Journal of Computational Physics **183,** 400–421 (2002)
doi:10.1006/jcph.2002.7190

PROST: A Parabolic Reconstruction of Surface Tension for the Volume-of-Fluid Method

Yuriko Renardy[1] and Michael Renardy[2]

Department of Mathematics and ICAM, 460 McBryde Hall, Virginia Tech, Blacksburg, Virginia 24061-0123
E-mail: renardy@math.vt.edu

Received August 28, 2001; revised August 13, 2002

DEDICATED TO KLAUS KIRCHGAESSNER ON THE OCCASION OF HIS SEVENTIETH BIRTHDAY

Volume-of-fluid (VOF) methods are popular for the direct numerical simulation of time-dependent viscous incompressible flow of multiple liquids. As in any numerical method, however, it has its weaknesses, namely, for flows in which the capillary force is the dominant physical mechanism. The lack of convergence with spatial refinement, or convergence to a solution that is slightly different from the exact solution, has been documented in the literature. A well-known limiting case for this is the existence of spurious currents for the simulation of a spherical drop with zero initial velocity. These currents are present in all previous versions of VOF algorithms. In this paper, we develop an accurate representation of the body force due to surface tension, which effectively eliminates spurious currents. We call this algorithm PROST: parabolic reconstruction of surface tension. There are several components to this procedure, including the new body force algorithm, improvements in the projection method for the Navier–Stokes solver, and a higher order interface advection scheme. The curvature to the interface is calculated from an optimal fit for a quadratic approximation to the interface over groups of cells.

Key Words: volume-of-fluid method; continuous surface force; multiphase flow.

1. INTRODUCTION

The volume-of-fluid method is based on solving the flow equations on a rectangular grid, regardless of the geometry of fluid interfaces. These interfaces are reconstructed from the values of a color (or VOF) function which represents the volume fraction of one of the fluids

[1] Supported by NSF-CTS 0090381, NSF-INT 9815106, ACS-PRF, NSF-SCREMS, NCSA.
[2] Supported by NSF-DMS 9870220, NSF-INT 9815106.

0021-9991/02 $35.00

in each grid cell:

$$C(\mathbf{x}) = \begin{cases} 1, & \text{fluid 1}, \\ 0, & \text{fluid 2}. \end{cases} \tag{1}$$

In the continuous surface force (CSF) algorithm [1–3], surface tension forces are computed as a body force,

$$\mathbf{f} = \sigma\kappa\mathbf{n}\delta_S, \tag{2}$$

where σ is the coefficient of surface tension, κ is the mean curvature, $\mathbf{n}$ is the normal to the surface, and δ_S is a delta function concentrated on the interface. Alternatively, the continuous surface stress (CSS) method [4, 5] represents surface tension as the divergence of a stress tensor,

$$\mathbf{f} = \nabla \cdot \mathbf{T}, \quad \mathbf{T} = [(\mathbf{1} - \mathbf{n} \otimes \mathbf{n})\sigma\delta_S]. \tag{3}$$

In the continuum limit,

$$\mathbf{n}\delta_S = -\nabla C, \tag{4}$$

where C is the color function. On discretization, the derivatives of the color function are replaced by finite differences. The problem is that finite difference approximations for derivatives of a discontinuous function are highly inaccurate. This leads to serious errors in the computation of both the normal and the curvature. These errors are particularly highlighted in the classical example given in Section 4. The initial condition is a spherical drop with zero velocity (and at zero gravity), so that this exact solution should be preserved for all time. However, previous VOF methods lead to the creation of spurious currents local to the interface. Although these spurious currents are small as long as the Ohnesorge number

$$Oh = (Ca/Re)^{1/2} \tag{5}$$

(where $Ca = \mu U/\sigma$, $Re = Ua\rho/\mu$, μ is viscosity, U is speed, σ is interfacial tension, a is length scale, and ρ is density) is sufficiently large, they do not disappear with mesh refinement, leading to nonconvergence of the method. The problem can be ameliorated by smoothing the color function before taking derivatives [6]. To actually restore convergence to the scheme, however, smoothing would have to be over a distance which is large relative to the mesh size; this is unrealistic in practice. Moreover, smoothing leads to artifacts of its own by effectively suppressing surface tension in high-curvature regions [7, 8].

The interface is reconstructed from the volume fraction function, usually as a linear interface per cell with the piecewise linear interface construction (PLIC) method. A potentially more accurate method is given in [9] for the two-dimensional case. The authors note that what they really need is an equation for the interface given the color function, but that the reverse is easier. Given an exact formula for the interface, namely arcs of circles, they calculate what the color function needs to be; they generate many such cases, which are stored in a data bank. They generate an empirical formula, which they fit to this data base. They postulate a third-degree polynomial to fit the data. The arcs of circles provide a higher degree correction to the traditional linear interface of the PLIC method. They show that this

drastically reduces the spurious currents. They remark that the method is extendable to three dimensions if a uniform grid with cubic cells is used. They suggest the use of spherical surfaces in place of circles. We remark, however, that this would not work because two independent principal radii of curvature are needed in 3D.

We also refer readers to the literature for other approaches to reduce spurious currents by appropriate methods of interface reconstruction. In [10] a cubic spline is fitted to the values of the color function. The code of [11] also uses cubic splines but tracks the interface using marker particles; the color function is reconstructed from the interface rather than the other way around. Both these schemes are limited to the two-dimensional case at this point. In [12] an algorithm is developed to reconstruct surface tension forces from a set of given points on an interface in three dimensions without any a priori assumptions on connectivity of the interface. In contrast to our approach, the works of [11, 12] track rather than capture the interface, i.e., the primary information is a set of points on the interface rather than volume fractions in grid cells.

In this paper, we introduce and test a novel algorithm for the computation of surface forces for the general three-dimensional case. Our algorithm is based on the VOF method, which reconstructs the interface from a color function. However, we refer to our method as a sharp interface because the advection of the color function is based on a Lagrangian scheme which allows no diffusion (i.e., the color function changes form zero to one within a mesh width). Unlike previous algorithms such as CSF or CSS, we never need to consider a smoothed color function. Instead, we calculate a least-squares fit of a quadratic surface to the color function for each interface cell and its neighbors. In 3D, there are 27 cells involved. The difference with the work of [9] is that we do not deal with empirical formulas and data bases. Our surface tension algorithm leads to a second-order accuracy in the interface reconstruction but is incompatible with the Lagrangian advection scheme described in [13] for PLIC, which is of low order. Therefore, it is necessary to advect the interface more accurately. In Section 3, a new higher order interface advection scheme is formulated. Our surface tension algorithm together with our improved advection scheme comprise PROST and result in the more accurate tracking of interfacial deformation.

In Section 4, we demonstrate that PROST can be used to effectively eliminate spurious currents. Here, the code is begun with an exact solution, that of a spherical drop with zero velocity, satisfying the governing equations, together with the body force of Eqs. (2) and (4):

$$\mathbf{f} = -\sigma\kappa\nabla C. \tag{6}$$

If κ is a constant (i.e., for a sphere), the surface tension force is cancelled by a pressure gradient and drives no flow. The key idea is that this still remains the case at the discrete level, as long as ∇C and ∇p are discretized by the same finite difference method (both ∇C and ∇p are singular at the interface, but they cancel each other out). Hence the crucial ingredient in avoiding spurious currents is an accurate approximation to the curvature. This part of our algorithm is tested in Section 4. We show a comparison of the performance of the past CSF with smoothing, and the CSS without smoothing. The magnitudes of the spurious currents for the CSF and CSS schemes are shown in various norms, for spatial mesh refinements, and are shown to be nonconvergent or convergent only in certain norms. In comparison, our method is shown to converge with spatial refinement and to produce no spurious currents.

The entire PROST algorithm including the higher order advection scheme is applied to three-dimensional drop deformation under simple shear in Section 5. Simulations for a spherical drop suspended in a second immiscible liquid, sheared between parallel plates, are compared with that of a boundary integral code with an adaptive mesh [14, 15]. This boundary integral code is expected to produce quite accurate results for the Stokes flow regime. The evolution to steady state is documented for CSF, CSS, and PROST. We show that for both CSF and CSS, the drop shapes converge with spatial refinement to steady-state lengths that are slightly too long. The PROST simulation shows convergence with spatial refinement toward the boundary integral results.

2. SHARP INTERFACE ALGORITHM FOR INTERFACIAL TENSION

We use the VOF code SURFER, which is described in [4, 16, 17]. There are three main components to the code: (i) the interface tracking component based on the piecewise linear interface construction method (PLIC) and a Lagrangian advection scheme, (ii) the Navier–Stokes solver based on a projection method and a multigrid solver for the pressure field, and (iii) the interfacial tension algorithm, CSF or CSS. Our basic scheme is that employed in SURFER++, which has additional algorithms and physics and has been continually developed [7, 18–26]. In this paper, we shall focus on the new aspects.

2.1. Approximation for Interface Shape

We need to identify those cells which contain the interface in order to determine the interface normal and curvature. The color function is zero in one of the two fluids and one in the other, and in the code we call a cell, an interface cell, if the value of the color function, is between 10^{-8} and $1 - 10^{-8}$.

Figure 1 shows a sample grid, two-dimensional for simplicity. For each interface cell, such as the one at the arrow with center coordinate $\mathbf{x}_0$, we attach the neighbors and examine the nine cells. In Fig. 1, four out of the nine cells contain the interface. For the central cell,

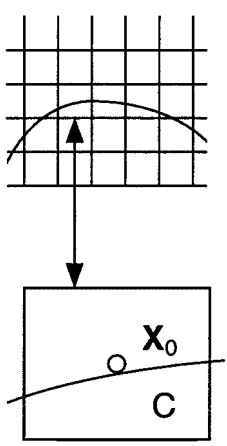

FIG. 1. Sample two-dimensional grid. The arrow points to an interface cell with center coordinate $\mathbf{x}_0$, and volume fraction C. The interface is fitted with a quadratic surface through this cell plus its neighbors, a total of nine cells in 2D. Here, four out of the nine are interface cells.

the interface is fitted with a quadratic equation of the form

$$k + \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{A}(\mathbf{x} - \mathbf{x}_0) = 0. \tag{7}$$

In three dimensions, there are 27 cells around the center node $\mathbf{x}_0 \cdot \mathbf{n}$ is a unit vector approximating the normal, and $\mathbf{A}$ is a symmetric matrix with the property that $\mathbf{An} = 0$; that is, the axis of the paraboloid is along $\mathbf{n}$. This quadratic equation involves six parameters: one for k, two for $\mathbf{n}$ with normalization, and three for $\mathbf{A}$. For any given choice of these parameters, we can compute the volume fraction v which the quadratic surface cuts out of a grid cell (details of this computation are described in Section 2.5). The parameters for the quadratic surface can be organized as a six-dimensional vector $\mathbf{q}$, which is to be determined in such a way that

$$f(\mathbf{q}) = \sum w(v - C)^2 \tag{8}$$

is minimized. Here the sum is over the given interface cell and all its neighbors, v is the volume fraction in each cell determined by the quadratic surface, C is the value of the color function in the cell, and w is a weight discussed below. For reasons of computational efficiency, we have modified this and extended the sum only over interface cells. For the 2D case in Fig. 1, $\mathbf{q}$ is a three-dimensional vector and v is the area under the parabola divided by the total area of the cell. $f(\mathbf{q}) = \sum_{i=1}^{4}(v_i - C_i)^2$ must be minimized. Ideally, we would like $v_i = C_i$ but this is four equations and there are three parameters for the parabola.

The reader may wonder why we restrict the quadratic terms by the constraint $\mathbf{An} = \mathbf{0}$. The rationale for this is that other quadratic terms in (7) would be effectively of cubic order. As a simple illustration, consider a curve

$$y = ax^2 + bxy + cy^2; \tag{9}$$

i.e., the curve passes through the origin and the normal is in the y-direction. We can rewrite the equation of the curve in the form

$$y = ax^2 + abx^3 + (ab^2 + ca^2)x^4 + O(x^5), \tag{10}$$

i.e., only a really contributes at quadratic order, while b and c contribute at cubic and quartic order.

Finally, we need to explain the weights included in (8). The simplest choice would be to simply set $w = 1$ in each cell. We modified this for two reasons. First, if an interface passes close to a corner of a cell, then the value of the color function is quite insensitive to the position of the interface. If the interface intersects a corner at a generic angle and is moved by a distance ϵ, the color function only changes by $O(\epsilon^3)$. To deal with this issue, we introduce a weight factor

$$w_1 = \frac{1}{C(1 - C) + \delta}, \tag{11}$$

which forces a better fit in cells where C is close to 0 or 1. Numerical experiments with different values of δ led us to the choice $\delta = 0.01$. Actually, this made no real difference to the results of the simulation, but it reduced the number of cases where the optimization algorithm was slow to converge. A second, much more serious issue relates to the formation

of checkerboards. Consider an interface that is a graph $z = f(x, y)$, where f is a step function assuming constant values in a checkerboard pattern. No quadratic surface can fit such a checkerboard, and an unweighted sum in (8) does not produce a surface tension force to suppress it. We saw such checkerboards emerge over time in some of our simulations. There is no numerical instability, but checkerboards represent a neutral mode which can gradually grow. As a mechanism for suppressing checkerboards, we introduce a weight w_2, which equals 1 for the "central" cell, $1/2$ for each cell sharing a face with it, $1/4$ for each cell sharing an edge with it, and $1/8$ for each cell sharing a corner with it. Finally, w is equal to the product $w_1 w_2$.

2.2. Optimal Fit

The basic algorithm for the least-squares optimization is that described in Sections A5.5.1 and A5.5.2 of [27]. This algorithm is implemented in the IMSL package, but we chose to write our own code rather than use IMSL because (as of the time we wrote our code) this part of the IMSL package is not thread safe for open MP parallelism. To minimize the function $f(\mathbf{q})$, we basically use the Newton iteration for finding zeros of ∇f; i.e.,

$$\mathbf{q}_{i+1} = \mathbf{q}_i - (\nabla^2 f(\mathbf{q}_i))^{-1} \nabla f(\mathbf{q}_i), \quad i = 0, 1, 2, \ldots, \tag{12}$$

where $\mathbf{q}_0$ is the starting value. The derivatives of f are calculated by finite difference approximations. The Newton iteration is set up in the following way, to make sure that $f(\mathbf{q}_{i+1})$ is less than for the previous iterate $f(\mathbf{q}_i)$. If the minimum is sufficiently close, then $\nabla^2 f$ is positive definite. On the other hand, the finite difference approximation of the matrix $\nabla^2 f$ may not be positive definite because of numerical errors. In order to hit a better iterate, a constant is added to the diagonal to make it positive definite [27]. We note that the Newton iteration assumes f to be smooth, and our function is actually not smooth when the interface crosses a corner of a grid cell. To handle these cases, we modify the algorithm when $f(\mathbf{q}^{n+1})$ does not come out less than $f(\mathbf{q}^n)$, $|\nabla f|$ is larger than a specified tolerance, or convergence is not obtained in a satisfactory number of steps. If this happens, we set

$$\mathbf{q}^{n+1} = \mathbf{q}^n - \lambda \nabla f(\mathbf{q}^n), \tag{13}$$

where λ is chosen such that $f(\mathbf{q}^{n+1})$ is as small as possible. If this algorithm still does not achieve convergence, we do an alternating search along coordinate directions, each time searching for the minimum along a line.

By combining these algorithms, we are able to obtain convergence of the optimization algorithm in almost all cases. Some exceptions occur as breakup is reached, and in these cases we simply "give up" and stop the optimization algorithm after 200 steps.

To improve the mass conservation of the scheme, we adjust the value of k at the end of the optimization so that the value of the color function C in the central cell is fitted exactly (without such a correction, we do not obtain acceptable results for mass conservation). We make an exception to this, however, if cells are almost empty or full, because, in these cases, as mentioned above, a fairly large shift of the interface may be needed to achieve the right value of the color function. This shift would then produce a quite inaccurate interface, leading to interface reconstructions in certain cells which fit poorly with neighboring cells. A satisfactory compromise was achieved by adjusting k only for cells with $0.02 \leq C \leq 0.98$.

The iteration for the optimization needs a starting value. At the initial time, we generate this starting value from the analytic form of the initial interface. At subsequent times, the starting value is the equation of the interface at the previous time step, updated by advection, as described below.

2.3. The Mean Curvature κ

The surface tension force is now calculated as $-\sigma\kappa\nabla C$, where κ is calculated as $2\mathrm{tr}(\mathbf{A})$, with $\mathbf{A}$ being the matrix from (7). This defines κ at cell centers, while the components of ∇C are defined at the centers of the faces of a cell. We interpolate the value of κ as a weighted average; that is, in the x-component, the combination $\kappa\partial C/\partial x$ is

$$\left[\kappa\frac{\partial C}{\partial x}\right](x_{i-1/2}, y_j, z_k) = \frac{w_1\kappa(x_{i-1}, y_j, z_k) + w_2\kappa(x_i, y_j, z_k)}{w_1 + w_2} \times\left(\frac{C(x_i, y_j, z_k) - C(x_{i-1}, y_j, z_k)}{\Delta x}\right), \tag{14}$$

where

$$w_1 = C(x_{i-1}, y_j, z_k)(1 - C(x_{i-1}, y_j, z_k)), \quad w_2 = C(x_i, y_j, z_k)(1 - C(x_i, y_j, z_k)). \tag{15}$$

The rationale behind this weighting is that we expect the approximation to the curvature to be more accurate when C is not close to 0 or 1. For example, if the cell at (x_{i-1}, y_j, z_k) is not an interface cell, then $\kappa(x_{i-1}, y_j, z_k)$ is not defined there, but its weight w_1 is zero, so that the combination is zero.

2.4. Singular Part of Pressure Gradient

In our Navier–Stokes solver, we decouple the velocity and pressure fields with a projection method [28]. At each time step, we first integrate the velocities as though the pressure term were absent and then solve for the pressure to make the velocity divergence free. Namely, given the quantities at time level n, an intermediate velocity $\mathbf{u}^*$ is calculated; i.e.,

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\mathbf{u}^n\cdot\nabla\mathbf{u}^n + \frac{1}{\rho}(\nabla\cdot(\mu\mathbf{S}) + \mathbf{F})^n, \tag{16}$$

where $\mathbf{F}$ includes gravity and the interfacial tension force, and $\mathbf{S}$ is the strain rate tensor. $\mathbf{u}^*$ is not, in general, divergence free, and

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p}{\rho}, \tag{17}$$

where $\mathbf{u}^{n+1}$ at time level $n+1$ satisfies

$$\nabla\cdot\mathbf{u}^{n+1} = 0. \tag{18}$$

When Eq. (17) is substituted into Eq. (18), a Poisson equation for the pressure field results:

$$\nabla\cdot\left(\frac{\nabla p}{\rho}\right) = -\frac{\nabla\cdot\mathbf{u}^*}{\Delta t}. \tag{19}$$

We can combine the equations above and obtain

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = -\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{\rho}(\nabla \cdot (\mu \mathbf{S}) + \mathbf{F})^n - \frac{\nabla p}{\rho}. \tag{20}$$

In our code, however, we do not use the explicit scheme. For the velocity solution, we use the semi-implicit method of [21], which is lengthy to write out in full. Here, therefore, we write it in symbolic form:

$$\begin{aligned} \rho \frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} &= \mathcal{A}\mathbf{u}^* + \mathcal{B}(\mathbf{u}^n) + \mathbf{f}, \\ \rho \frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} &= -\nabla p^{n+1}. \end{aligned} \tag{21}$$

$\mathcal{A}$ represents the operator that acts on the term $\mathbf{u}^*$ and $\mathcal{B}$ represents the one acting on $\mathbf{u}^n$ and are given in [21]. Again, $\mathbf{f}$ is the body force which includes the singular surface tension force. Combining the two equations yields

$$\rho \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = \mathcal{A}\mathbf{u}^{n+1} + \mathcal{B}(\mathbf{u}^n) + \mathbf{f} - \nabla p^{n+1} + (\Delta t)\mathcal{A}(\rho \nabla p^{n+1}). \tag{22}$$

The last term may be significant unless Δt is very small, since p includes the singular pressure which is needed to balance surface tension. For this reason, we modify our scheme as follows:

$$\begin{aligned} \rho \frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} &= \mathcal{A}\mathbf{u}^* + \mathcal{B}(\mathbf{u}^n) + \mathbf{f} - \nabla p_1, \\ \rho \frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} &= -\nabla p_2. \end{aligned} \tag{23}$$

Here, p_1 is chosen such that $\mathbf{f} - \nabla p_1$ is divergence free. Combining the two equations now yields

$$\rho \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = \mathcal{A}\mathbf{u}^{n+1} + \mathcal{B}(\mathbf{u}^n) + \mathbf{f} - \nabla(p_1 + p_2) + (\Delta t)\mathcal{A}(\rho \nabla p_2), \tag{24}$$

which shows that since ∇p_2 is orders of magnitude smaller than ∇p_1, the $O(\Delta t)$ error term is much smaller. This algorithm measurably improves the size of the time step that can be used. For example, the results of Section 5 are obtained with 10 times the time-step size used in [8, 21, 22, 26, 29] and are more accurate.

2.5. Second-Order Algorithm for Volume Fraction

To complete the description of the numerical algorithm, we need to describe how we calculate the volume which the quadratic surface given by (7) cuts out of a cell. That is, we want to find the volume within the parallelepiped

$$\begin{aligned} x_0 - \Delta x/2 \le x \le x_0 + \Delta x/2, \\ y_0 - \Delta y/2 \le y \le y_0 + \Delta y/2, \\ z_0 - \Delta z/2 \le z \le z_0 + \Delta z/2, \end{aligned} \tag{25}$$

where

$$k + \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{A}(\mathbf{x} - \mathbf{x}_0) \leq 0. \tag{26}$$

A naive approach to computing the volume fraction would be to use a "pixel method," i.e., to take a regularly spaced grid of points in a cell and determine the volume fraction by counting the number of points on the right side of the interface. This, however, is hopelessly slow and totally impractical. An exact analytical evaluation of the volume fraction is possible, and indeed we programmed it. We found, however, that this was about five times slower than the algorithm we describe below and, even more serious, we were unable to control round-off errors. For this reason, we settled for an approximate evaluation of the volume fraction.

We can simplify the calculation if we assume that the mesh size is small and the curvature is of order 1. First, we reorient our coordinates such that the largest component of $\mathbf{n}$ is in the z direction and is positive. This will guarantee that within our mesh cell the interface is a graph $z = g(x, y)$. Next, we can approximate g by a quadratic function, as follows. First, use the linear approximation in Eq. (7),

$$z - z_0 \sim -\frac{1}{n_3}(k + n_1(x - x_0) + n_2(y - y_0)), \tag{27}$$

and second, we substitute this expression for z in the quadratic terms. We thus need to find the volume under the graph of the function $z = g(x, y)$ which lies above $z = z_0 - \Delta z/2$ minus the volume which lies above $z = z_0 + \Delta z/2$. To find the volume which lies above, say $z = z_0 - \Delta z/2$, we find the intersection of the graph with the plane $z = z_0 - \Delta z/2$. This is a quadratic curve, and we need to integrate over the area which lies on one side of the curve.

If the projection of the normal onto the x, y-plane is small, we can expect the quadratic curve to have a high curvature, and in this case we use a seminumerical procedure, where the integration in one direction is carried out numerically. Otherwise, we use an analytic approximation. We first, if necessary, cut the base rectangle into quadrants such that in each quadrant the curve representing the interface is represented by a monotone function $y(x)$. Next, the line of intersection is approximated by a straight line obtained from interpolating between the intersection points with the edges of the rectangle. In this fashion, the problem is reduced to integrating a quadratic function over a set which is the union of rectangles and triangles. This procedure determines the volume fraction in the cell (the volume of the intersection $O(h^5)$ divided by the cell volume $O(h^3)$) to within an error of order h^2, where

$$h = \max(\Delta x, \Delta y, \Delta z) \tag{28}$$

is the spatial grid size.

3. HIGHER ORDER INTERFACE ADVECTION SCHEME

The original advection scheme in SURFER is based in a piecewise linear interface reconstruction and a Lagrangian scheme which advects successively in the three coordinate directions, To advect in the x-direction, for instance, let us consider the cell indexed by (i, j, k). The volume fraction in this cell is $c(i, j, k)$. At each time step, fluid contained in

cell (i, j, k) can move to cells $(i-1, j, k)$, (i, j, k), and $(i+1, j, k)$. The three fractions are determined as follows. First, the right and left face of the cell are moved, respectively, with the velocity lying at the nodal point at the center of each face, and the interface is subjected to the affine deformation imposed by this motion. After deforming in this way, one determines the amount of fluid which has moved to each neighboring cell by this deformation.

In the new scheme, we apply this same algorithm, with the difference that we use our quadratic representation of the interface instead of a linear reconstruction. At each advection step, we update the equation of the interface as well as the values of the color function. If the interface moves into new cells, we extrapolate the interface equation from a neighboring cell to define an interface equation in the new cell.

To improve the accuracy of the advection, we apply a "shear correction," which takes account of the fact that the speed with which fluid 1 is advected in a cell may differ from the average speed because of shear. Since the difference in speed is small, of order h, we use a Eulerian scheme based on fluxes. For each face of the cell, we then calculate the area on this face which lies within fluid 1 and the center of mass of this area. We use linear interpolation from nodal values to determine a normal velocity at the center of mass. We then take the difference between this velocity at the center of mass and the nodal velocity at the center of the face of the cell. For this difference velocity, a volume flux is calculated from area $\times$ normal velocity $\times$ time step. For each face of the cell, we determine the volume of fluid leaving the cell if this flux is outward. The volume remaining in the cell is the original volume minus the total volume leaving. Finally, the new value of the color function in each cell is found by adding the volume remaining in that cell and the volume entering from neighboring cells. We note that the linear interpolation of velocity near the interface is not correct unless the viscosity ration is 1, and hence the improvement of the shear correction is only qualitative. A "better" alternative would then depend on the interpretation of what exactly the nodal values of velocity represent. We do not attempt to address such issues since, in any case, the proper treatment of discontinuities in the velocity gradient has not been addressed in other parts of the code, such as the momentum equation.

To calculate the area and center of mass, we can usually assume that the curve representing the intersection of the interface and the face of the cell is close to a straight line. If this is the case, we use an analytical approximation where the area is determined as a union of rectangles and triangles which are found from the intersections of the interface with the edges. At points where the curvature is large, we use a numerical evaluation instead. For a smooth interface, this situation occurs when, for instance, the face of the cell is parallel to the x, y-plane and the normal of the interface is close to the z-direction.

4. NUMERICAL RESULTS ON SPURIOUS CURRENTS

Spurious or parasite currents are described in [5] as vortices "in the neighborhood of interfaces despite the absence of any external forcing. They are observed with many surface tension simulation methods, including the CSF method, the CSS method, and the lattice-Boltzmann method, in which they were first discovered."

In this section, we compare the CSF, CSS, and PROST methods. The specific version of the CSF method we use is based on [1]; for the smoothing kernel we use the kernel K_8 of [6]. One smoothing iteration is used for the normal and two for the curvature. (Some alternative implementations of the CSF method, which lead to some improvement of spurious currents, are discussed in [6].)

TABLE I
Norms of Velocity at 200th Time Step, $\Delta t = 10^{-5}$

Δx	L_∞	L_2	L_1	Method
1/96	0.00179982	0.00008403	0.00001473	CSF
1/128	0.00184090	0.00008542	0.00001539	
1/160	0.00189053	0.00008596	0.00001569	
1/192	0.00196880	0.00008627	0.00001569	
1/96	0.00377043	0.00014183	0.00001920	CSS
1/128	0.00358876	0.00012446	0.00001615	
1/160	0.00360416	0.00011230	0.00001438	
1/192	0.00398403	0.00010453	0.00001346	
1/96	0.00002243	0.00000087	0.00000014	PROST
1/128	0.00001309	0.00000053	0.00000009	
1/160	0.00000954	0.00000041	0.00000007	
1/192	0.00000568	0.00000023	0.00000004	

Our computational domain is $1 \times 1 \times 1$, the spatial mesh keeps $\Delta x = \Delta y = \Delta z$, and the time step is $\Delta t = 10^{-5}$. The boundary conditions are zero velocity at the top and bottom walls, and periodicity in x- and y-directions. Initially, a spherical drop is centered at (0.5, 0.5, 0.5), with radius $a = 0.125$ and surface tension $\sigma = 0.357$. Both fluids have equal density, 4, and viscosity, 1. The initial velocity field is zero. The exact solution is zero velocity for all time. In dimensionless terms, the relevant parameter is the Ohnesorge number $Oh = (\mu^2/\sigma\rho a)^{1/2} \sim 2.37$.

The amplitude Max $|\mathbf{u}|$ of the spurious currents is difficult to estimate a priori but Refs. [4, 5] find from direct measurements that they are of order $0.01\sigma/\mu$, leading to a Reynolds number based on spurious currents, which is of the order $0.01/Oh^2$. At small Ohnesorge numbers, this can lead to troublesome flow instabilities. Table I verifies this scaling for the CSF and CSS methods. The table shows the L_∞, L_2, and L_1 norms of the velocity field for the spurious currents. While the L_∞ norm gives the maximum speed, the L_2 and L_1 norms indicate a measure in an average sense for the computational domain. Table II shows results for PROST with larger time steps, showing that entries of Table I are converged with respect to time step. The scaling of spurious currents with σ/μ is obvious, since the surface tension force driving them is proportional to σ and is counteracted by viscous damping. Since the PROST algorithm reduces spurious currents by two to three orders of magnitude, it will also eliminate the instabilities at low Ohnesorge numbers.

TABLE II
Norms of Velocity at 20th Time Step, $\Delta t = 10^{-4}$

Δx	L_∞	L_2	L_1	Method
1/96	0.00002156	0.00000084	0.00000014	PROST
1/128	0.00001264	0.00000051	0.00000008	
1/160	0.00000905	0.00000039	0.00000006	
1/192	0.00000545	0.00000022	0.00000004	

Note. This shows that the results in Table I for the PROST method are converged with respect to the time step.

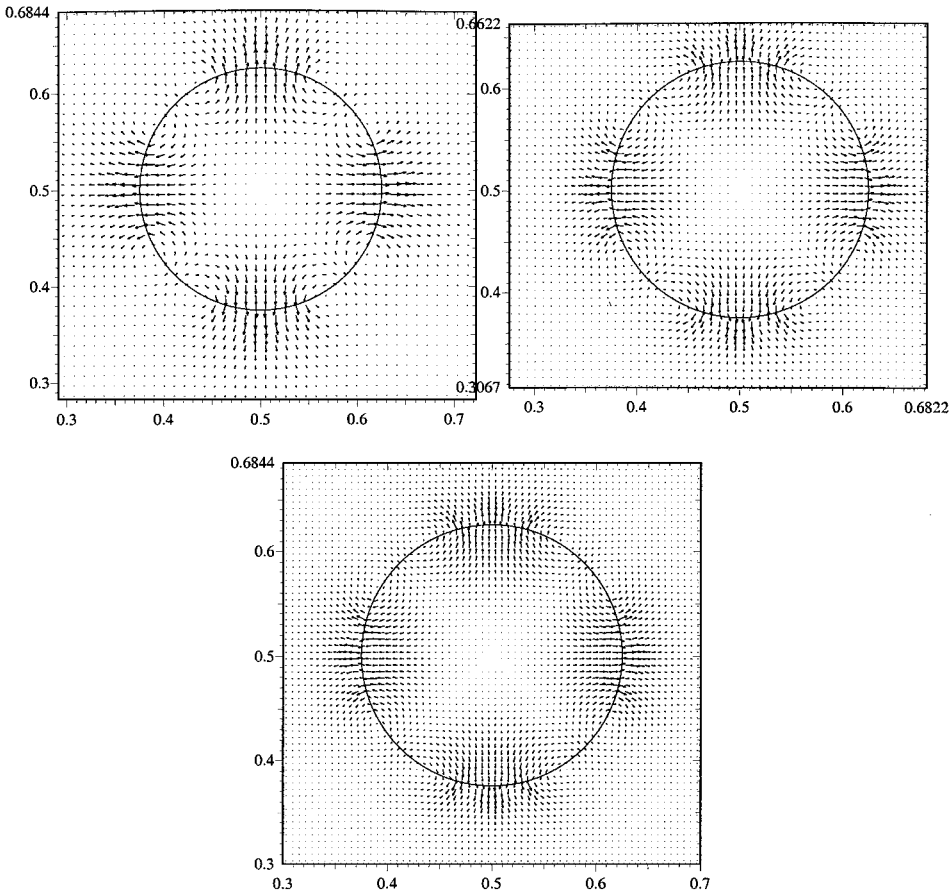

FIG. 2. CSF, velocity vector plot across centerline in the x–z-plane at 200th time step, $\Delta t = 10^{-5}$. These show the locations of the spurious currents with mesh refinement, $\Delta x = 1/96,\ 1/128,\ 1/160$. The relative magnitudes of the vectors are given in Table II.

These tables show the following:

1. For the CSF method, mesh refinement does not decrease the spurious currents in any of the norms. In fact, they increase slightly in L_∞ and L_2 and stay about the same in L_1. To illustrate this, Fig. 2 shows the two-dimensional cross section of the drop in the x–z-plane. The spurious vortices are present at the same positions and spread over the same amount of the domain for all the meshes.

2. For theCSS method, mesh refinement does not decrease the L_∞ norm of spurious currents but decreases the L_2 and L_1. Figure 3 shows half of the cross section of the drop in the x–z-plane. With mesh refinement, the number of vortices increases, and they cover less domain.

3. For the PROST method, spurious currents are so small that they are effectively not present. This is no surprise. As we pointed out earlier, there would be NO spurious currents if curvature were constant and our initial interface a sphere. The only reason the curvature is not exactly constant is because the least-squares fit approximates the sphere as piecewise paraboloids. The magnitude for the $\Delta x = 1/96$ mesh is 1/100th that of CSF or CSS and thereafter decreases with mesh refinement with $O(h), h = \text{Max}\{\Delta x, \Delta y, \Delta z\}$. Figure 4 plots

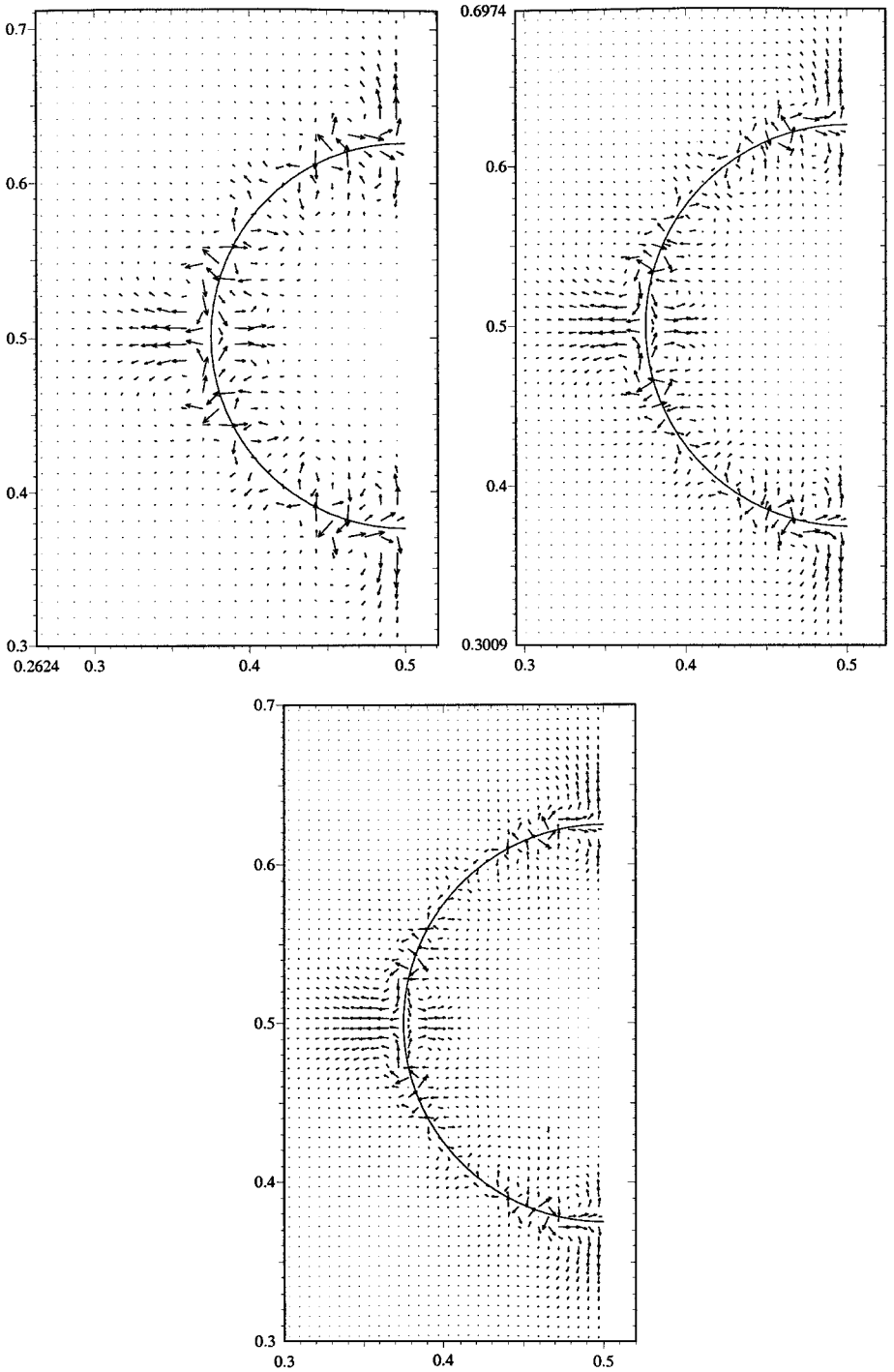

FIG. 3. CSS, not mollified, velocity vector plot across centerline in the x–z-plane at 200th time step, $\Delta t = 10^{-5}$. Mesh refinement study, $\Delta x = 1/96,\ 1/128,\ 1/160$. Each vector plot is magnified so that the vectors are visible. The relative magnitudes of the vectors are given in Table II.

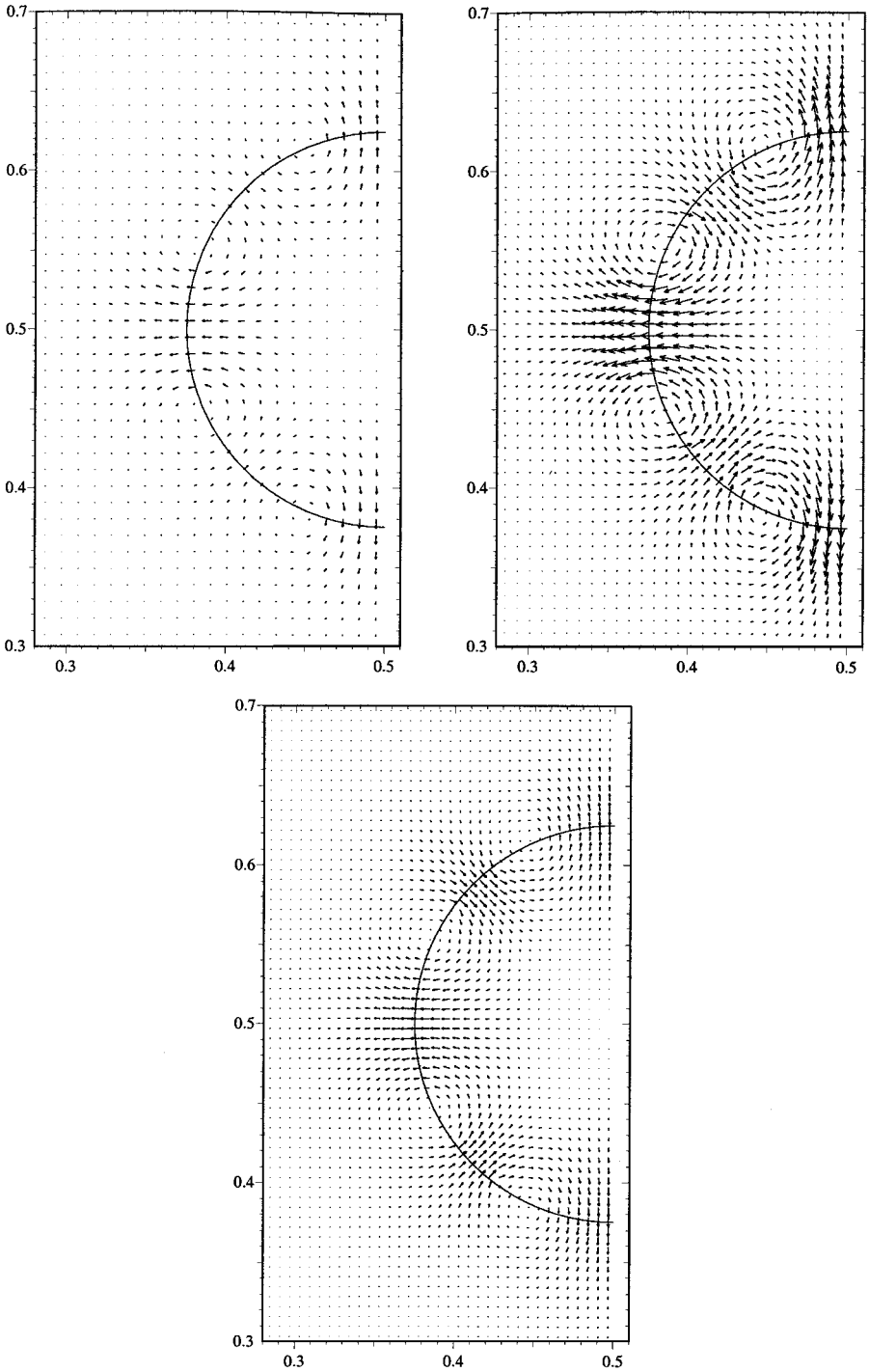

FIG. 4. VOF-PROST, velocity vector plot across centerline in the x–z-plane at 200th time step, $\Delta t = 10^{-5}$. Variation of spurious current with mesh refinement, $\Delta x = 1/96,\ 1/128,\ 1/160$. Each vector plot is magnified so that the vectors are visible, the latter two plots being magnified much more than the first. The relative magnitudes of the vectors are given in Table II.

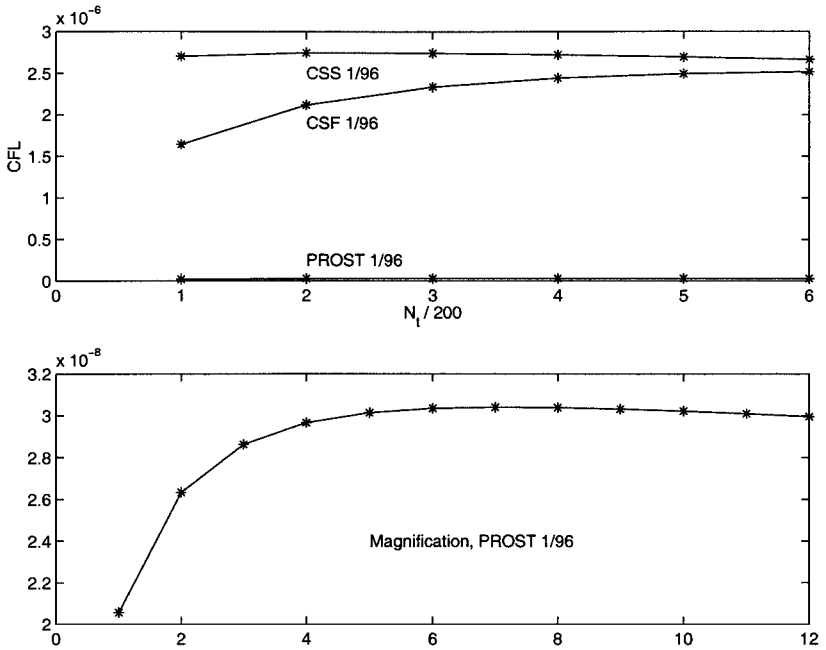

FIG. 5. CFL numbers vs time steps (200), $\Delta t = 10^{-5}$. The CFL numbers remain the same order of magnitude as that at the 200th time step.

the velocity vectors in comparison to Figs. 2 and 3. The three plots in Fig. 4 should not be compared for relative magnitudes. In order for the spurious currents to be discernible, the $1/128$ and $1/160$ cases are magnified much more than for the $1/96$ mesh. To obtain the true relative magnitudes among the three meshes, we refer the reader to Table II. Figures 2–4 and Tables I and II show that the locations of the spurious currents for PROST are similar to those of CSF, and that the magnitudes decay with spatial refinement.

Figure 5 shows the evolution of the Courant–Friedrichs–Levy number (CFL),

$$\mathrm{CFL} = U_{\max}\Delta t/\Delta x, \quad U_{\max} = \max|\mathbf{u}|. \tag{29}$$

This is a measure of the L_∞ norm of the spurious current. The figure shows the evolution of the CFL number over a longer time interval to show that it settles to some saturation level, and the sampling at the 200th time step for Table I is already at that magnitude.

5. NUMERICAL RESULTS ON DROP DEFORMATION

VOF-PROST is compared with the boundary integral method of [14, 30] for drop deformation under simple shear. The geometry is a Couette device, with a drop suspended in a second liquid, and sheared by the motion of outer walls. A capillary number is defined by

$$Ca = \frac{\eta\dot{\gamma}a}{\sigma}, \tag{30}$$

where $\dot{\gamma}$ is the shear rate (normalized to 1 in the computations), η is the viscosity of the matrix fluid, σ is the interfacial tension coefficient, and a is the initial drop radius. At $Ca = 0.35$,

TABLE III

Table of Capillary Time, Physical Time, the Boundary Integral Results, Results for $Re = 0.0625$ from VOF-CSF with (1) $h = 1/128$, $\Delta t = 5 \times 10^{-4}$, (2) $h = 1/128$, $\Delta t = 2 \times 10^{-4}$

Capillary time	Time	B. I. Stokes	CSF 1	CSF 2
0.000	0.000	0.997	0.999	0.999
1.000	0.350	1.150	1.154	1.154
2.000	0.700	1.265	1.277	1.276
3.000	1.050	1.351	1.371	1.367
4.000	1.400	1.422	1.447	1.441
5.000	1.750	1.475	1.511	1.495
6.000	2.100	1.521	1.554	1.543
7.000	2.450	1.558	1.597	1.582
8.000	2.800	1.586	1.629	1.615
9.000	3.150	1.611	1.659	1.644
10.000	3.500	1.631	1.682	1.665
15.000	5.250	1.691	1.757	1.736
20.000	7.000	1.711	1.787	1.761
30.000	10.500	1.718	1.815	1.775
40.000	14.000	1.717	1.816	1.776
50.000	17.500	1.717	1.816	1.776

the drop does not break up in Stokes flow. For our similations, we use a Reynolds number of 0.0625, which should be small enough for the results to be only slightly different from Stokes flow.

Tables III and IV show the comparison of the boundary integral method with VOF-PROST and VOF-CSF at $Ca = 0.35$ (the boundary integral method is validated against experiments

TABLE IV

Table of Capillary Time, Results for $Re = 0.0625$ from VOF-PROST with (1) $h = 1/96$, $\Delta t = 10^{-3}$, (2) $h = 1/96$, $\Delta t = 5 \times 10^{-4}$, (3) $h = 1/128$, $\Delta t = 10^{-3}$, (4) $h = 1/128$, $\Delta t = 5 \times 10^{-4}$

Capillary time	PROST 1	PROST 2	PROST 3	PROST 4
0.000	0.997	0.997	0.999	0.999
1.000	1.154	1.153	1.154	1.153
2.000	1.269	1.267	1.272	1.269
3.000	1.359	1.355	1.363	1.358
4.000	1.424	1.417	1.435	1.426
5.000	1.478	1.470	1.490	1.480
6.000	1.521	1.509	1.536	1.524
7.000	1.556	1.542	1.573	1.559
8.000	1.585	1.570	1.604	1.588
9.000	1.607	1.590	1.630	1.612
10.000	1.625	1.607	1.652	1.631
15.000	1.672	1.650	1.711	1.685
20.000	1.695	1.660	1.736	1.702
30.000	1.702	1.664	1.749	1.709
40.000	1.702	1.665	1.749	1.709
50.000	1.703	1.665	1.749	1.709

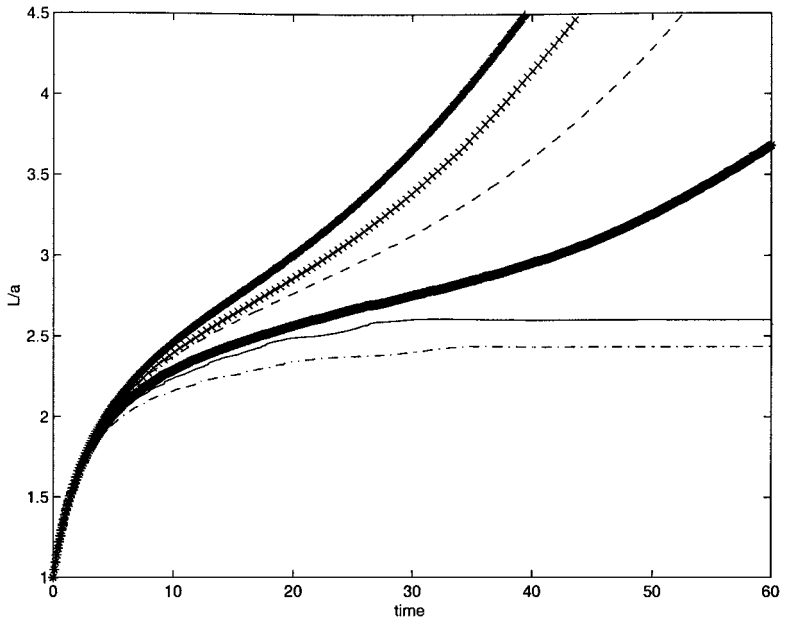

FIG. 6. Evolution of half-length/initial radius for VOF-PROST, with mesh $\Delta x = \Delta y = \Delta z = 1/96$, $\Delta t = 10^{-3}$ (curve 2); $\Delta x = \Delta y = \Delta z = 1/96$, $\Delta t = 5 \times 10^{-4}$ (curve 1); $\Delta x = \Delta y = \Delta z = 1/128$, $\Delta t = 10^{-3}$ (curve 4); $\Delta x = \Delta y = \Delta z = 1/128$, $\Delta t = 5 \times 10^{-4}$ (curve 3); $\Delta x = \Delta y = \Delta z = 1/160$, $\Delta t = 10^{-3}$ (curve 6). Curves are numbered from bottom to top. The results of the boundary integral code (curve 5) are shown for comparison.

in [30] and shows excellent agreement). We used two different mesh sizes (1/96 and 1/128) and different time steps. Clearly, the PROST method agrees much better with the boundary integral results than does CSF. We note that while CSF overpredicts the final length of the drop, PROST has two competing effects. Time discretization overpredicts it, while spatial

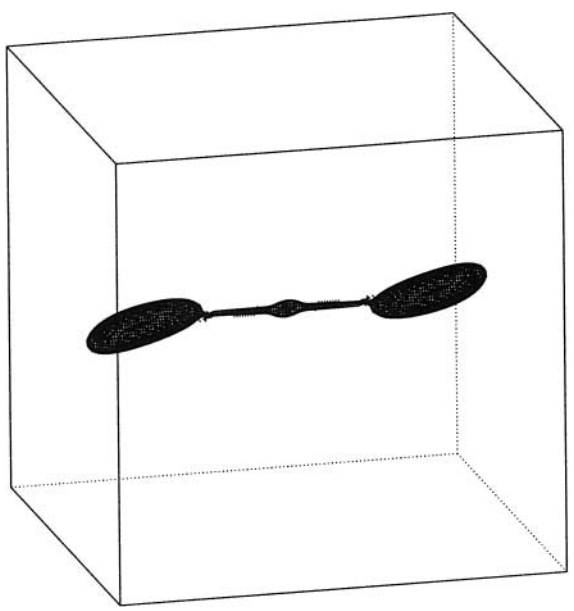

FIG. 7. Drop at time 47.7, just before breakup; full view.

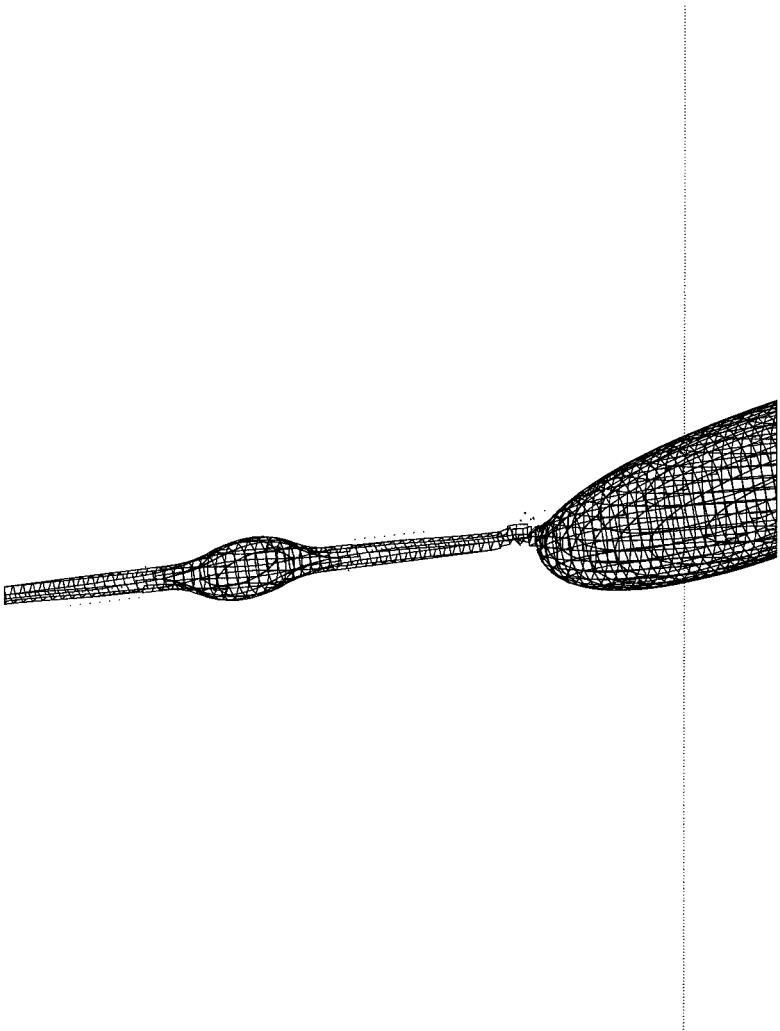

FIG. 8. Drop at time 47.7, just before breakup; view of neck.

discretization underpredicts it; we believe this latter effect is due to the fact that the method overestimates curvature.

We also compared the CPU timings of PROST and CSF for the same time step and mesh size (5×10^{-4} and $1/128$). The optimization to find the least-squares fit to the interface becomes the dominant factor determining the computing time. We can cut down on computing time by performing the optimization only every other time step (the interface still gets advected at every time step; only the "reconciliation" between the equation for the interface and the values of the color function is postponed to the second time step). This change made only minimal difference to the results, and the difference between CSF and PROST for the same mesh size and time step is then roughly a factor of 3. However, the PROST method gives more-accurate results than CSF even on a courser mesh, and the CSF results do not become significantly better with further mesh refinement (due to time constraints we did not do a full calculation for finer meshes, but we did partial calculations, e.g., to find the final equilibrium length).

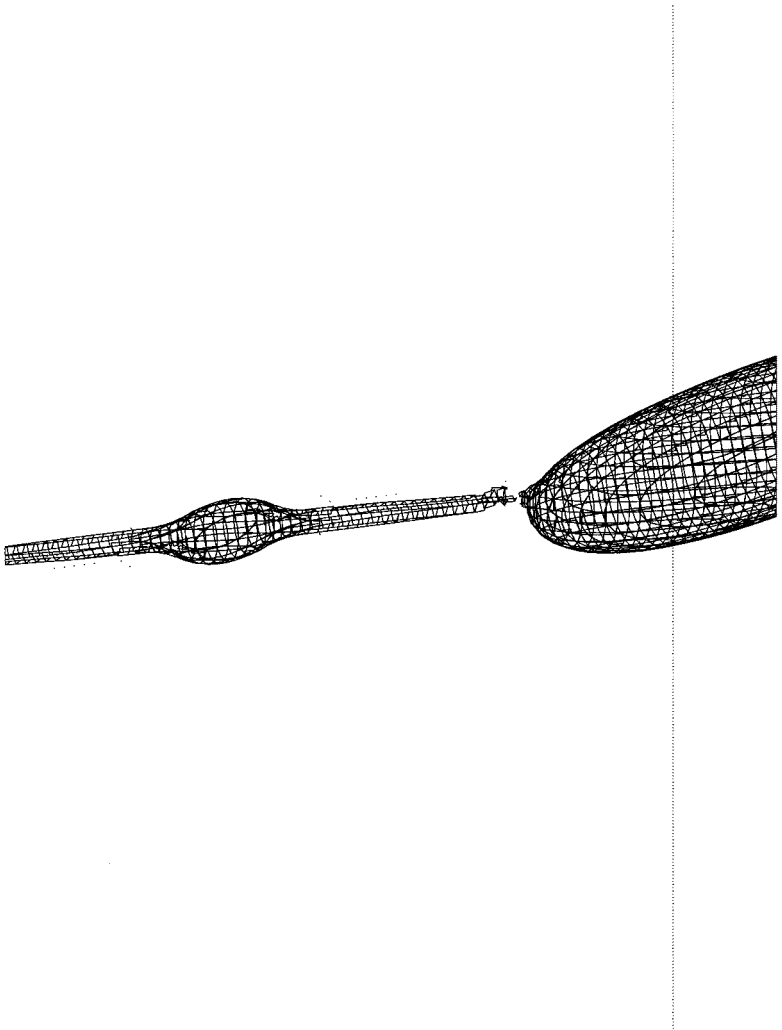

FIG. 9. Drop at time 47.8, close to breakup; view of neck.

We next show a situation where drop breakup occurs. The fluids have equal density, 9, and viscosity, 1, the surface tension parameter is 0.1893, and the initial drop radius is $1/12$. This corresponds to a capillary number of 0.44 and a Reynolds number of 0.0625. This situation is quite close to the critical capillary number and is therefore much more challenging for simulation than the case shown above. Figure 6 shows the evolution of the end-to-end length. The results confirm the trend observed above that the length decreases with temporal refinement and increases with spatial refinement. We also show results of Cristini's boundary integral code. Clearly, the spatial resolution of the 96 and 128 mesh is insufficient, but for the 160 mesh we are beginning to obtain reasonable agreement.

Figures 7–10 show the breakup of the drop, based on our computation with the 160 mesh. It occurs at approximately time 47.8; at this point the length of the drop has increased by a factor of 5.39 from the original sphere. For comparison, Fig. 11 shows the drop shape

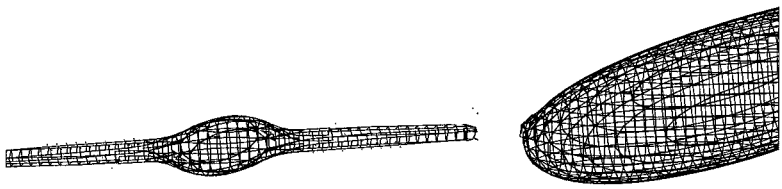

FIG. 10. Drop at time 47.9, after breakup; view of neck.

just before breakup, computed with the boundary integral method. For this plot, the time is 47.5, and the length of the drop (relative to its initial diameter) is 4.84.

For the PROST simulation on the 160 mesh, the volume of the drop is equal to 2482.24 grid cells at the beginning of the simulation and equal to 2487.84 grid cells at time 47.9, just after breakup, illustrating acceptable conservation of mass.

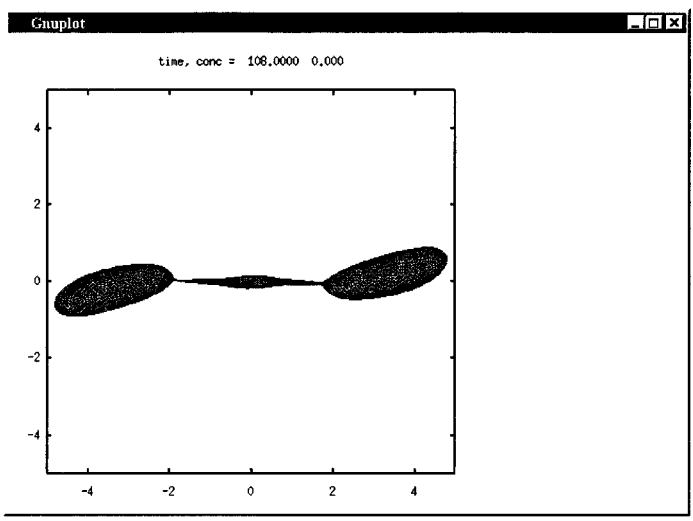

FIG. 11. Drop at time 47.5, boundary integral simulation.

6. CONCLUSION

The PROST algorithm is shown to converge spatially, a feature that is absent from its predecessors, CSF and CSS. PROST effectively eliminates the spurious currents for VOF methods. The general trend is that temporal discretization error causes drops to stretch too rapidly, while spatial discretization error causes them to stretch too slowly.

ACKNOWLEDGMENTS

This research was sponsored by NSF-INT 9815106, ACS-PRF, NSF-DMS SCREMS 0077177, NSF-DMS 9870220, NSF-CTS 0090381, and the Illinois NCSA under Grants CTS990010N, CTS990059N, and CTS990063N and utilized the NCSA SGI Origin 2000. We thank Stephane Zaleski and Jie Li for the use of SURFER. Acknowledgement is made to the donors of the Petroleum Research Fund, administered by the ACS, for partial support of this research. We thank Vittorio Cristini for providing his boundary integral results for drop deformation.

REFERENCES

1. J. U. Brackbill, D. B. Kothe, and C. Zemach, A continuum method for modeling surface tension, *J. Comput. Phys.* **100**, 335 (1992).
2. A. V. Coward, Y. Renardy, M. Renardy, and J. R. Richards, Temporal evolution of periodic disturbances in two-layer Couette flow, *J. Comput. Phys.* **132**, 346 (1997).
3. W. J. Rider and D. B. Kothe, Reconstructing volume tracking, *J. Comput. Phys.* **141**, 112 (1998).
4. 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).
5. R. Scardovelli and S. Zaleski, Direct numerical simulation of free surface and interfacial flow, *Annu. Rev. Fluid Mech.* **31**, 567 (1999).
6. D. B. Kothe, M. W. Williams, and E. G. Puckett, Accuracy and convergence of continuum surface tension models, in *Fluid Dynamics at Interfaces*, edited by W. Shyy and R. Narayanan (Cambridge Univ. Press, Cambridge, UK, 1998), p. 294.
7. Y. Renardy, M. Renardy, and V. Cristini, A new volume-of-fluid formulation for surfactants and simulations of drop deformation under shear at a low viscosity ratio, *Eur. J. Mech. B/Fluids* **21**, 49 (2002).
8. Y. Renardy, V. Cristini, and J. Li, Drop fragment distributions under shear with inertia, *Int. J. Mult. Flow* **28**, 1125 (2002).
9. M. Meier, G. Yadigaroglu, and B. L. Smith, A novel technique for including surface tension in PLIC-VOF methods, Preprint.
10. I. Ginzburg and G. Wittum, Two-phase flows on interface refined grids modeled with vof, staggered finite volumes, and spline interpolants, *J. Comput. Phys.* **166**, 302 (2001).
11. S. Popinet and S. Zaleski, A front-tracking algorithm for the accurate representation of surface tension, *Int. J. Numer. Methods Fluids* **30**(6), 775 (1999).
12. D. Torres and J. Brackbill, The point-set method: Front tracking without connectivity, *J. Comput. Phys.* **165**, 620 (2000).
13. J. Li, Calcul d'interface affine par morceaux (piecewise linear interface calculation), *C. R. Acad. Sci. Paris* **320**, 391 (1995).
14. V. Cristini, J. Blawzdziewicz, and M. Loewenberg, An adaptive mesh algorithm for evolving surfaces: Simulations of drop breakup and coalescence, *J. Comput. Phys.* **168**, 445 (2001).
15. V. Cristini, S. Guido, A. Alfani, J. Blawzdziewicz, and M. Loewenberg, *Drop Breakup in Shear Flow*, Preprint.
16. S. Zaleski, J. Li, and S. Succi, Two-dimensional Navier-Stokes simulation of deformation and break-up of liquid patches, *Phys. Rev. Lett.* **75**, 244 (1995).
17. S. Zaleski, *Simulation of High Reynolds Breakup of Liquid-Gas Interface*, Lecture Series 1996-2 (Von Kármán Institute for Fluid Dynamcs, 1996).

18. J. Li and Y. Renardy, Direct simulation of unsteady axisymmetric core-annular flow with high viscosity ratio, *J. Fluid Mech.* **391**, 123 (1999).
19. Y. Renardy and J. Li, Comment on 'A numerical study of periodic disturbances on two-layer Couette flow Phys Fluids 10 (12), pp. 3056–3071,' *Phys. Fluids* **11**(10), 3189 (1999).
20. J. Li and Y. Renardy, Shear-induced rupturing of a viscous drop in a Bingham liquid, *J. Non-Newt. Fluid Mech.* **95**(2–3), 235 (2000).
21. J. Li, Y. Renardy, and M. Renardy, Numerical simulation of breakup of a viscous drop in simple shear flow through a volume-of-fluid method, *Phys. Fluids* **12**(2), 269 (2000).
22. Y. Renardy and J. Li, Numerical simulation of two-fluid flows of viscous immiscible liquids, in *Proceedings of the IUTAM Symposium on Nonlinear Waves in Multiphase Flow* (Kluwer Academic, Dordrecht/Norwell, MA, 2000), p. 117.
23. Y. Renardy and J. Li, Parallelized simulations of two-fluid dispersions, *SIAM News* December, p. 1 (2000).
24. J. Li and Y. Renardy, Numerical study of flows of two immiscible liquids at low Reynolds number, *SIAM Rev.* **42**, 417 (2000).
25. M. Renardy, Y. Renardy, and J. Li, Numerical simulation of moving contact line problems using a volume-of-fluid method, *J. Comput. Phys.* **171**, 243 (2001).
26. Y. Renardy and J. Li, Merging of drops to form bamboo waves, *Int. J. Mult. Flow* **27**(5), 753 (2001).
27. J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice Hall, New York, 1983).
28. A. J. Chorin, A numerical method for solving incompressible viscous flow problems, *J. Comput. Phys.* **2**, 12 (1967).
29. J. Li and Y. Renardy, Shear-induced rupturing of a viscous drop in a Bingham liquid, *J. Non-Newt. Fluid Mech.* **95**(2–3), 235 (2000).
30. V. Cristini, J. Blawzdziewicz, and M. Loewenberg, Drop breakup in three-dimensional viscous flows, *Phys. Fluids* **10**(8), 1781 (1998).