid and constructs a characteristic function by expanding in tensor products of B-splines about grid points. York's formulation calculates values of the characteristic function on a grid, which when interpolated to the interfacial particles, give a constant value along the interface. Hoppe [19] computes normals by fitting the best least-squares plane to a surface at a point using neighboring points and taking the plane normal to be the point normal. Hoppe's algorithm is simple and robust, but the unit normals computed are not as accurate as York's for the smooth surfaces we considered in 3D.

The cost and difficulty of maintaining the logical connectivity of interfacial points, especially when modeling coalescence or breakup in three dimensions, has motivated our consideration of the geometric description of interfaces with unconnected points and our development of a new algorithm, the point-set method, which follows York's and Tryggvason *et al.*'s work as described below.

In Section 2, we review briefly the standard front-tracking method. In Section 3, we describe the point-set method. We discuss how one can compute normals and curvatures from point-set data by constructing an indicator function. We also explain our means of calculating surface areas of individual interfacial points without connectivity. In Section 4, we present numerical results for free surfaces. We test the accuracy of the geometric description given by the point-set method. For fluid flows with surface tension, we model droplet oscillations and coalescence in two and three dimensions. In Appendix 1, we describe an alternative to the usual projection method for incompressible flow, which reduces parasitic currents in numerical front-tracking calculations.

2. A BRIEF REVIEW OF FRONT-TRACKING

We briefly review front-tracking methods developed by Unverdi and Tryggvason [1] and Glimm *et al.* [2]. In these methods, an interface of logically connected points is embedded within a computational grid. The interface marks the boundary between two phases of a material whose density, temperature, and other properties may be quite different. Within each phase, however, it is assumed that fluid properties are constant. From the connected interfacial points, one forms surface elements. In 2D, the points are members of a linked list, and a surface element is defined to be the line segment between two adjacent members of the list. In 3D, associated with each point is a table of nearest neighbors, and the interfacial elements are triangles formed by connecting an interfacial point with two of its neighbors. Elements can be added, deleted, or reshaped during the course of a calculation, but when changes are made, the logical connectivity of the points must be updated.

In two dimensions, the surface tension force exerted upon an interfacial element, for example, is

$$\Delta \mathbf{F}_s = \sigma (\hat{t}_2 - \hat{t}_1), \quad (1)$$

where \hat{t}_1 and \hat{t}_2 are the unit tangent vectors at the endpoints of the segment [20] and σ is the surface tension coefficient. The tangent vectors are computed by fitting a Legendre polynomial through the end points of the elements and adjacent elements.

In three dimensions, the force on a triangular element [20] is

$$\Delta \mathbf{F}_s = \sigma \oint \hat{t} \times \hat{n} ds, \quad (2)$$

where the integral is over the closed line path bordering the interfacial element. The unit normal, \hat{n} , is found by fitting a quadratic surface through an element's vertices and those of its neighbors.

The interfacial forces are transferred to a volume grid using

$$\frac{\mathbf{F}_s}{\rho} \Big|_{\mathbf{x}_g} = \sum_p \frac{\Delta \mathbf{F}_s}{\rho} \Big|_{\mathbf{x}_p} \tilde{\delta}(\mathbf{x}_g - \mathbf{x}_p), \quad (3)$$

where $\tilde{\delta}(\mathbf{x}_g - \mathbf{x}_p)$ denotes a tensor product of one-dimensional approximate delta functions, \mathbf{x}_p are interfacial point locations, and \mathbf{x}_g are grid locations.

The momentum equation,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \frac{\mathbf{F}_s}{\rho}, \quad (4)$$

along with the incompressibility constraint,

$$\nabla \cdot \mathbf{u} = 0, \quad (5)$$

are then solved on a grid.

Here \mathbf{u} is the velocity, p is the pressure, ρ is the density, $\boldsymbol{\tau}$ is the viscous stress tensor, and \mathbf{F}_s is the volume force due to surface tension. Among the approximations that have been used are the MAC scheme [21] and Godunov's method [22]. Fluid properties such as density and viscosity are not advected but are updated once the interfacial points have been moved with the local velocity [1]. The fluid properties are updated by constructing an indicator function $I(\mathbf{x})$ whose value varies only as the interface is crossed. Thus, from the value of the indicator function, all of the properties can be inferred. The indicator function is calculated by solving a Poisson equation,

$$\nabla^2 I = \nabla \cdot \mathbf{G}, \quad (6)$$

where \mathbf{G} is the field created when the unit normals at the interface are interpolated to the grid [1].

3. THE POINT-SET METHOD

3.1. The Indicator Function

In contrast to standard front-tracking, where the indicator function is calculated from the surface normals, we extract information about a surface of unconnected interfacial points by first constructing an indicator function, $I(\mathbf{x})$. As in Unverdi and Tryggvason [1],

the interfacial points are embedded within a computational grid. Given the grid and the location of the interfacial points, we construct an indicator function which labels points, \mathbf{x} , lying inside and outside of the interface to distinguish one phase or material from another. An indicator function for an ideally thin interface would have the following values:

$$I(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is inside the interface,} \\ 0 & \text{if } \mathbf{x} \text{ is outside the interface.} \end{cases} \quad (7)$$

The calculation of the indicator function requires two logically distinct operations. First, an approximate indicator function $I_a(\mathbf{x})$ is generated by solving Laplace's equation on the computational grid,

$$\nabla^2 I_a = 0. \quad (8)$$

Second, the values of this solution are adjusted to match a constant value along the interface.

The solution of Laplace's equation on a grid gives the values of $I_a(\mathbf{x}_g)$ at the centers of grid cells, \mathbf{x}_g . The role of $I_a(\mathbf{x})$ is to distinguish interior and exterior grid cells. In the simplest case, the interface does not intersect the computational boundaries, and boundary conditions for the solution of Laplace's equation are specified as follows: The grid boundary is outside the interface, and it is sufficiently accurate in this simple example to specify $I_a = 0$ at cell centers adjacent to the boundary. In cells through which interfacial points pass, we set $I_a = 1$. Figure 1 shows a circular interface and how the boundary conditions are set. (If the interface does intersect the computational boundaries, one must also enforce $I_a = 1$ on boundaries which enclose the interface.) Laplace's equation is then solved everywhere on the grid except in those cells where the value of I_a is prescribed by the boundary conditions.

The solution to Laplace's equation within a closed boundary with a constant value on the boundary will be a constant. Thus, interior to the interface, the solution will be one. Exterior to the interface, the solution will decrease smoothly from one near the interface to

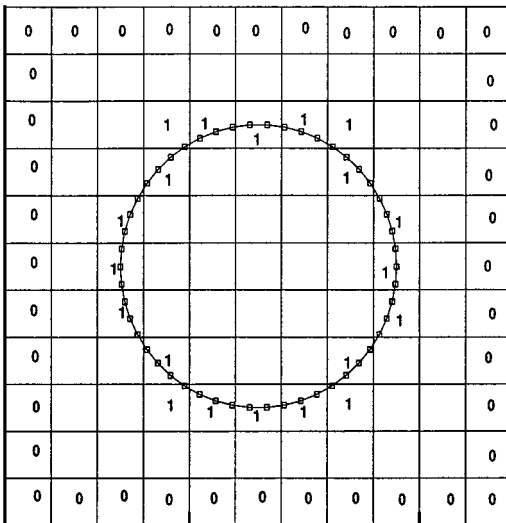


FIG. 1. The value of the approximate indicator function, I_a , is prescribed to be equal to 0 in cells adjacent to the boundary and 1 in cells which contain an interface point.

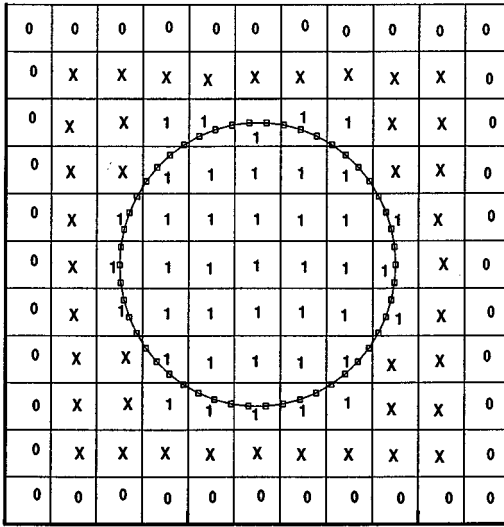


FIG. 2. Solving Laplace's equation with the prescribed values shown in Fig. 1 results in an indicator function equal to 1 at grid points interior to the interface and between 0 and 1, denoted by X, between the interface and the boundary.

zero near the grid boundary. Figure 2 shows the solution that arises from solving Laplace's equation using the boundary conditions in Fig. 1. In Fig. 2, X denotes a value of I_a between 0 and 1.

The numerical solution of Laplace's equation is then adjusted. I_a is reset to zero wherever $I_a < 1 - \epsilon$, where ϵ is some small parameter. The effect of this adjustment is to set I_a to zero in most of those cells labeled X in Fig. 2. This solves the problem of distinguishing inside from outside, without the aid of connectivity.

We define an interior cell as a cell whose approximate indicator value is $I_a = 1$ and that is surrounded by $I_a = 1$ cells. Similarly, we define an exterior cell as an $I_a = 0$ cell that is surrounded by $I_a = 0$ cells. Typically, the interface will not cross an entire cell in a time step, and interior and exterior cells remain interior and exterior cells from one time step to the next. For all but the first cycle, interior and exterior I_a values can be fixed when Laplace's equation is solved. This allows the iterative solver used for Laplace's equation to iterate only over cells that are near the interface. Thus the solution of Laplace's equation for the approximate indicator function adds little to the computational work required in a cycle.

The solution of Laplace's equation gives I_a at cell centers. A continuous and smoothed \tilde{I}_a is defined between cell centers using B-spline interpolation functions,

$$\tilde{I}_a(\mathbf{x}) = \sum_g I_a(\mathbf{x}_g) S(\mathbf{x} - \mathbf{x}_g), \quad (9)$$

where $S(\mathbf{x} - \mathbf{x}_g)$ is a tensor product of one-dimensional B-splines, M , given by

$$S(\mathbf{x} - \mathbf{x}_g) = M(x - x_g; \Delta x) M(y - y_g; \Delta y) M(z - z_g; \Delta z) \quad (10)$$

on a grid with grid spacings Δx , Δy , and Δz . The support of the B-splines is a small

multiple of the mesh spacing in each coordinate direction. See [23] for a description of B-splines. The cubic B-spline $M_3(x; h)$ and quintic B-spline $M_5(x; h)$ are

$$M_3(x; h) = \begin{cases} \frac{2}{3} - \left(\frac{|x|}{h}\right)^2 + \frac{1}{2}\left(\frac{|x|}{h}\right)^3, & 0 \leq \frac{|x|}{h} \leq 1, \\ \frac{1}{6}\left(2 - \frac{|x|}{h}\right)^3, & 1 < \frac{|x|}{h} \leq 2, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

and

$$M_5(x; h) = \frac{1}{120} \begin{cases} \left(3 - \frac{|x|}{h}\right)^5 - 6\left(2 - \frac{|x|}{h}\right)^5 + 15\left(1 - \frac{|x|}{h}\right)^5, & 0 \leq \frac{|x|}{h} \leq 1, \\ \left(3 - \frac{|x|}{h}\right)^5 - 6\left(2 - \frac{|x|}{h}\right)^5, & 1 < \frac{|x|}{h} \leq 2, \\ \left(3 - \frac{|x|}{h}\right)^5, & 2 < \frac{|x|}{h} \leq 3, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

The approximate indicator function $\tilde{I}_a = 1$ interior to the interface, and $\tilde{I}_a = 0$ exterior to the interface. Convolution of I_a with the B-splines causes \tilde{I}_a to vary smoothly between 1 and 0 near the interface and yields an interfacial transition region of thickness h .

The second step in the construction of the indicator function is to calculate a correction to \tilde{I}_a such that the value of the indicator function is constant along the interface. That is, corrections, δI_p , are made to each point on the interface so that contours of I coincide with the interface, where $I = I_c$. The corrected indicator function is defined by

$$I(\mathbf{x}) = \sum_{p=1}^{N_p} \delta I_p S(\mathbf{x} - \mathbf{x}_p) + \tilde{I}_a(\mathbf{x}), \quad (13)$$

where N_p is the number of interfacial points and \mathbf{x}_p are the interfacial positions. The coefficients δI_p are calculated by solving Eq. (14) at every point along the interface:

$$I(\mathbf{x}_p) = I_c. \quad (14)$$

This results in a system of linear equations, written

$$\sum_{p'=1}^{N_p} \delta I_{p'} S(\mathbf{x}_p - \mathbf{x}_{p'}) = I_c - \tilde{I}_a(\mathbf{x}_p). \quad (15)$$

We choose I_c to be the average value of \tilde{I}_a on the interface,

$$I_c = \frac{\sum_{p=1}^{N_p} \tilde{I}_a(\mathbf{x}_p)}{N_p}.$$

The idea is similar to York's mass matrix method [17, 18]. However, unlike the mass matrix method, interface data is interpolated directly from interface points to interface points, rather than from interface points to grid points, and then from grid points to interface points. The resulting linear system is larger, but it is not rank deficient, it does not require a

least-squares solution, and it does not require partial lumping to avoid singularity [24]. The banded symmetric linear system which arises is solved using a conjugate gradient iteration.

One can also account for interfaces embedded within interfaces by using an indicator function which can take on values other than 0 or 1.

3.2. Surface Normals and Curvatures

We compute the normal vectors, \mathbf{n}_p , at the point locations by differentiating the indicator function at the interface point:

$$\mathbf{n}_p = -\nabla I(\mathbf{x})|_{\mathbf{x}_p}. \quad (16)$$

The unit normals are

$$\hat{\mathbf{n}}_p = \frac{\mathbf{n}_p}{|\mathbf{n}_p|}. \quad (17)$$

The gradient in (16) is computed by differentiating the B-splines in (13) analytically.

The curvature at each interfacial point is calculated by differentiating the unit normals [25]:

$$\kappa_p = -\nabla \cdot \hat{\mathbf{n}}_p = -\left(\nabla \cdot \frac{\nabla I(\mathbf{x})}{|\nabla I(\mathbf{x})|} \right) \Big|_{\mathbf{x}_p}. \quad (18)$$

Once again, the B-splines are differentiated analytically. Since any calculation of curvature requires two derivatives, the B-spline used should be at least twice continuously differentiable. The lowest order spline which is twice continuously differentiable is the cubic spline. The level-set method calculates the surface normal and curvature in the same way [26].

3.3. Surface Area

The description of interfacial dynamics often requires the calculation of surface tension. The surface tension volume force has the form

$$\mathbf{F}_s = \sigma \kappa \delta(d) \hat{\mathbf{n}}, \quad (19)$$

where δ is the Dirac delta function, d is the normal distance to the interfacial surface, and $\hat{\mathbf{n}}$ is the unit normal to the interface [26].

We follow [27] and [1] in using (3) to transfer the surface tension forces calculated on the interface to a grid with grid spacings Δx , Δy , and Δz . Our objective is to show that the point-set method can be integrated into the standard front-tracking algorithm.

The calculation of the surface tension force requires integration over some volume, which results in a value for s_p , the arclength (2D) or surface area (3D) associated with the interfacial point. The arclength or surface area multiplies the force per area ($\sigma \kappa \hat{\mathbf{n}}$) to give the force at the interface point. The surface tension force $s_p \sigma \kappa \hat{\mathbf{n}}$ is interpolated to the grid using

$$\frac{\mathbf{F}_s}{\rho} \Big|_{\mathbf{x}_g} = \sum_p s_p \frac{\sigma \kappa \hat{\mathbf{n}}}{\langle \rho \rangle} \Big|_{\mathbf{x}_p} \tilde{\delta}(\mathbf{x}_g - \mathbf{x}_p), \quad (20)$$

where $\langle \rho \rangle$ is the average density across the interface and $\tilde{\delta}(\mathbf{x}_g - \mathbf{x}_p)$ denotes a tensor product of one-dimensional approximate delta functions, e.g., in 3D,

$$\tilde{\delta}(\mathbf{x}_g - \mathbf{x}_p) = \frac{S(\mathbf{x}_g - \mathbf{x}_p)}{\Delta x \Delta y \Delta z},$$

where $S(\mathbf{x}_g - \mathbf{x}_p)$ is defined by (10). The B-spline $(1/h_g)M(d; h_g)$, where h_g determines the support of M , can serve as an approximate delta function since its integral is 1, its maximum increases without bound as $h_g \rightarrow 0$, it has compact support, and it is a monotonically decreasing function of increasing d . (Note that the approximate delta function is finite when $\mathbf{x}_g = \mathbf{x}_p$.)

To accomplish this interpolation, the surface area or arclength (s_p) of each individual interfacial point must be determined. Following Peskin's [27] definition of a boundary integral, we define the surface area of a point by constructing a circular element of area around each point on the surface and dividing each circular area (possibly overlapping) by the number of interfacial points ($N < N_p$) whose distance to the center is less than a circle with radius h_p .

$$s_p \approx \frac{\text{Area}}{N}. \quad (21)$$

Since a differentiable surface $H(\mathbf{x}) = 0$ can be approximated by a plane in the neighborhood of a point \mathbf{x}_p ,

$$H(\mathbf{x}) = 0 \approx H(\mathbf{x}_p) + \nabla H|_{\mathbf{x}_p} \cdot (\mathbf{x} - \mathbf{x}_p) = 0,$$

such a definition can be justified if the size of the circle is small compared with the radius of curvature.

However, if the number of points contained within a circle of radius h_p is small, small variations in h_p may cause s_p to change discontinuously. A smoother approximation of the area results if the zeroth-order B-spline in the expression, which gives equal weight to all points, is replaced by a higher order B-spline, which gives smaller weights to points further away from \mathbf{x}_p than to nearby points. For this reason, we adopt an integral formulation. The integral of a positive function $f(\mathbf{x})$ can be approximated in a Monte Carlo sense by

$$\iint_{\text{Area}} f(\mathbf{x}) dA = \frac{\text{Area}}{N} \sum_{i=1}^N f(\mathbf{x}_i),$$

which with (21) allows us to write

$$s_p \approx \frac{\iint_{\text{Area}} f(\mathbf{x}) dA}{\sum_{i=1}^N f(\mathbf{x}_i)}.$$

Choosing $f(\mathbf{x})$ to be a B-spline $M(x; h_p)$, its argument to be the distance from \mathbf{x}_p , and the integration bounds to be the circular support of $M(x; h_p)$, one has in 2D and 3D respectively,

$$s_p \approx \frac{h_p}{\sum_{i=1}^{N_p} M(|\mathbf{x}_i - \mathbf{x}_p|; h_p)} \quad (2D), \quad (22)$$

$$s_p \approx \frac{2\pi h_p^2 \int_0^\infty M(r) r dr}{\sum_{i=1}^{N_p} M(|\mathbf{x}_i - \mathbf{x}_p|; h_p)} \quad (3D), \quad (23)$$

where now while all interfacial points, N_p , are included in the summation, the only nonzero contributions will come from points in the support of M . The summations can be arranged to span only nearby interfacial points by binning the interfacial points. Note that these equations are consistent with Eq. (21), which defines the surface area in an average sense.

We note that one can avoid computing the surface areas of interfacial points by returning to (19) and approximating $\hat{n}\delta(d)$ with

$$\frac{\nabla I(\mathbf{x})}{[I]},$$

where $[I]$ is the jump in I across the interface, following the continuum surface force (CSF) formulation [25].

4. NUMERICAL RESULTS

4.1. Tests of Geometric Accuracy

Table I shows the maximum relative percent errors in the unit normals, curvatures, arclengths of individual points, and total arclength for an ellipse

$$\frac{x^2}{9} + \frac{y^2}{4} = 1$$

computed with quintic and cubic B-splines. We observe that although the cubic B-splines are twice continuously differentiable, the added smoothness of the quintic B-splines reduces the errors significantly. The interfacial points have been distributed regularly according to

$$x = 3 \cos(\theta_i), \quad y = 2 \sin(\theta_i), \quad \text{where } \theta_i = \frac{2\pi i}{N_p}, \quad i = 1, N_p, \quad (24)$$

where $N_p = 100$. Tests were performed on a $[-6, 6]^2$ domain with a 32×32 grid. Individual arclengths were computed by running the arclength computation twice, once with equal h_p for all interfacial points, and subsequently with $h_p = s_p$. Exact individual arclengths were taken to be the average of the straight line segments between two neighboring points.

TABLE I
Maximum Relative Percent Error to Ellipse $\frac{x^2}{9} + \frac{y^2}{4} = 1$

1.9 points per cell	Cubic (%)	Quintic (%)
Unit normal	3.9×10^{-1}	1.5×10^{-2}
Curvature	34	5.4×10^{-1}
Arclength	2.5×10^{-1}	3.4×10^{-1}
Total arclength	9.3×10^{-3}	8.1×10^{-3}

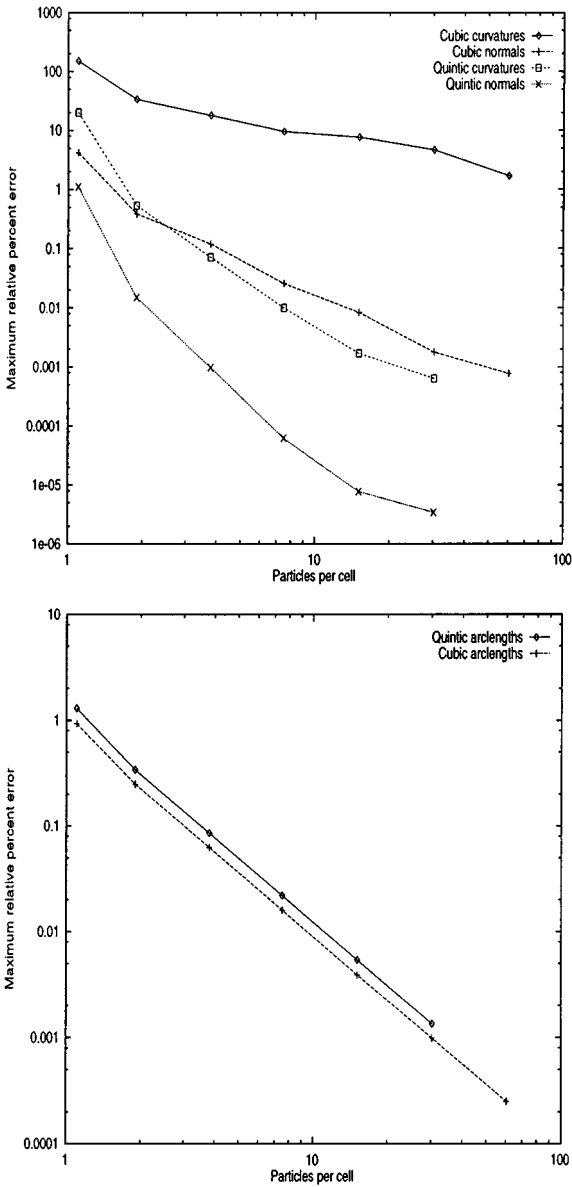


FIG. 3. The errors in the calculation of curvatures, normals, and arclengths for an ellipse decrease with an increasing density of points on the interface and are lower for quintic B-splines than for cubic ones.

Figure 3 shows the maximum relative percent errors in curvatures, normals, and arclengths using quintic and cubic splines plotted as a function of the number of points per cell. The total number of points used ranged from 50 to 3200 or 1.1 to 60.4 points per cell. Both axes are plotted logarithmically. The convergence appears to be polynomial. If the errors are assumed to decay as n_{cell}^{-p} where n_{cell} is the number of interfacial points per cell, we calculate p with a least-squares fit to the lines in Fig. 3. For cubic B-splines, $p = 2.0$ for arclengths, $p = 2.1$ for normals, and $p = .96$ for curvatures. For quintic B-splines, $p = 2.0$ for arclengths, $p = 3.8$ for normals, and $p = 3.0$ for curvatures.

TABLE II
Maximum Relative Percent Error to $\frac{x^2}{9} + \frac{y^2}{4} = 1$ with Randomly Distributed Points (Quintic Splines)

Number points (# pts/cell)	50 (1.3)	100 (2.1)	200 (3.8)	400 (7.7)
Unit normal	13	2.7×10^{-2}	3.4×10^{-3}	2.0×10^{-4}
Curvature	530	2.2	3.5×10^{-1}	5.0×10^{-2}
Total arclength	16	2.5	3.2×10^{-1}	7.5×10^{-2}

Table II shows the maximum relative percent errors when the points have been randomly distributed according to

$$x = 3 \cos(\theta_i), \quad y = 2 \sin(\theta_i), \quad \text{where } \theta_i = \frac{2\pi i}{N_p} \pm \frac{2\xi\pi}{N_p} \quad i = 1, N_p, \quad (25)$$

where ξ is a random number between 0 and .5. Random distributions reflect more accurately what is likely to be encountered in a real calculation, where any initial ordering will be lost as a result of the distortion of the interface. In Table II, the arclength computation was computed with fixed $h_p = .2$. The total arclength converges, as does the arclength of any fixed segment of the curve, as the number of interfacial points is increased. Good accuracy is achieved with two interfacial particles per grid cell.

We also performed calculations with a nonconvex “starfish,” the results of which are displayed in Table III (for 50, 100, 200, and 400 points) and Figs. 4 and 5 (for 100 points). The points on the starfish are distributed according to

$$x = \cos(\theta_i) + .4 \sin(5\theta_i) \cos(\theta_i), \quad y = \sin(\theta_i) + .4 \sin(5\theta_i) \sin(\theta_i), \quad \theta_i = \frac{2\pi i}{N_p}. \quad (26)$$

Calculations were also performed with a square with 100 points as shown in Figs. 4 and 5. Considering interfacial points four points removed from the corner point, the maximum relative percent error was $9.0 \times 10^{-2}\%$ for the unit normals and the largest curvature was 6.6×10^{-3} (it should be identically zero).

Table IV and Fig. 6 show the results of a 3D calculation for the ellipsoid, $x^2/16 + y^2/9 + z^2/4 = 1$, with quintic splines. Interfacial points were distributed on the ellipsoid by iteratively subdividing and projecting the faces of an inscribed polyhedron, starting with the equilateral triangular faces of a 20-sided icosahedron [28]. The surface area was computed with $h_p = 3\sqrt{\text{surfacearea}/(N_p\pi)}$, which is 3 times the radius of an average interfacial element. Calculations were performed with a 17^3 grid on a $[-10, 10]^3$ domain.

TABLE III
Maximum Relative Percent Error to Starfish (Quintic Splines)

Number points (# pts/cell)	50 (1.6)	100 (2.8)	200 (5.6)	400 (11.1)
Unit normal	29	3.2×10^{-1}	6.8×10^{-2}	1.2×10^{-2}
Curvature	270	5.7	2.3	1.2
Arclength	30	19	9.0	2.9
Total arclength	2.0	3.1×10^{-1}	9.4×10^{-2}	2.6×10^{-2}

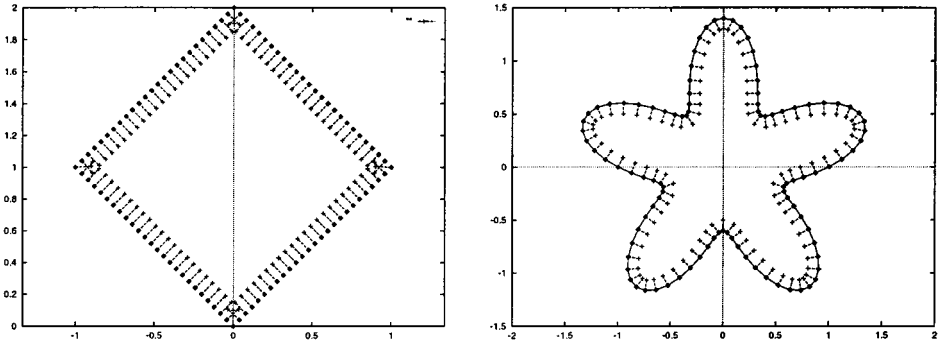


FIG. 4. The computed normals for a square and a starfish-shaped interface are plotted. The normals are well behaved, even where the curvature is high.

Again, the convergence appears to be polynomial. If the errors are assumed to decay as n_{cell}^{-p} , where n_{cell} is the number of interfacial points per cell, $p = .68$ for quintic arclengths, $p = 2.3$ for quintic normals, and $p = 1.9$ for quintic curvatures.

4.2. Droplet Oscillations

We solve the incompressible Navier–Stokes equations (4), (5) on the fixed grid using a MAC [21] scheme. As do Unverdi and Tryggvason [1], we update density and viscosity using the indicator function. The surface tension force is computed using (20) where the interfacial curvatures and normals are computed using the formulas in Section 3.2, and the surface areas are computed using (22) and (23).

In our simulations of droplet oscillations, density acquires a constant (although different) value inside and outside the interface. As such, we use the Boussinesq approximation

$$\nabla p \times \nabla \rho = 0. \tag{27}$$

The relation holds outside the interfacial region since $\nabla \rho = 0$. Inside the interfacial region, using the continuity of stresses at a fluid boundary, one has

$$[-p + \tau]\hat{n} = -\sigma\kappa\hat{n},$$

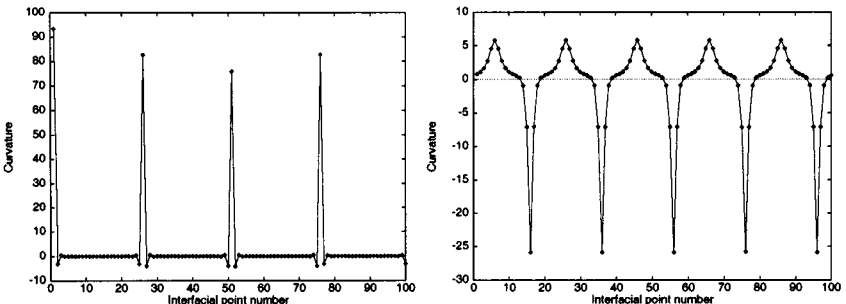


FIG. 5. The computed curvatures for a square interface exhibit large but bounded variation as one traverses the interface, even at the corners of the square (left panel). Computed curvatures for starfish-shaped interface vary dramatically, yet are good approximations at 2.8 points per cell as Table III shows.

TABLE IV
Maximum Relative Percent Errors to Ellipsoid $\frac{x^2}{16} + \frac{y^2}{9} + \frac{z^2}{4} = 1$

Number points (# pts/cell)	42 (1.0)	162 (1.8)	642 (6.7)	2562 (25.6)
Unit normal	200	4.4×10^{-1}	6.6×10^{-2}	9.1×10^{-4}
Curvature	490	2.1	3.0×10^{-1}	1.2×10^{-2}
Total surface area	7	3.3	1.3	5.3×10^{-1}

where the brackets denote the jump across the interface. In the interfacial zone, the gradient of p should be aligned (or nearly aligned) with the gradient of ρ for the specific numerical simulations we perform. A resolved numerical experiment (on a 128×128 mesh) for a simple 2D oscillation shows small differences between when the Boussinesq approximation (27) was and was not used.

Thus, $\frac{\nabla p}{\rho}$ behaves like a potential $\nabla\phi$ since

$$\nabla \times \frac{\nabla p}{\rho} = \frac{\nabla \times \nabla p}{\rho} + \frac{\nabla p \times \nabla \rho}{\rho^2} = 0.$$

Using $\nabla\phi$ instead of $\frac{\nabla p}{\rho}$ allows us to use fast Poisson solvers (we use the 2D and 3D cyclic reduction routines in FISHPACK), instead of more expensive iterative solvers such as the Incomplete Cholesky Conjugate Gradient (ICCG) method when (4), (5) are solved.

In our simulations of droplet oscillation, we employ an alternate projection method described in Appendix 1, which significantly reduces the parasitic current in the 2D and 3D droplet oscillations. The parasitic current is a spurious velocity flow which occurs at an interface with nonzero curvature. It becomes most conspicuous as a droplet approaches equilibrium. (See Fig. 15.)

We first perform theoretical comparisons with an oscillating droplet. A 2D droplet with a radius perturbed according to

$$r = r_0 + \alpha \cos(n\theta)$$

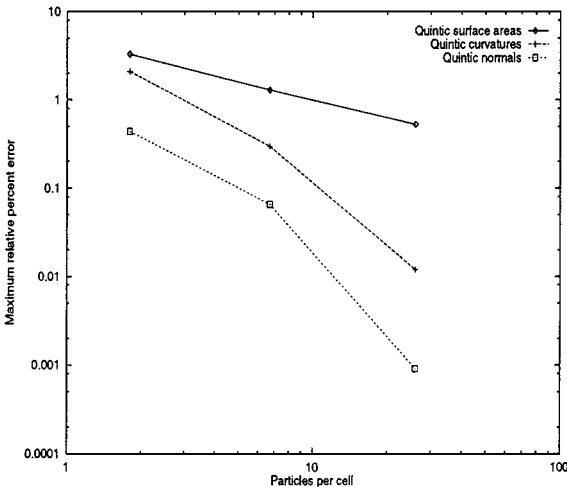


FIG. 6. The errors in computed values of the curvatures, normals, and surface areas for an ellipsoid on a 17^3 3D grid converge with increasing particle density.

should have a frequency of oscillation given by

$$\omega_n^2 = \frac{(n^3 - n)\sigma}{(\rho_d + \rho_e)r_0^3}$$

as shown by Fyfe *et al.* [29] where ρ_d and ρ_e are the density, interior and exterior to the droplet respectively. For $n = 2$, $\sigma = 1$, $\alpha = .01$, $\rho_d = 1$, $\rho_e = .01$, and $r_0 = 2$, in a $[-10, 10]^2$ doubly periodic computational domain with 400 interfacial points, the frequency is 13.2% slower in a 64^2 grid, 6.1% slower in a 128^2 grid, and 1.5% slower in a 256^2 grid. Tryggvasson *et al.* [20] are able to approach the theoretical frequency to within 1.9% with a 128^2 grid under similar conditions.

Figure 7 shows unconnected and connected 2D calculations showing plots of kinetic energy $\frac{1}{2} \int \rho \mathbf{u} \cdot \mathbf{u} dV$, performed with a MAC scheme on a doubly periodic 64×64 mesh with $\sigma = 1$. The interface is initially the ellipse $x^2/9 + y^2/4 = 1$ at rest. The number of interfacial points placed on the ellipse varies from 100 for the connected case to 150 for the unconnected case. The densities (viscosities) inside and outside the ellipse are 1. (.01) and .01 (5×10^{-5}) respectively. Particles are periodically redistributed on the interface using Legendre interpolation when interfacial points are connected and periodically regenerated (Section 4.4) when interfacial points are not connected.

Figure 8 shows unconnected and connected 3D calculations performed with a MAC scheme [21] on a triply periodic $[-10, 10]^3$, 32^3 mesh with $\sigma = .2$, and 642 interfacial points placed on the ellipsoid $x^2/16 + y^2/12.25 + z^2/14.44 = 1$. The geometry of the interface is calculated on a 16^3 mesh for the unconnected case. Points are not regenerated or redistributed in either the connected or unconnected case. The densities (viscosities) inside and outside the ellipse were 1. (.05) and .01 (1×10^{-4}) respectively. In the connected case, the normals are computed using York's mass matrix scheme, and the curvatures are calculated using (2).

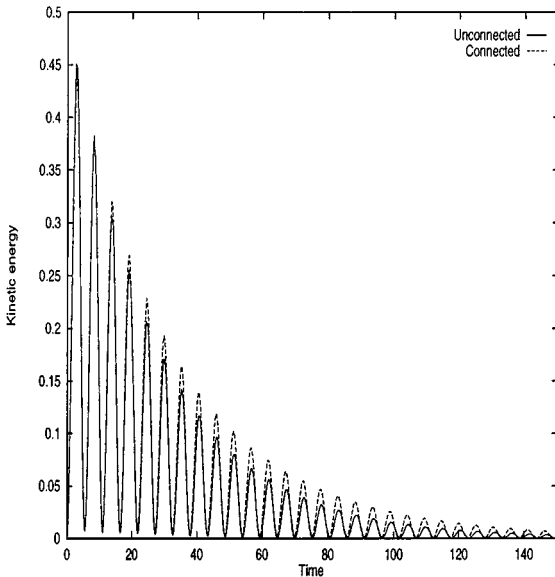


FIG. 7. The kinetic energy histories of an oscillating cylinder on a 64×64 grid in 2D with connected (dashed curve) and unconnected (solid curve) interfacial points are compared. The kinetic energy decreases slightly more rapidly with unconnected points, but the frequency of the oscillations is the same for both calculations.

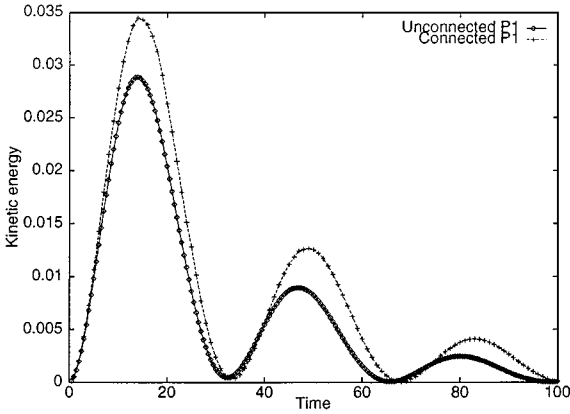


FIG. 8. The kinetic energy histories of an oscillating ellipsoidal drop in 3D on a 32^3 grid with 642 interfacial points are compared for connected (dashed curve) and unconnected (solid curve) points. As in 2D, the decay of kinetic energy is more rapid with unconnected points.

Both 2D schemes agree well in matching the droplet oscillation period. The 2D connected scheme has a slightly higher kinetic energy peak. If we assume the peaks of the oscillation decay as $e^{-\alpha t}$, then $\alpha = .0306$ for the connected case and $\alpha = .0358$ for the unconnected case. The differences in the 3D schemes are more conspicuous. We attribute the main difference to possible inaccuracies in the surface area calculation in the unconnected case. Figure 6 shows that in 3D, the surface area calculation is the least accurate computed attribute.

4.3. Coalescence

Figures 9 and 10 show a coalescence of two cylindrical droplets in 2D. The simulations begin with interfacial points distributed about two unit circles placed in close proximity

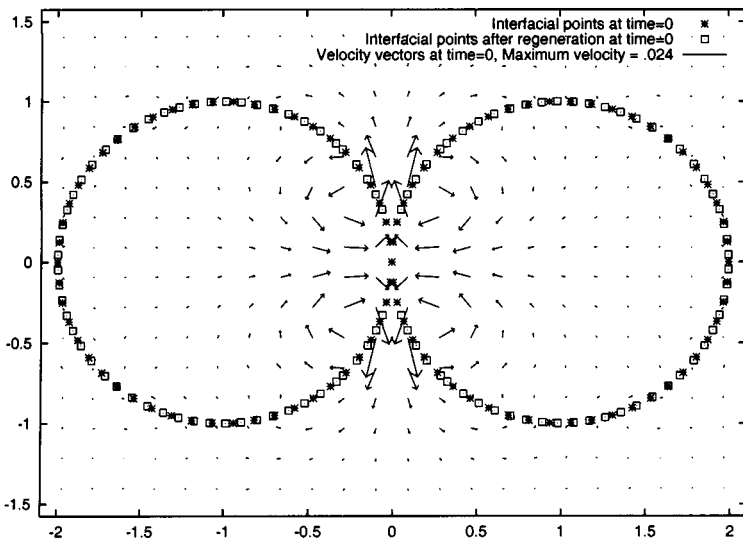


FIG. 9. Initial distribution of a 100 interfacial points shows two undeformed cylinders in close contact. After regeneration on a 64×64 grid, interfacial points near the point of contact are deleted. Velocity vectors generated from the surface tension forces are shown for the regenerated interface.

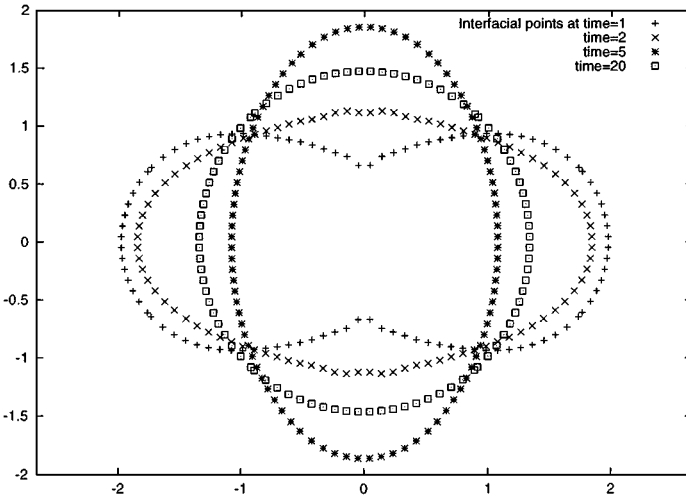


FIG. 10. Coalescence (continued) of two cylinders in Fig. 9 is calculated with the point-set method with $\sigma = .5$ on a 64×64 grid and 90–100 points on the interface. Surface tension drives their coalescence and deformation into an ellipse which eventually relaxes into a circle.

to each other. The 2D calculation is performed on a $[-6, 6]^2$ domain with a 64×64 grid and $\sigma = .5$. The densities (viscosities) inside and outside the ellipse are 1. (.05) and .01 (2.5×10^{-4}) respectively. Figure 10 shows the actual interfacial points at different times. Figure 12 shows a 3D calculation (where isosurfaces of the droplet are shown) of coalescence run on a $[-4, 4]^3$ domain with a 32^3 grid. Initially interfacial points are distributed about two unit spheres placed in close proximity. The geometry is computed on a 16^3 grid with $\sigma = .2$. The densities (viscosities) inside and outside the ellipse are 1. (.05) and .01 (1×10^{-4}) respectively. Approximately 90–100 points are used for the 2D interface and fewer than 500 points are used for the 3D interface. The number of points used on the interface varies due to regeneration. Figure 11 shows a 2D calculation where a specified velocity field is chosen to cause an initial circular interface to break up.

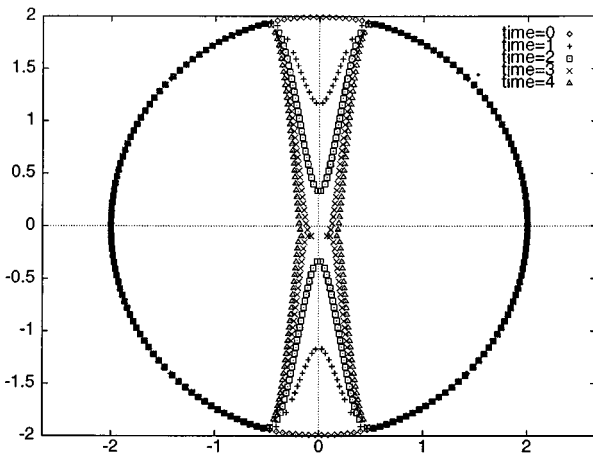


FIG. 11. Breakup is shown of a cylinder in 2D with the velocity prescribed. Points on the interface are shown at equally spaced intervals in time. The dark points on the undisturbed interface represent multiple times.

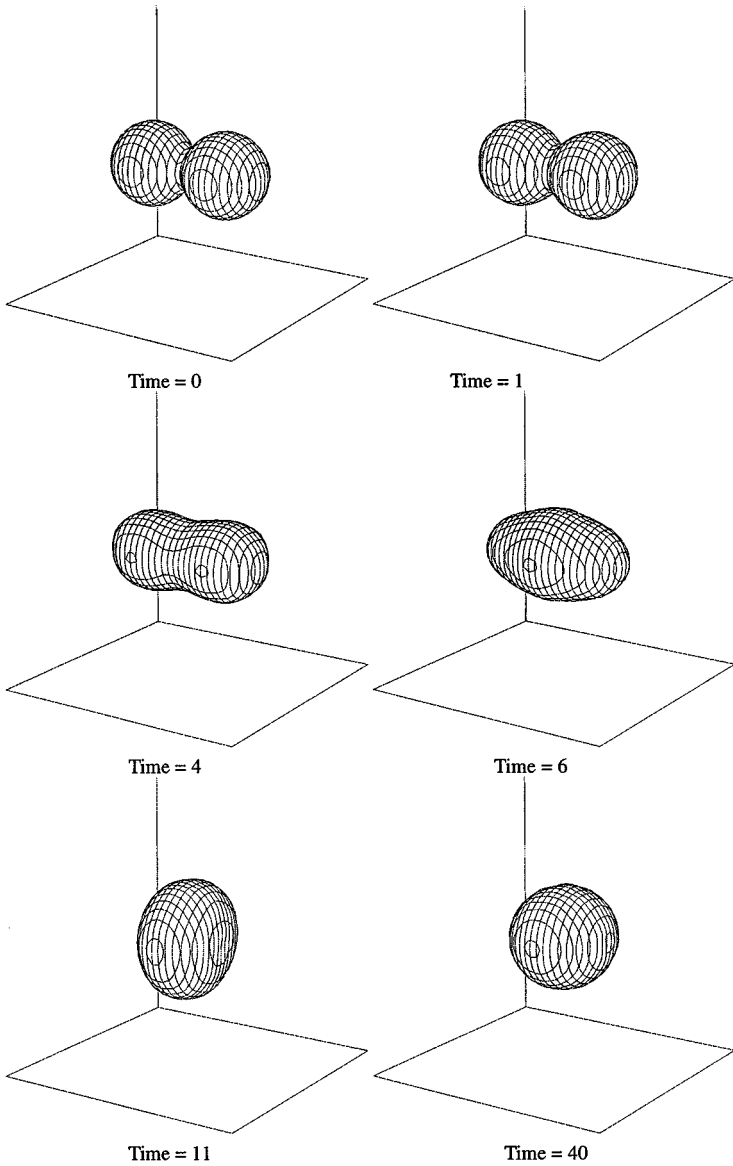


FIG. 12. Two spherical drops coalesce in 3D on a 32^3 grid with $\sigma = .2$. Isosurfaces of the indicator function are plotted.

4.4. Regenerating Interfacial Points

Periodically, points are regenerated on the interfaces as in, for example, Figs. 7, 9, 10, 11, and 12. We note that regeneration is required for regularization and to maintain an acceptable number of interface points per grid cell. The indicator function $I(\mathbf{x})$ is constructed on the computational grid with the existing interfacial points. New interfacial points are then generated by embedding the indicator function within a finer grid. The finer grid is created by subdividing cells of the computational grid as shown in Fig. 13, where bold lines mark the boundaries of the actual computational mesh and lighter lines mark the boundaries of the finer grid. Cell centers of the fine grid (e.g., point A) are projected onto the interface

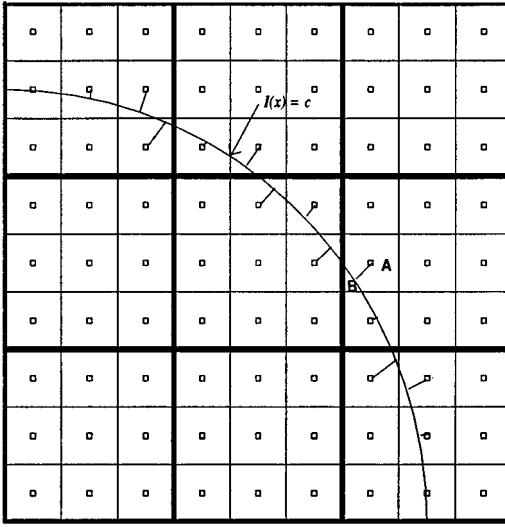


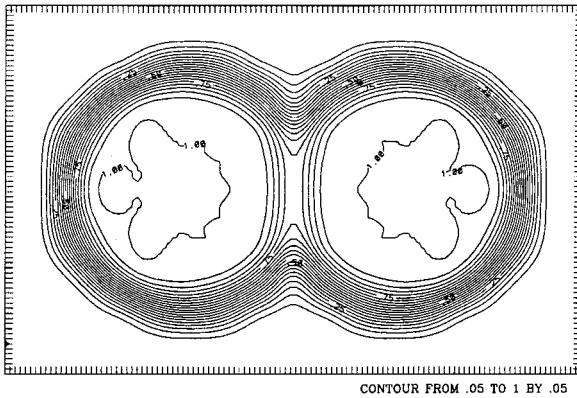
FIG. 13. A new point is added to the interface if a cell center and its projection onto the interface lie in the same fine cell. Bold lines mark the boundaries of the actual computational mesh and lighter lines mark the boundaries of the finer grid. Straight line segments between cell centers on the fine grid and the interface mark cells in which new interface points have been created, e.g., AB . Computation of the projection uses $I(\mathbf{x})$.

(e.g., point B) using a one-dimensional Newton iteration, searching either in the normal direction, $\nabla I(\mathbf{x})$, or in one of the coordinate directions. One knows that the interface has been located when the indicator function is equal to I_c , the interfacial value, Eq. (14).

If point B lies within the same fine cell as the cell center A, point B is retained as an interfacial point. All points (e.g., point A) whose projection onto the interface (e.g., point B) have been retained as interfacial points are indicated in Fig. 13 by a line drawn between fine cell centers and corresponding points on the interface. The same operations are performed whether coalescence occurs or not.

The regeneration scheme can preselect cell-centered points of the fine grid on the outside of the interface, $I_c - \epsilon < I(\mathbf{x}) < I_c$, or on the inside of the interface, $I_c + \epsilon > I(\mathbf{x}) > I_c$, as candidates for this regeneration process, where say $\epsilon \approx 0.2I_c$.

Regenerating points from the outside allows interfacial points at a contact surface to be deleted. (Regenerating points from the inside of the interface in Fig. 9 recreates both circles and would not allow points near the point of contact to be deleted.) For example, for coalescence to occur in Figs. 9 and 12, one must not regenerate points near the point of contact between the two spheres or circles. This can be enforced by regenerating points from the outside of the two interfaces. Figure 14 shows contours of I generated by two circles in close proximity. The contours are densely packed away from the point of contact and disperse ($|\nabla I|$ small) near the point of contact. The distance of a contour line of $I(\mathbf{x}) = I_c^* < I_c$ from the contour line $I(\mathbf{x}) = I_c$ would vary, growing relatively large near the point of contact. Thus, if one were to select points on the fine grid to project onto the interface, one could preselect points whose indicator value satisfies say $I_c - \epsilon < I(\mathbf{x}) < I_c^*$, and points near the contact surface would not fall into this category. If a cell-centered point on the fine grid did project itself onto the point of contact, the distance traveled would be too large for the cell center (A) and the projection location (B) to be in the same fine cell. Thus, points near the point of contact will not be regenerated. Subsequent construction of the



CONTOUR FROM .05 TO 1 BY .05

FIG. 14. Contours of the indicator function, I , for two cylinders in close contact are plotted. The variation in I near the contact line is small.

indicator function $I(\mathbf{x})$ will smooth the transition where the interfacial points were deleted. Such an effect is realized in Fig. 9. The plot shows the initial placement of interfacial points at time = 0 and the placement of the interfacial points after regeneration, still at time = 0, with the incompressible velocity field generated from the regenerated points.

We also successfully performed a numerical experiment with quintic splines where the distance between two circles is controlled and successively decreased. Initially both circles are regenerated, but at a sufficiently small distance, coalescence occurs.

Interfaces also can undergo breakup. Regenerating points from the inside of the two interfaces is compatible with breakup, because it allows points to be added as contact diminishes. Inside regeneration is employed in Fig. 11 where one begins with a circle and a specified velocity field is chosen to cause breakup.

When one does not know whether breakup or coalescence will occur, one must regenerate points both from the inside and outside of the interface, using the points from previous regeneration for the second regeneration. We have successfully regenerated smooth interfaces and even the square interface in Fig. 4 using the regeneration scheme. However, we assume our means of constructing the indicator function $I(\mathbf{x})$ via (8)–(13) only achieves the interfacial constant value I_c (14) at the interface as defined by the interfacial points. Since any smooth interface can be locally approximated by a plane, this is a valid assumption. However, we do not specifically impose $I(\mathbf{x}) \neq I_c$ off the interface. An alternate construction of $I(\mathbf{x})$ which couples the solution of the approximate indicator function and the correction step should avoid this potential problem if the interface is sufficiently resolved, but it also requires that one solve a larger linear system.

5. CONCLUSION

We have demonstrated that the normals, curvatures, and surface areas can be extracted from unconnected points on a surface using the point-set method, and we have used this information to perform front-tracking computations without connectivity. These computations include an oscillating ellipse and ellipsoid, a breakup driven by a specified velocity, and coalescence in 2D and 3D. We rely on a regeneration scheme to redistribute interfacial points.

The point-set method costs more than standard front-tracking. One must solve a linear system every time step to refine the indicator function (15). For a reasonable number of

interfacial points and grid dimensions, the oscillating ellipse problem in 2D with a pressure Poisson Incomplete Cholesky Conjugate Gradient solver runs two to three times slower for the unconnected case. However, one can minimize the additional cost by using only 2–3 interfacial points per cell, with which normal and curvatures can be computed with reasonable accuracy.

In the Appendix, we also demonstrate a projection technique for solving incompressible flow problems which suppresses parasitic currents in point-set and connected front-tracking computations. The benefits of such a projection scheme are quite dramatic. One does pay a penalty in 3D however, in that three elliptic equations need to be solved to project the velocity. Only one elliptic solver needs to be solved in 3D using a conventional projection technique.

More work is needed to apply these techniques to more complex fluid phenomena involving surface tension, but the results of this study suggest the potential of the point-set method, especially in three dimensions.

APPENDIX: PARASITIC CURRENTS

Surface tension models attempt to account for forces confined to an unresolved thin membrane by using the finite resources of a computational grid. In incompressible flow, the exchange of information from this membrane to the computational grid can lead to the production of parasitic currents. Figure 15 shows the parasitic currents produced when a perfect circle, which should be in equilibrium, is subjected to surface tension on a 32×32 Cartesian grid with 200 interfacial points. The left plot shows the parasitic currents (the actual velocities) in a conventional projection technique and the right plot shows the reduction in the parasitic current (2400 times reduced) using an alternate projection technique. The ratio of kinetic energy to surface energy is 4.5×10^{-8} in the left plot and is 2.0×10^{-16} in the right plot.

Popinet and Zaleski [30] also discuss a means of reducing the parasitic current using a pressure-gradient correction. The difference between our method and Popinet's is that in our

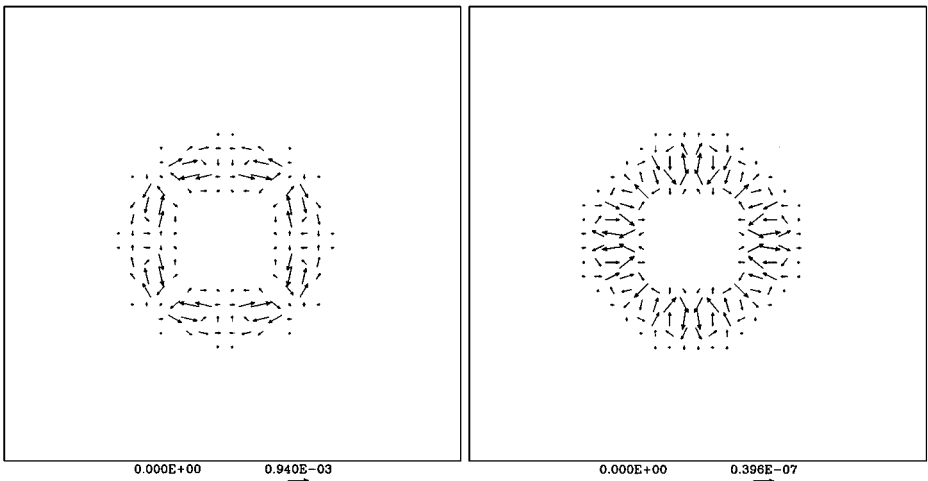


FIG. 15. Parasitic currents are shown for interfacial points distributed on a perfect circle using a standard projection scheme (left plot) and the alternate projection scheme (right plot) described in the Appendix.

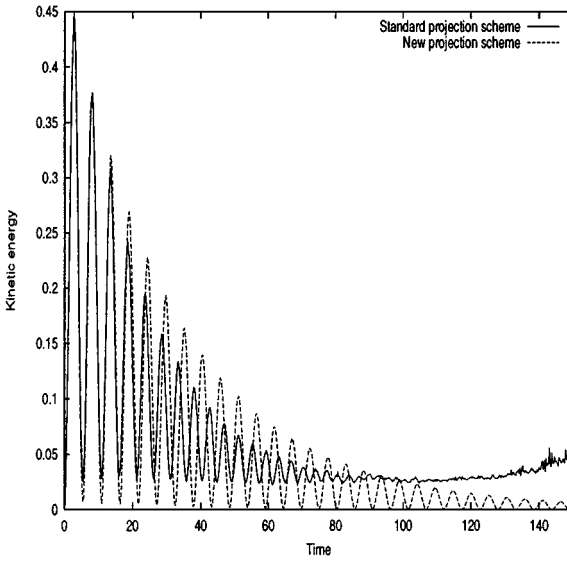


FIG. 16. The kinetic energy histories of an oscillating cylinder on a 64×64 grid in 2D with connected interfacial points are compared using a standard projection scheme and the alternate projection scheme. The kinetic energy decays as it should using the alternate projection scheme, unlike the standard projection scheme which suffers from a visible parasitic current.

method, the magnitude of the parasitic current depends on how well one computes the interfacial curvatures, and thus it is largely independent of the resolution of the computational grid and dependent on the interfacial point resolution.

In Fig. 16, the new projection scheme is compared to the standard projection scheme under the same conditions as Fig. 7. The new projection suppresses the parasitic current quite dramatically compared to the standard projection scheme.

A.1. Projection Scheme

To motivate our discussion, let us begin with the incompressible Navier–Stokes equations with surface tension forces (4), (5), repeated here again for clarity,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \frac{\mathbf{F}_s}{\rho}, \quad \nabla \cdot \mathbf{u} = 0. \quad (\text{A.1})$$

At equilibrium, a balance between two nonzero terms ($\nabla p/\rho$ and \mathbf{F}_s/ρ) must be maintained, but this is difficult to enforce numerically. Local imbalances in the two forces lead to parasitic currents [31].

Numerically, one normally computes a new velocity field from the surface tension forces and the viscous stress tensor and then projects the velocity field into solenoidal space by adding the pressure term. Specifically, one computes a temporary $\tilde{\mathbf{u}}$, which may not be solenoidal, using

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} = \mathbf{R},$$

where

$$\mathbf{R} = -\mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \frac{\mathbf{F}_s}{\rho}.$$

One then computes a final velocity by adding the pressure term

$$\mathbf{u} = \tilde{\mathbf{u}} - \frac{\nabla p}{\rho}. \quad (\text{A.2})$$

The pressure is computed by taking the divergence of (A.2), assuming $\nabla \cdot \mathbf{u} = 0$:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p \right) = \nabla \cdot \tilde{\mathbf{u}}. \quad (\text{A.3})$$

We propose an alternate way of projecting $\tilde{\mathbf{u}}$ into solenoidal space [32]. Rewrite (A.3) in the form

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p - \tilde{\mathbf{u}} \right) = 0. \quad (\text{A.4})$$

The expression inside the parentheses of (A.4) is solenoidal, i.e.,

$$\tilde{\mathbf{u}} = -\nabla \times \boldsymbol{\lambda} + \frac{\nabla p}{\rho}. \quad (\text{A.5})$$

Multiply (A.5) by ρ , and apply the curl operator

$$\nabla \times \rho \nabla \times \boldsymbol{\lambda} = -\nabla \times (\rho \tilde{\mathbf{u}}). \quad (\text{A.6})$$

Equation (A.6) reduces to

$$-\nabla \cdot \rho \nabla \lambda \hat{k} = -\nabla \times (\rho \tilde{\mathbf{u}}) \quad (\text{A.7})$$

in 2D. Solve the elliptical PDE for λ and set

$$\mathbf{u} = -\nabla \times \boldsymbol{\lambda}.$$

Let us compare the curl formulation with the standard divergence formulation. Denote the pressure Poisson operator (A.3) by \mathcal{L} , and denote the operator in (A.6) by \mathcal{L}^* :

$$\mathcal{L} = \nabla \cdot \left(\frac{1}{\rho} \nabla \right), \quad \mathcal{L}^* = \nabla \times (\rho \nabla \times).$$

The net correction to the velocity field with the divergence formulation is

$$\frac{\partial \mathbf{u}}{\partial t} = \left(\mathbf{R} - \frac{1}{\rho} \nabla (\mathcal{L})^{-1} \nabla \cdot \mathbf{R} \right). \quad (\text{A.8})$$

The net correction with the curl formulation is

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla \times (\mathcal{L}^*)^{-1} \nabla \times (\rho \mathbf{R}). \quad (\text{A.9})$$

To enforce the zero divergence of \mathbf{u} in (A.8),

$$\nabla \cdot \frac{1}{\rho} \nabla(\mathcal{L})^{-1} \phi = \phi$$

must be enforced, for arbitrary ϕ and ρ on a discrete level. To enforce the zero divergence of \mathbf{u} in (A.9), one must enforce the standard vector identity

$$\nabla \cdot \nabla \times \mathbf{v} = 0,$$

for arbitrary \mathbf{v} . This is easily achieved on a rectilinear grid.

Such an idea is not new. Sussman *et al.* [26] use the curl formulation to project $\tilde{\mathbf{u}}$. The vorticity–stream function formulation of the Navier–Stokes equations is essentially a curl projection scheme where ρ is assumed not to be spatially varying.

A.2. Projecting the Surface Tension Force

In front-tracking, the projection idea can be exploited so that when the curl of the surface tension force is computed, the differentiation occurs on the surface of the interface. Using (19), one has

$$\nabla \times \mathbf{F}_s = \nabla \times (\sigma \kappa \delta(d) \hat{\mathbf{n}}) = \nabla(\sigma \kappa \delta(d)) \times \hat{\mathbf{n}} + \sigma \kappa \delta(d) \nabla \times \hat{\mathbf{n}}. \quad (\text{A.10})$$

Now

$$\nabla \times \hat{\mathbf{n}} = -\frac{\nabla |\mathbf{n}| \times \mathbf{n}}{|\mathbf{n}|^2} \quad (\text{A.11})$$

since $\mathbf{n} = \nabla I(\mathbf{x})$. For an interface with uniform thickness, $\nabla |\mathbf{n}|$ will have no variation in a direction tangential to the surface. Thus it will point in the direction of \mathbf{n} . Consequently $\nabla |\mathbf{n}| \times \mathbf{n} = 0$. Thus (A.10) simplifies to

$$\nabla \times \mathbf{F}_s = \nabla(\sigma \kappa \delta(d)) \times \hat{\mathbf{n}} = \sigma \kappa \nabla \delta(d) \times \hat{\mathbf{n}} + \delta(d) \nabla(\sigma \kappa) \times \hat{\mathbf{n}}.$$

Using the fact that $\nabla \delta(d) = 0$ at the interfacial point gives

$$\nabla \times \mathbf{F}_s = \delta(d) \nabla(\sigma \kappa) \times \hat{\mathbf{n}} \quad (\text{A.12})$$

at the interface.

When one has connectivity in 2D, one can calculate $\nabla \kappa$ as required in (A.12) at an interfacial point by using centered differences and nearest neighbors. However, unconnected points and 3D calculations require a different approach.

One can achieve acceptable results by calculating $\nabla \kappa$ by differentiating (18) for unconnected points. However, for the case of a circle, we found that the following algorithm for calculating $\nabla \kappa$ further suppressed the magnitude of the parasitic current and could be generalized to be used in a front-tracking code which used connectivity.

Define an averaged grid curvature by interpolation from the interface to the grid (\mathbf{x}_g),

$$\kappa_g = \frac{\sum_p \kappa_p S(\mathbf{x}_p - \mathbf{x}_g)}{\sum_p S(\mathbf{x}_p - \mathbf{x}_g)}. \quad (\text{A.13})$$

Writing

$$\kappa_p = \sum_g \kappa_g S(\mathbf{x} - \mathbf{x}_g)|_{\mathbf{x}_p}, \quad (\text{A.14})$$

one can calculate $\nabla\kappa$ at the interfacial point location by differentiating (A.14):

$$\nabla\kappa_p = \sum_g \kappa_g \nabla S(\mathbf{x} - \mathbf{x}_g)|_{\mathbf{x}_p}. \quad (\text{A.15})$$

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the NASA Microgravity Science and Applications Program funded under NRA-94-OLMSA-05. This research is supported by the US Department of Energy under contract W-7405-ENG-36.

REFERENCES

1. S. O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible, multifluid flows, *J. Comput. Phys.* **100**, 25 (1992).
2. J. Glimm, J. W. Grove, X. L. Li, K. M. Shyue, Y. N. Zeng, and Q. Zhang, 3-Dimensional front tracking, *SIAM J. Sci. Comput.* **19**, 703 (1998).
3. J. Glimm, O. McBryan, R. Menikoff, and D. H. Sharp, Front tracking applied to Rayleigh–Taylor instability, *SIAM J. Sci. Stat. Comput.* **7**, 230 (1986).
4. U. Bielert and M. Sichel, Numerical simulation of premixed combustion processes in closed tubes, *Combust. Flame* **114**, 397 (1998).
5. B. Bukiet, Application of front tracking to two-dimensional curved detonation fronts, *SIAM J. Sci. Stat. Comput.* **9**, 80 (1988).
6. F. X. Garaizar and J. Trangenstein, Front tracking for shear bands in an antiplane shear model, *J. Comput. Phys.* **131**, 54 (1997).
7. H. L. Tsai and B. Rubinsky, A front tracking finite-element study on change of phase interface stability during solidification processes in solutions, *J. Cryst. Growth* **70**, 56 (1984).
8. S. O. Unverdi and G. Tryggvason, Computations of multifluid flows, *Physica D* **60**, 70 (1992).
9. M. R. Nobari, Y. J. Jan, and G. Tryggvason, Head-on collisions of drops: A numerical investigation, *Phys. Fluids* **8**, 29 (1996).
10. S. Nas and G. Tryggvason, Computational investigation of thermal migration of bubbles and drops, in *Proceedings ASME Winter Annual Meeting, AMD 174/FED 175 Fluid Mechanics Phenomena in Microgravity*, edited by D. A. Siginer *et al.*, pp. 71–83.
11. A. Esmaeeli, E. A. Ervin, and G. Tryggvason, Numerical simulations of rising bubbles and drops, *J. Fluid Mech.* **314**, 315 (1996).
12. A. Esmaeeli and G. Tryggvason, Direct numerical simulations of bubbly flows. Part 1. Low Reynolds numbers, *J. Fluid Mech.* **377**, 313 (1998).
13. A. Esmaeeli and G. Tryggvason, Direct numerical simulations of bubbly flows. Part 2. Moderate Reynolds numbers, *J. Fluid Mech.* **385**, 325 (1999).
14. D. Juric and G. Tryggvason, A front-tracking method for dendritic solidification, *J. Comput. Phys.* **123**, 127 (1996).
15. D. Juric and G. Tryggvason, Computation of boiling flows, *Int. J. Multiphase Flow* **24**, 387 (1998).
16. Y. C. Chang, T. Y. Hou, B. Merriman, and S. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, *J. Comput. Phys.* **124**, 449 (1996).
17. A. R. York, Development of modifications to the material point method for the simulation of thin membranes, compressible fluids, and their interactions, Sandia National Laboratory Report SAND97-1893 (1997).

18. A. R. York, D. Sulsky, and H. L. Schreyer, The material point method for simulation of thin membranes, *Int. J. Numer. Method Eng.* **44**, 1429 (1999).
19. H. Hoppe, *Surface reconstruction from unorganized points*, Ph.D. thesis (University of Washington Department of Computer Science and Engineering, 1994).
20. G. Tryggvason, B. Bunner, O. Ebrat, and W. Tauber, Computations of multiphase flows by a finite difference/front tracking method. I. Multi-fluid flows, Lecture notes, Department of Mechanical Engineering and Applied Mechanics, University of Michigan (1998).
21. F. H. Harlow and J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids* **8**, 2182 (1965).
22. B. K. Swartz and B. Wendroff, AZTEC: A front tracking code based on Godunov's method, *Appl. Numer. Math.* **2**, 385 (1986).
23. J. Monaghan, Particle methods for hydrodynamics, *Comput. Phys. Rep.* **3**, 71 (1985).
24. D. Burgess, D. Sulsky, and J. U. Brackbill, Mass matrix formulation of the FLIP Particle-in-Cell method, *J. Comput. Phys.* **103**, 1 (1992).
25. J. U. Brackbill, D. B. Kothe, and C. Zemach, A continuum method for modeling surface tension, *J. Comput. Phys.* **100**, 335 (1992).
26. M. Sussman, P. Smereka, and S. Osher, A level-set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
27. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
28. J. R. Baumgardner and P. O. Frederikson, Icosahedral discretization of the two-sphere, *SIAM J. Numer. Anal.* **22**, 1107 (1985).
29. 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**, 349 (1988).
30. S. Popinet and S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, *Int. J. Numer. Methods Fluids* **30**, 775 (1999).
31. B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, *J. Comput. Phys.* **113**, 134 (1994).
32. J. U. Brackbill, D. Juric, D. Torres, and E. Kallman, Dynamic modeling of microgravity flow, in *Proceedings of the Fourth Microgravity Fluid Physics and Transport Phenomena Conference, Cleveland, OH, 1998*, 584–589.