JOURNAL OF COMPUTATIONAL PHYSICS 113, 134–147 (1994)

Modelling Merging and Fragmentation in Multiphase Flows with SURFER*

BRUNO LAFAURIE

Laboratoires de géologie, École Normale Supérieure, 24 rue Lhomond, 75231 Paris Cedex 05, France

CARLO NARDONE AND RUBEN SCARDOVELLI[†]

CRS4 (Centre for Advanced Studies and Research in Sardinia), Via N. Sauro, 10, 09123 Cagliari, Italy

STÉPHANE ZALESKI

Laboratoire de Modélisation en Mécanique, CNRS, Université Pierre et Marie Curie, Tour 66, 4 Place Jussieu, 75252 Paris Cedex 05, France

AND

GIANLUIGI ZANETTI

CRS4 (Centre for Advanced Studies and Research in Sardinia), Via N. Sauro, 10, 09123 Cagliari, Italy

Received July 26, 1993

We introduce a new numerical method, called "SURFER," for the simulation of two- and three-dimensional flows with several fluid phases and free interfaces between them. We consider incompressible fluids obeying the Navier–Stokes equation with Newtonian viscosity in the bulk of each phase. Capillary forces are taken into account even when interfaces merge or break up. Fluid interfaces are advanced in time using an exactly volume conserving variant of the volume of fluid algorithm, thus allowing for full symmetry between fluid phases. The Navier–Stokes equation is solved using staggered finite differences on a MAC grid and a split-explicit time differencing scheme, while incompressibility is enforced using an iterative multigrid Poisson solver. Capillary effects are represented as a stress tensor computed from gradients of the volume fraction function. This formulation is completely independent of the topology of interfaces and relatively easy to implement in 3D. It also allows exact momentum conservation in the discretized algorithm. Numerical spurious effects or "parasite currents" are noticed and compared to similar effects in Boltzmann lattice gas methods for immiscible fluids. Simulations of droplets pairs colliding in 2D and in 3D are shown. Interface reconnection is performed easily, despite the large value of capillary forces during reconnection. © 1994 Academic Press, Inc.

I. INTRODUCTION

In many scientific and technical applications it is important to describe and predict the behavior of moving fluid interfaces. Topological changes connected to the processes of merging and break up, as, for instance, in the collision between fluid drops make this problem both interesting and difficult.

In this article we present a new numerical method, based on the volume of fluid method (VOF) of [10], that is able to simulate the behavior of a mixture of immiscible fluids with non-zero surface tension. The novelty of our method consists in how we model surface tension. The simplest idea is to compute the curvature of the interface and then add a force to the fluid momentum balance. We have instead preferred to introduce surface tension as a correction to the momentum stress tensor. This correction is constructed starting from the local gradients of the volume fraction. The resulting scheme is simple to code, computationally efficient, and easy to parallelize. We have implemented our method in both two and three dimensions, and we are currently working on a parallel implementation.

The existing methods for the treatment of interfaces between immiscible fluids fall in two broad categories: front capturing and front tracking. In front capturing methods, a data structure is defined in the entire computational domain (for instance, a concentration, or volume fraction field, C). Evolution schemes for C will capture any discontinuities of C and force them to propagate at the interface velocity. This method has the disadvantage that many errors are accumulated while evolving the field C: the most common is numerical diffusion, which makes the discontinuities representing the interface rapidly disappear. On the other

* Work carried out with the financial contribution of Sardinia Regional Authorities and in fulfillment of Contract 93.1023/A000 of Délégation Générale à l'Armement.

[†] Laboratorio di Ingegneria Nucleare di Monte cuccolino, via dei colli, 16, 40126 Bologna, Italy.

hand, in the front tracking approach a large number of markers follow the position of the interface explicitly. In this category we find volume marker [2] and interface marker [3] methods. Deformable finite element methods [4] may be classified as front tracking methods, where boundary elements follow the discontinuity.

Even though front tracking methods can follow the evolution of a simple interface very accurately, they encounter difficulties in dealing with changes in interface topology. Part of the problem stems from the fact that in these methods one tries to envision and to operate on interfaces as if they were large-scale coherent structures. Moreover, it is intrinsically difficult to write a front tracking method that is able to ensure the respect of global constraints, such as volume and momentum conservation.

The method we present here is local, except for the calculation of the pressure. The other calculations, including that of the surface tension tensor are completely local and can easily be implemented on massively parallel computers.

It should be stressed at this point that we are not modelling in any detail the microscopic interfacial physics [1], which is clearly dependent on intermolecular forces or other microscopic details at the interface such as the presence of surfactant molecules. These phenomena have typical time and length scales far beyond the scope and computational resources of CFD simulations. However, at the points and times of reconnection in an interface simulation such microscopic effects can be relevant. In these cases we think of our correction to the stress tensor as phenomenological.

We will refer to the general method described in this paper as "SURFER." The two essential features of SURFER are the ability to capture interfaces, and a computationally efficient algorithm for surface tension. While for the front capturing scheme we have been mostly inspired by the VOF algorithm of [10], the surface tension algorithm is directly related to what is used in Boltzmann lattice gases [7], to model immiscible fluids. The latter method is a sort of finite difference extension of the lattice gas automaton method described in [6] (see also [5]).

The principal advantage of VOF, compared to other front capturing methods, is the inherent volume conserving nature of the fluxing scheme as its core. Unfortunately, in the original version of VOF this advantage is lost because of the need to eliminate small errors, such as the flotsam generated by advancing interfaces. We will show that, by imposing symmetry between the fluid phases and incompressibility of the velocity field, VOF can be modified so that the most dangerous part of the flotsam is eliminated and exact volume conservation is recovered. To ensure that the incompressibility condition is accurately obeyed, our Navier–Stokes solver makes use of a multigrid method.

The idea for SURFER and the 2D code were developed by Zaleski, following VOF ideas and multigrid ideas of [17]. (We never used, however, the original VOF code). The results of Section 2.4 were obtained by Lafaurie. The 3D implementation of the SURFER technique was by Nardone, Scardovelli, and Zanetti.

In section two we discuss the VOF method and our modifications to it. The third section is dedicated to the Navier–Stokes solver and to the implementation of the surface tension algorithm. The fourth section reports on the first numerical tests of SURFER. In particular we have tested our surface tension algorithm in detail and considered flow examples, i.e., liquid droplet collisions in 2D and 3D that emphasize interface reconnection. The last sections contain conclusions and acknowledgments.

2. THE KINEMATIC PROBLEM

The first task of SURFER is to advance an interface with a prescribed velocity field, a purely kinematic problem. This is done by fluxing the volume fraction from one cell to its neighbors. We first define the basic upwind and downwind fluxing schemes used for that purpose. Then we describe how either of these schemes is selected in a cell. Finally we show the results in some simple cases.

2.1. *Elementary Downwind and Upwind Schemes*

Let us consider the incompressible flow of two immiscible fluids. The divergence-free velocity field $\mathbf{u}(\mathbf{x}, t)$ obeys

$$\nabla \cdot \mathbf{u} = 0. \tag{1}$$

The location of the two fluids is specified with the help of a volume fraction function C, with $C = 1$ inside one fluid and $C = 0$ in the other. We can express volume conservation of the first fluid as

$$\frac{\partial C}{\partial t} + \nabla \cdot (\mathbf{u}C) = 0. \tag{2}$$

In situations where C varies sharply on interfaces the above equation is understood in a weak sense. The description of the problem should not depend on the order used to label the different fluid phases and thus the equation above should be invariant with respect to the transformation $C \to 1 - C$. This is true if and only if Eq. (1) is satisfied. It is therefore important that the numerical scheme used to simulate the time evolution of the fluid phases enforces (1) as strictly as possible. We are currently considering the extension of the method to compressible flow, but in this paper we consider only divergence-free flows.

Our numerical method will represent physical reality by subdividing the domain into square or cubic cells of size h^D. $C^n(\mathbf{x})$ is the volume fraction of the cell centered at $\mathbf{x}$ and occupied by fluid 1 at time t_n. A natural choice for discretization of the velocity field $\mathbf{u}$ is to define it on cell boundaries, i.e., at points $\mathbf{x}_k^{\pm} = \mathbf{x} \pm \frac{1}{2}\mathbf{c}_k$, where $\mathbf{c}_k = h\mathbf{e}_k$ points to the neighboring cell in direction k, and $\mathbf{e}_k$ is a Cartesian unit vector. We describe the volume flux with $\mathbf{f}$. This flux is again discretized on cell boundaries and we adopt the convention that $f_k(\mathbf{x}_k^+)$ represents the outgoing flux of C from the cell at $\mathbf{x}$ to the cell at $\mathbf{x} + \mathbf{c}_k$ across the intermediate boundary.

A conservative method will then be of the form

$$C^{n+1}(\mathbf{x}) = C^n(\mathbf{x}) + \sum_{k=1}^{D} [f_k(\mathbf{x}_k^-) - f_k(\mathbf{x}_k^+)]. \tag{3}$$

It is interesting to discuss first the symmetry between phases 1 and 2 of the *discrete* method. The volume fraction for phase 2 is $C' = 1 - C$. We shall see when we give the full definition of the fluxes that the fluxes f'_k computed from C' obey

$$f'_k = u_k \tau / h - f_k. \tag{4}$$

Replacing C with $1 - C'$ in Eq. (3) yields

$$C'^{n+1}(\mathbf{x}) = C'^n(\mathbf{x}) + \tau \nabla^h \cdot \mathbf{u} + \sum_{k=1}^{D} [f'_k(\mathbf{x}_k^-) - f'_k(\mathbf{x}_k^+)], \tag{5}$$

where $\nabla^h \cdot \mathbf{u} = h^{-1}[\sum_{k=1}^{D} u_k(\mathbf{x}_k^+) - u_k(\mathbf{x}_k^-)]$ is a discrete divergence operator. From Eqs. (3) and (5) we see that the method is symmetric by exchange of phases 1 and 2 if and only if the *discrete* incompressibility condition $\nabla^h \cdot \mathbf{u} = 0$ holds everywhere in the domain.

In general, $f_k(\mathbf{x}_k^+)$ is estimated as a product of $u_k(\mathbf{x}_k^+)$ by a locally determined value of the volume fraction $C(\mathbf{x}_k^+)$. There are many possible ways to define $C(\mathbf{x}_k^+)$ but here we will use either an "upwind" scheme, that is,

$$C^{\text{upwind}}(\mathbf{x}_k^+) = C(\mathbf{x}_k^+ - \varepsilon \mathbf{c}_k/2),$$

or a "downwind" scheme, that is,

$$C^{\text{downwind}}(\mathbf{x}_k^+) = C(\mathbf{x}_k^+ + \varepsilon \mathbf{c}_k/2),$$

where $\varepsilon = \text{sign}(u_k)$.

While the "upwind" scheme is well known (e.g., Ref. [11]) to be stable but diffusive, the "downwind" one is unstable; but it has some desirable properties for interface tracking. Figure 1 shows an ideal interface before and after a downwind step. The front visibly sharpens after the stepping.

Both fluxing schemes have some desirable properties. They are simple and involve only nearest neighbors. Moreover, they both yield the correct speed for interfaces under the assumption that these remain sharp and are nearly flat. Consider a uniform velocity field $\mathbf{u}$. The speed of the interface may be obtained by a classical Rankine–Hugoniot argument [18]. The volume δQ flowing into the interface during time δt across a surface S is $\mathbf{u} \cdot \mathbf{n}\, \delta t S$, where $\mathbf{n}$ is the normal to the surface. Since the volume is conserved, this must also be the increase in volume of phase 1 during time δt. This yields the speed of the interface to be $\mathbf{u} \cdot \mathbf{n}$.

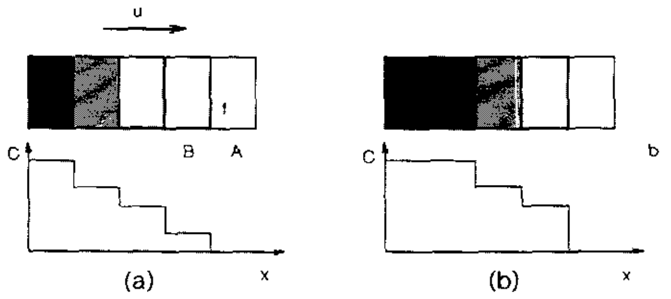

FIG. 1. Two steps of the downwind method. The fluid velocity is from left to right. With respect to cell face f the downwind cell is A and the upwind cell is B. The downwind cell A is empty and thus flux across f is blocked. Thus, the front sharpens in time.

Despite its advantages, the instability of the downwind method may, however, leads to unphysical results, as, for example, in Fig. 2, where the situation resulting from a few more downwind steps is shown. To avoid such pathologies we wil use a somewhat more complex mixture of these two fluxing algorithms.

2.2. *Definition of the Fluxing Schemes*

We will now discuss in more detail how the fluxing schemes are actually implemented. The essential point here is that the fluxes should not be such that they either underflow or overflow the cells.

We will assume for simplicity that $u_k > 0$. The flux computed at point $\mathbf{x}_k^+$, $f(\mathbf{x}_k^+)$, is defined as "incoming" for the cell at $\mathbf{x} + \mathbf{c}_k$ while it is "outgoing" for the cell at $\mathbf{x}$. We define two basic fluxing schemes, the upwind scheme resulting in a flux f^+ and the downwind scheme resulting in a flux f^-.

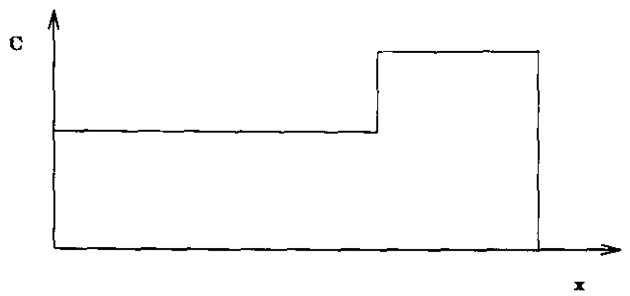

FIG. 2. The fate of the elementary downwind scheme, following the two steps of Fig. 1. The upwind cell tends to overflow.

The flux in the upwind scheme is

$$f(\mathbf{x}_k^+) = f^+(\mathbf{x}_k^+) = [\tau u(\mathbf{x}_k^+)/h]\, C(\mathbf{x}), \tag{6}$$

where τ and h are the time step and mesh size. Thus whenever the CFL condition

$$\tau\, |u|/h < 1 \tag{7}$$

is verified we are guaranteed that we are not out-fluxing more than what is contained in the cell.

On the other hand, in the downwind fluxing scheme

$$f(\mathbf{x}_k^+) = f^-(\mathbf{x}_k^+) = [\tau u(\mathbf{x}_k^+)/h]\, C(\mathbf{x}+\mathbf{c}_k), \tag{8}$$

and therefore we could, in principle, try to extract from the cell at $\mathbf{x}$ more than what it contains. To avoid this we proceed as follows. We first constrain the concentration at $\mathbf{x}+\mathbf{c}_k$ within its legal values, and then we construct the putative flux

$$\tilde{f}^-(\mathbf{x}_k^+) = [\tau u(\mathbf{x}_k^+)/h] \max\{0, \min[1, C(\mathbf{x}+\mathbf{c}_k)]\}. \tag{9}$$

Now, as we discussed above, we want our numerical scheme to be symmetric between the transport of one phase or the other. Thus we have three distinct cases depending on the value of $C(\mathbf{x})$:

$$\begin{aligned} &\alpha: C(\mathbf{x}) < \tilde{f}^-(\mathbf{x}_k^+) \\ &\beta: \tilde{f}^-(\mathbf{x}_k^+) < C(\mathbf{x}) < 1 - \tau u(\mathbf{x}_k^+)/h + \tilde{f}^-(\mathbf{x}_k^+) \\ &\gamma: 1 - \tau u(\mathbf{x}_k^+)/h + \tilde{f}^-(\mathbf{x}_k^+) < C(\mathbf{x}). \end{aligned} \tag{10}$$

When $u < 0$, opposite signs are taken for u and f. From (4) cases α and γ are exchanged by the symmetry $C \to 1 - C$. Moreover, cases α and γ may not simultaneously occur if and only if the CFL condition,

$$|\mathbf{u}|\, \tau/h < 1, \tag{11}$$

is satisfied. In what follows we will always require that this condition is fulfilled.

In case α we limit the flux to what is necessary for a clean sweep of the upwind (at $\mathbf{x}$) cell. In case β the upwind cell is neither too empty nor too full and we leave the fluxes as they are. Finally, case γ is just the symmetric of case α by the exchange of phases 1 and 2. Thus, we will redefine the downwind flux as

$$\begin{aligned} &\alpha: f^-(\mathbf{x}_k^+) = C(\mathbf{x}) \\ &\beta: f^-(\mathbf{x}_k^+) = \tilde{f}^-(\mathbf{x}_k^+) \\ &\gamma: f^-(\mathbf{x}_k^+) = \tau u(\mathbf{x}_k^+)/h - 1 + C(\mathbf{x}). \end{aligned} \tag{12}$$

One way to achieve this in Fortran is similar to the one proposed in Ref. [10]:

```
v=tau * u/h
ftilde=v * cdownwind
f=MIN(cupwind,ABS(ftilde))
f=SIGN(v) * MAX(f,ABS(v)-1+cupwind)
```

with obvious notations.

2.3. *Choice of Upwind and Downwind Schemes*

The front sharpening aspects of the downwind scheme are just what we want when the interface is perpendicular to the flow. However, when the interface is parallel to the flow direction, as schematically represented on Fig. 3, the downwind scheme tends to wrinkle it. In fact, the situation depicted on Fig. 3 is quite similar to the one shown in Fig. 1, but now the front sharpening tends to create spikes on the interface. Thus, we use the downwind scheme whenever propagation is mainly perpendicular to the interface and the upwind scheme whenever it is parallel to the interface. Let the angle with the k direction be

$$\theta_k(\mathbf{x}) = \arccos(n_k), \tag{13}$$

where n_k is the local approximation to the interface normal

$$\mathbf{n} = \nabla^h C / |\nabla^h C| \tag{14}$$

and ∇^h is a finite difference approximation to the gradient operator. We define a critical angle θ_c and use the following definition for the fluxes:

$f_k = f_k^+$, i.e., upwind scheme for $\theta_c < \theta_k$,

$f_k = f_k^-$, i.e., downwind scheme, modified as in 2.2, otherwise.

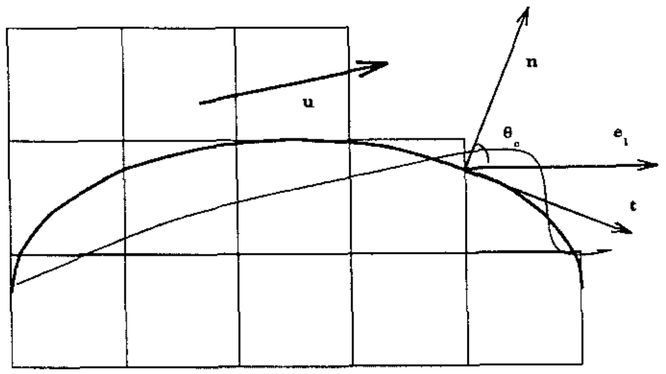

FIG. 3. A configuration where the interface is nearly parallel to the flow and to the horizontal axis. The thick line shows the original configuration while the thin line shows schematically the effect of the downwind scheme.

A problem with this definition of the fluxes is that, in almost uniform regions of a given phase, mixed cells or flotsam are locked in place. This may be noted easily since the downwind scheme results in null fluxes for isolated flotsam. We unlock the flotsam by propagating it with an upwind scheme. We introduce a "flotsam indicator field" $I(\mathbf{x})$ which is 1 when the cell is considered a flotsam and 0 otherwise. Various methods are used for the numerical definition and detection of flotsam. One may, for instance, consider that a cell, all of whose nearest neighbors are of the opposite phase, is flotsam. Then, whenever all first neighbors are either almost full (i.e., $C \approx 1$) or almost empty (i.e., $C \approx 0$), we let $I = 1$; otherwise $I = 0$. This has the effect of treating the homogeneous regions with the upwind scheme, but there both schemes are equivalent. We then have the final definition of the fluxes

$$\begin{aligned} f_k &= f_k^+ \quad \text{for} \quad \theta_c < \theta_k \quad \text{or} \quad I = 1 \\ f_k &= f_k^- \quad \text{otherwise.} \end{aligned} \tag{15}$$

2.4. *Results for Pure Interface Propagation*

While we are unable to prove the stability of our scheme a priori, numerical tests show that thin interfaces are stable and well preserved for adequately chosen critical angles θ_c.

In a first test we fill a square domain of size 0.125×0.125 with phase 1 and attempt to propagate it in a uniform velocity field $(u, v) = (1, -1)$. The results for $\theta_c = 1$ and a 64×64 grid are shown on Fig. 4. Level lines at $C = 0.05$, 0.4,

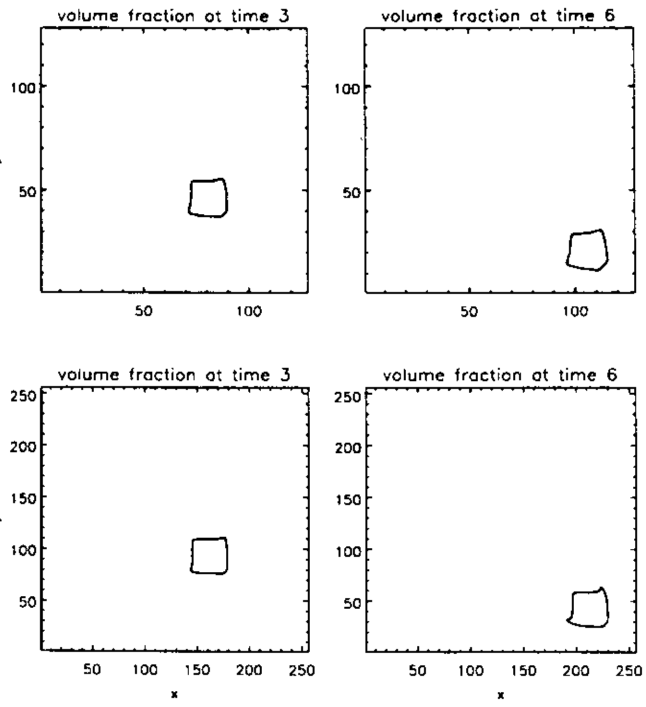

FIG. 5. Propagation of a square cell, 128×128 and 256×256 grids.

0.6, and 0.95 are shown. While propagation creates some damage, it is sufficiently limited. Figure 5 shows results for resolutions 128×128 and 256×256.

The effect of the critical angle is shown on Fig. 6. For a very diffusive (i.e., upwind biased) value of $\theta_c = 0.1$ we observe spreading of the object. For very anti-diffusive

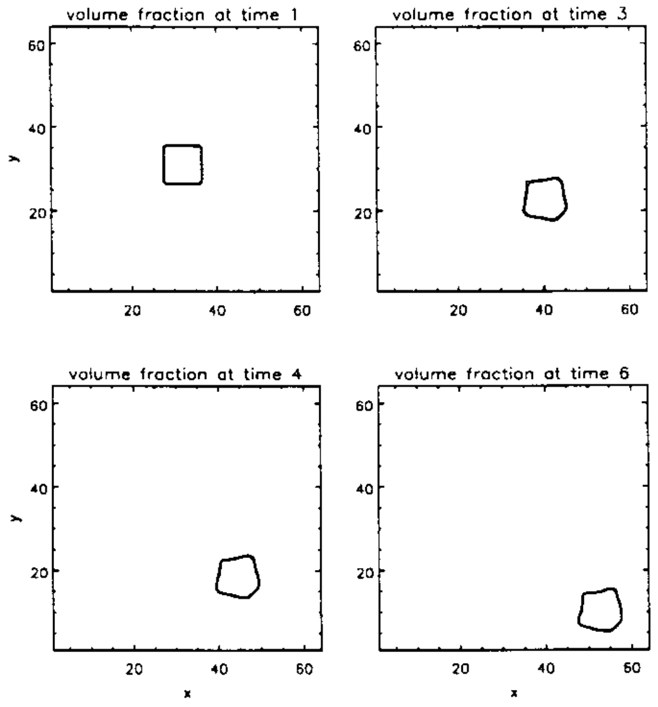

FIG. 4. Propagation of a square cell for $\theta_c = 1$, 64×64 grid.

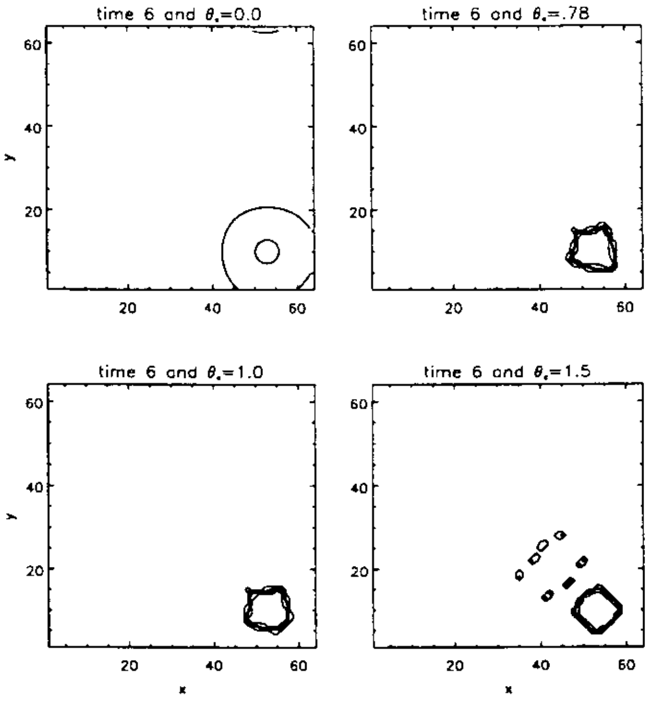

FIG. 6. Effect of the critical angle on propagation.

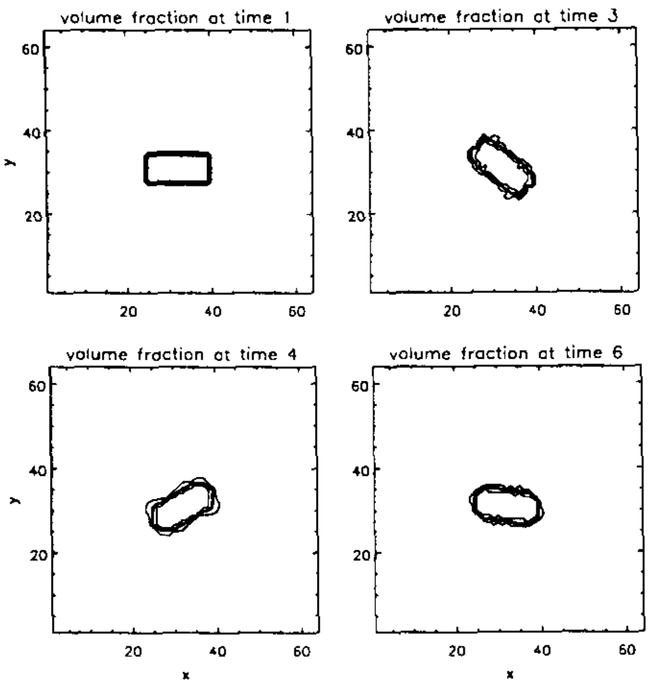

FIG. 7. Rotation of a rectangle.

choices, the front becomes unstable. We found that values of $1 < \theta_c < 1.05$ were optimal.

A more difficult test is the rotation of a rectangle, as performed, for instance, by Chorin for a nine-point scheme in Ref. [9]. Our numerical experiments are shown on Figs. 7 and 8. A larger amount of flotsam is seen on large lattices (Fig. 8).

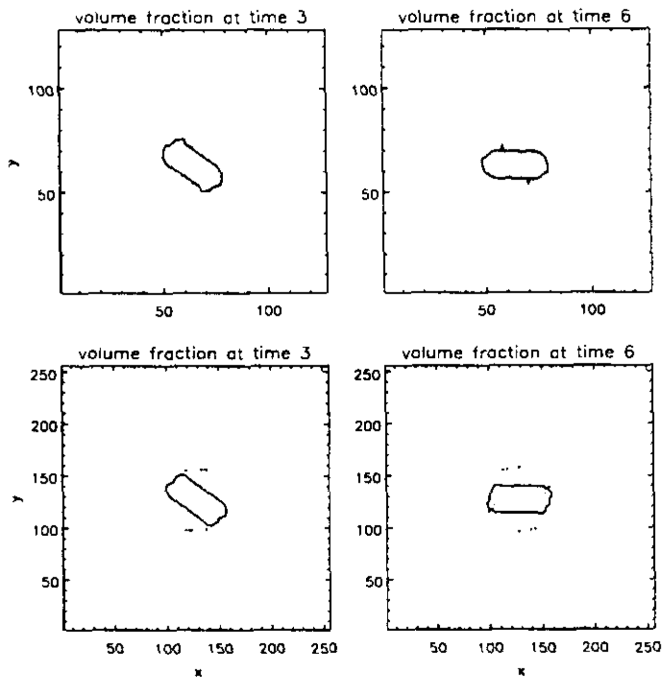

FIG. 8. Rotation of a rectangle—larger grids.

We also performed 3D tests of bubble propagation with results similar to the 2D case.

2.5. *Comparison with* VOF

Our algorithm is similar but not identical to the VOF algorithm of [10]. Comparison between the two methods shows that SURFER has better stability properties. To eliminate flotsam VOF needs to reset periodically the volume fraction to 0 or 1. This type of flotsam would otherwise bring the simulation to a catastrophic end. However, the flotsam in SURFER have a much milder behavior and we thus let them live. Thus SURFER conserves volume exactly. As the discussion below will show, flotsam mostly disappears when a small surface tension is applied in the full SURFER scheme.

3. THE NAVIER–STOKES SOLVER

In many respects the Navier–Stokes part of SURFER is independent of the interface part described in the previous section. We chose to use rather classical finite difference schemes. To bring the two fluid equations as close as possible to the discretisation used in the code, we first effect a transformation that highlights the momentum conserving character. We then describe the two most original parts of the solver: the multigrid algorithm and the surface tension algorithm.

3.1. *Definition of the Problem in Momentum Conserving Form*

We consider viscous incompressible flow with surface tension. For simplicity, we consider interfaces with a constant tension σ. We let the mean curvature of the surface be κ. On the surface we define also the unit normal $\mathbf{n}$. We also define a delta function concentrated on the surface δ_S, defined in more detail in Appendix A. The classical Navier–Stokes equation [13, 16] for incompressible flow with surface tension is

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \nabla \cdot \eta \mathbf{S} + \sigma\kappa\, \delta_S \mathbf{n}, \tag{16}$$

where ρ is the density, η is the shear viscosity, and $\mathbf{S}$ is the rate of strain tensor

$$S_{ij} = \frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}. \tag{17}$$

Together with (16) we require incompressibility

$$\nabla \cdot \mathbf{u} = 0 \tag{18}$$

and the interface motion condition (2). Note that in (16) η

and ρ may vary in space. In fact, they are slaved to the volume fraction C

$$\rho = \rho_1 C + \rho_2(1 - C) \tag{19}$$

$$\eta = \eta_1 C + \eta_2(1 - C). \tag{20}$$

It is possible to rewrite the Navier–Stokes equation in an explicitly momentum conserving form. We first define $\mathbf{\Pi}$ to be the *advection tensor*

$$\mathbf{\Pi} = \mathbf{u} \otimes \mathbf{u}. \tag{21}$$

We also define $\mathbf{T}$, the *capillary pressure tensor*, as

$$\mathbf{T} = -\sigma(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, \delta_S, \tag{22}$$

where $\mathbf{I}$ is the unit tensor δ_{ij}. The fact that the normal is not defined outside the surface may appear to be a problem for the interpretation of this condition. In fact, when we derive the above equation in Appendix A we also give meaning to $\mathbf{n}(\mathbf{x})$ outside the surface S.

In Appendix A we find that the capillary force may be represented as the divergence of $\mathbf{T}$

$$\sigma\kappa \mathbf{n}\, \delta_S = -\nabla \cdot \mathbf{T}. \tag{23}$$

Thus we find the momentum conserving form

$$\partial_t \rho \mathbf{u} = -\nabla p - \nabla \cdot (\rho \mathbf{\Pi} - \eta \mathbf{S} + \mathbf{T}). \tag{24}$$

Note that we also have

$$\mathbf{T} = -\sigma \sum_{k=1}^{D-1} \mathbf{t}^{(k)} \otimes \mathbf{t}^{(k)}\, \delta_S, \tag{25}$$

where the $\mathbf{t}^k$ are $D-1$ orthogonal tangent vectors to the interface. We will return to this definition of $\mathbf{T}$ when we discuss the discretisation of capillary terms.

In the simulations described here we have used either periodic boundary conditions in all directions or free-slip in one direction and periodicity in the remaining directions. For instance, assuming that the free-slip condition is imposed on the $x_3(z)$ axis,

$$u_3 = 0, \qquad \frac{\partial u_1}{\partial x_3} = \frac{\partial u_2}{\partial x_3} = 0 \qquad \text{for} \quad x_3 = 0, L_3, \tag{26}$$

while we impose a mirror condition on the volume fraction

$$\frac{\partial C}{\partial x_3} = 0 \qquad \text{for} \quad x_3 = 0, L_3. \tag{27}$$

3.2. *Discretisation and Multigrid Poisson Solver*

The spatial derivatives are discretized in a classical way on a staggered MAC (marker and cell) grid [11]. In the MAC grid velocity is precisely defined on the faces of the cells, which is just what we want.[1] To ensure incompressibility, we first compute a "provisional" field $\mathbf{u}^*$ from the velocity and volume fraction fields at time t_n:

$$\mathbf{u}^* = \mathbf{u}^n - \tau \left[\nabla^h \cdot \mathbf{\Pi}^h + \frac{1}{\rho} \nabla^h \cdot (\mathbf{T}^h - \eta \mathbf{S}^h) \right]^{(n)}. \tag{28}$$

The discretisation involves classical [11] centered finite difference derivatives ∇^h and a similarly discretized tensor $\mathbf{S}^h$. The calculation of the capillary pressure tensor $\mathbf{T}^h$ will be discussed in what follows. Equation (28) is advanced using a split-explicit technique, the velocity field is first updated with the viscous and capillary tensors, then with the advection term.

We then project the provisional field $\mathbf{u}^*$ on a divergence free field $\mathbf{u}^{n+1}$, using another classical method [11],

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\tau}{\rho^n} \nabla p, \tag{29}$$

where p is chosen so that

$$\nabla^h \cdot \mathbf{u}^{n+1} = 0; \tag{30}$$

we thus ensure that the discrete incompressibility condition is verified for the SURFER scheme. To find p we need to solve the Poisson equation

$$\nabla^h \cdot \left(\frac{\tau}{\rho^n} \nabla^h p \right) = \nabla^h \cdot \mathbf{u}^*. \tag{31}$$

The solution of this equation is made difficult by the fact that ρ may vary sharply and by several orders of magnitude in the domain. However, we found that it could be solved quickly and accurately by a multigrid algorithm. We adapted the "full multigrid" algorithm [14, 15, 17] to our problem. Specifically, our boundary conditions are different from those most frequently used [17]. In order to keep calculations as simple as possible, the interpolation operator [14, 15, 17] is of first order only. Despite this rather drastic simplification the performance of the multigrid method is very good. Usually only a few iterations of the full multigrid method are sufficient.

3.3. *An Algorithm for Surface Tension Inspired by Microscopic Physics*

In this section we give an interpretation of surface tension on mesoscopic scales, which may be defined as follows. On

[1] We do not use any "marker," despite the historical name of the method.

the usual macroscopic scales of fluid mechanics interfaces are represented by infinitely thin surfaces and singular forces, whereas on microscopic scales matter is discrete. We consider an intermediate, mesoscopic scale where matter is continuous but interfaces have a finite thickness; one may then observe a smooth transition from phase 1 to phase 2. We introduce this point of view for numerical reasons. We believe that it can lead to a useful algorithm for the inclusion of surface tension terms such as those discussed above.

To simulate the mesoscopic scale we construct a smoothly varying volume fraction $\tilde{C}$. For instance, we may have

$$\tilde{C}(\mathbf{x}) = \int_V C(\mathbf{x}')\, H(\mathbf{x}-\mathbf{x}'; \varepsilon)\, d\mathbf{x}', \tag{32}$$

where the $H(\mathbf{x}; \varepsilon)$ is a smooth integration kernel. We will let $H \to \delta_S$ as $\varepsilon \to 0$. Such a smoothly varying field has been introduced in the CSF method of Ref. [12]. An approximation to the singular tensor defined in Eq. (22) is then

$$\mathbf{T} = \sigma(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, |\nabla \tilde{C}|, \tag{33}$$

where the vector $\mathbf{n}$ is defined by

$$\mathbf{n} = \frac{\nabla \tilde{C}}{|\nabla \tilde{C}|} \tag{34}$$

as in the method of Ref. [12]; this approximation of the normal converges to the true normal on the interface as the smoothing kernel becomes more concentrated on the interface. Note that in Eq. (14) above we did not use the smoothed function. Our numerical experience does not show improvement when we use (34) instead of (14) in the kinematic part of the code.

A mesoscopic physical interpretation of expressions (33) and (34) may be obtained from the theory of capillarity [19]. Consider an interface orthogonal to the ζ_3 direction with coordinates ζ_1, ζ_2 tangent to the interface. Let the interface be inside the interval $(-a, a)$ with $\tilde{C}$ constant outside. Outside the interface the capillary pressure tensor is null, $T = 0$. Inside the interface region the "normal" pressure is $p_N = p + T_{33}$ and the tangential pressure is $p_T = p + T_{11} = p + T_{22}$. For an interface in thermodynamic equilibrium, $\mathbf{u}$ is constant and uniform and the rate of strain tensor S vanishes. Momentum conservation expressed in (24) implies $p_N = p_0$, where p_0 is the constant pressure outside the interface. From (24) and (25) we obtain

$$\sigma = \int_{-a}^{a} (p_0 - p_T)\, d\zeta_3 \tag{35}$$

which coincides with the classical "mechanical" definition of surface tension. Thus we see that our formulation of the Navier–Stokes equation with interfaces coincides with classical continuum equations with sharp interfaces and with the equilibrium mechanical properties of smooth interfaces. The latter property is shared with the Boltzmann methods for interfacial flow [7].

Our discretisation of the expression (25) is somewhat preliminary. We simply use a discrete equivalent of (33), using the smoothed color field

$$T^h_{ij} = -\sigma \sum_{k=1}^{D-1} t_i^{(k)} t_j^{(k)}\, |\nabla^h \tilde{C}|, \tag{36}$$

where the vectors $\mathbf{t}^{(k)}$ are estimated from a finite difference approximation similar to Eq. (14). We do not give the full details of the computation here, since it may exceedingly complicate the exposition. Several ways of improving on this way of estimating $\mathbf{T}^h$ were tried. Increasing the amount of smoothing of the color fraction before estimating $\mathbf{t}^{(k)}$ removes some of the discretisation errors, as reported in Section 4.2 below. However, we see that good results are obtained even if no smoothing is performed.

3.4. *Implementation of* SURFER

The 2D version of SURFER was written and executed in FORTRAN on a variety of workstations. It is rather simple and short with a total of 1650 lines for the version used in this section. The fastest speed was reached on an IBM RISC (Model 370) and is 6.5×10^4 sites per second. However, increasing the density ratio between the two fluids slows down the computation as more multigrid iterations become necessary. The 2D interface propagation part was also implemented on the CM2 parallel machine.

The 3D version was also written in FORTRAN and ran on several IBM RISC workstations. The code is running at half the speed of its 2D version: 2.4×10^4 sites per second (on an IBM RISC model 550). It is also of comparable length.

4. TESTS OF THE SURFACE TENSION ALGORITHM

Since the truly original part of SURFER is the surface tension algorithm, we paid particular attention to the testing of the capillary effects. A particular pathology that afflicts the surface tension method is the occurrence of parasite currents, which we discuss in detail.

4.1. *Bubble Tests*

In order to verify that our surface tension method yields Laplace's law for equilibrium interfaces, we set up as initial condition a distribution of C close to a bubble of radius R centered in the computational domain. Both fluids have equal densities and viscosities. The simulation is performed in 2D in a 32×32 box and in 3D in boxes of $16 \times 16 \times 16$

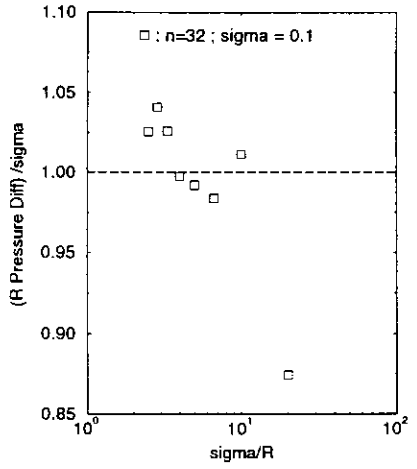

FIG. 9. Observation of Laplace's law in a bubble experiment in 2D. Ratio between the measured pressure difference and its estimate based on the simulation parameters. We plot here the results of 2D simulations.

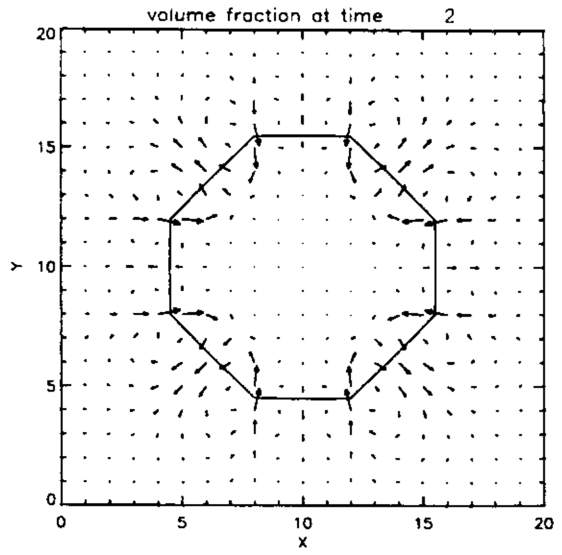

FIG. 11. Parasite currents in a 32 × 32 bubble simulation shortly after the beginning of the run.

and 32 × 32 × 32 lattice sites, and it is characterized by a capillary number computed with mesh size h. The surface tension, viscosity, and box size are all equal to 1.

Laplace's law is well verified. Figures 9 and 10 show that even small bubbles of radius $2h$ are not too distant from the theoretical result. For large bubbles the measured error is around 1 % or lower.

4.2. *Parasite Currents*

Part of the price we have to pay for the locality of our method is the presence of what, we suspect, are unavoidable "parasite" currents. Even a macroscopically static bubble, as the one used in the tests described above, is surrounded by a small amplitude velocity field due to the slight unbalance between the stresses at the sites in the interfacial region. For instance, in Figs. 11 and 12 we consider the relaxation of a bubble from somewhat arbitrary initial conditions to a circular shape. As we can see in Fig. 12 the bubble has relaxed to a circular shape, but it is surrounded by a small amplitude velocity field having the fourfold symmetry of the lattice.

Such "parasite currents" are absent on flat interfaces parallel to the grid axes or making a 45° angle with them. However, they are observed for a generic orientation of a flat interface with respect to grid directions. These parasite currents were also observed in Boltzmann interfacial methods and were discussed in detail in the thesis of Gunstensen [8].

Both in the Boltzmann method and in SURFER, the parasite currents scale with surface tension and viscosity.

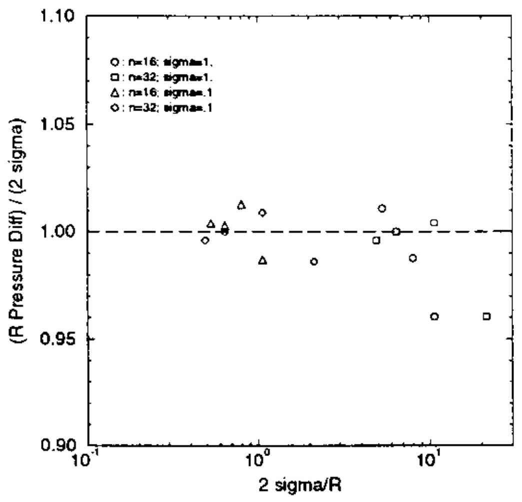

FIG. 10. Observation of Laplace's law in 3D simulations.

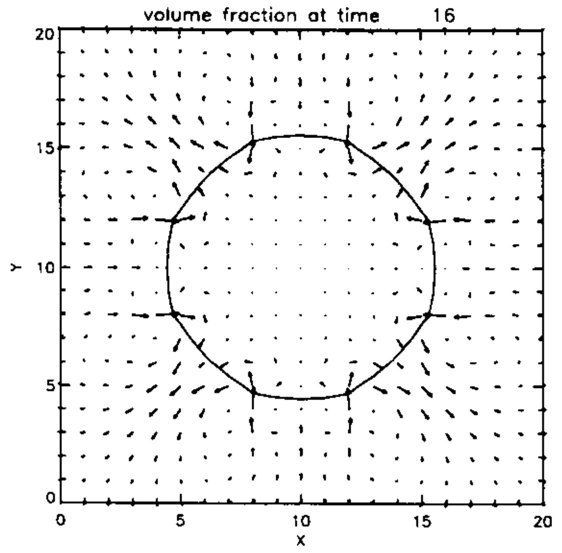

FIG. 12. Parasite currents in a bubble simulation after equilibration.

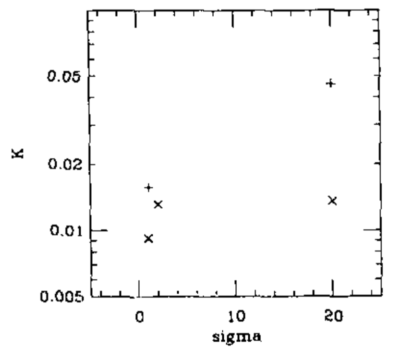

FIG. 13. Dimensionless magnitude $K = u_{\max}\eta/\sigma$ of the parasite currents for various experiments and parameter values.

Dimensional analysis shows that the maximum velocity around a bubble of radius R is given by $u_{\max} = K\sigma/\eta$, where K is some constant. Numerical experiments verify this law with $K \simeq 10^{-2}$. Our numerical experiments on parasite currents are summarized on Fig. 13. There we plot the measured values of K against the dimensionless radius of the bubble R/R_ν, where R_ν is the capillary-viscous length scale of fluid 1 defined by $R_\nu = \rho_1 \nu_1^2/\sigma$. The experiments were repeated for several values of the grid spacing h and radius R. We investigated three orders of magnitude of the ratio R/R_ν. The observed values of K are somewhat fluctuating in time and this explains the scatter of the data in Fig. 13. However, K remains of the same order of magnitude for $R/R_\nu < 10$. The fluctuations of the parasite currents become larger and more irregular as R/R_ν increases. This may be explained by the fact that R/R_ν is proportional to the Reynolds number based on parasite current velocity and bubble size $\mathrm{Re}_K = u_{\max} R/\nu$. As this number becomes large the fluctuations of parasite currents become larger also and eventually interfaces start sputtering debris. Thus there is an intrinsic upper limit on the size of the objects we can simulate with SURFER without generating debris.

On the other hand, for very small values of R/R_ν capillary effects are negligible and parasite currents should have little effect since the dynamics is controlled by viscous forces. The order of magnitude of K is similar for the 2D Boltzmann algorithm and we expect similar limitations to hold there. We are not aware, however, of a systematic study of the dependence of parasite currents on length scales in the Boltzmann method.

The parasite currents may be attenuated by a smoothing of the volume fraction C used to calculate T_{ij} in the right-hand side of (33). The smoothing is realized in 2D by repeated applications of a Laplacian filter $\mathscr{F}$ that transforms C into $\mathscr{F}(C)$ defined by

$$\mathscr{F}(C)(\mathbf{x}) = \tfrac{1}{2}C + \tfrac{1}{8}[C(\mathbf{x}-\mathbf{c}_1/2) + C(\mathbf{x}+\mathbf{c}_1/2) + C(\mathbf{x}-\mathbf{c}_2/2) + C(\mathbf{x}+\mathbf{c}_2/2)]. \tag{37}$$

We then repeat the application of the filter m times yielding $\tilde{C} = \mathscr{F}^{(m)}(C)$. This filter damps small scale variations of C. The measured values of K may be decreased by a factor of 2 or 4 by a few applications of the filter. The optimal number of filters seems to be $m = 1$ or 2. With more filtering, the amplitude of the parasite currents stops decreasing.

4.3. *Capillary Waves*

We set up capillary wave experiments in a series of 2D square boxes of different sizes $N = 1/h$. The initial conditions are zero velocity. The interface is a sine wave with a wavelength equal to the box size. For a small amplitude wave, linearized hydrodynamics predicts a dispersion relation [13],

$$\sigma/\rho = \omega^2/[k^3 \tanh(kL/2)]. \tag{38}$$

We performed a series of experiments and compared the measured ratio on the right-hand side of Eq. (38) to the parameter values. The amplitude of the wave was decreased until no variation in the measured frequency was seen. The results are shown on Fig. 14. The method is seen to converge to the correct value of the frequency.

4.4. *Simulations of Colliding Droplets in 2D*

An example of the ability of our code to simulate the reconnection of interfaces is shown on Figs. 15 and 16. Two droplets of denser fluid are initially sent towards each other. The fluid density ratio is 2. For coarse grids (32 × 32) the droplets simply fuse. For finer grids, a small region of light fluid is trapped in the dense fluid. All parameters are in Tables I and II.

The formation of the trapped region may be explained in the lubrication limit. In that limit a Poiseuille flow forms in the low density channel between the two droplets. To expel the fluid from that layer, the pressure has to be higher in the center. This in turns repels the bubbles away from each other in the center.

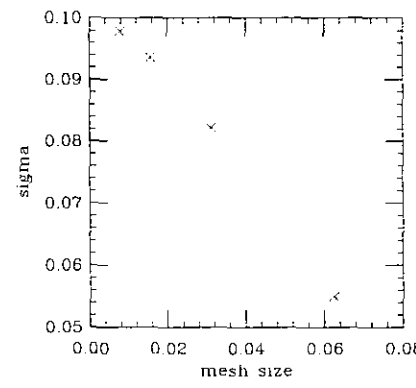

FIG. 14. Measurements of frequency in a capillary wave experiment. The right-hand side of the dispersion relation (36) is plotted versus mesh size h. The theoretical value is 0.1.

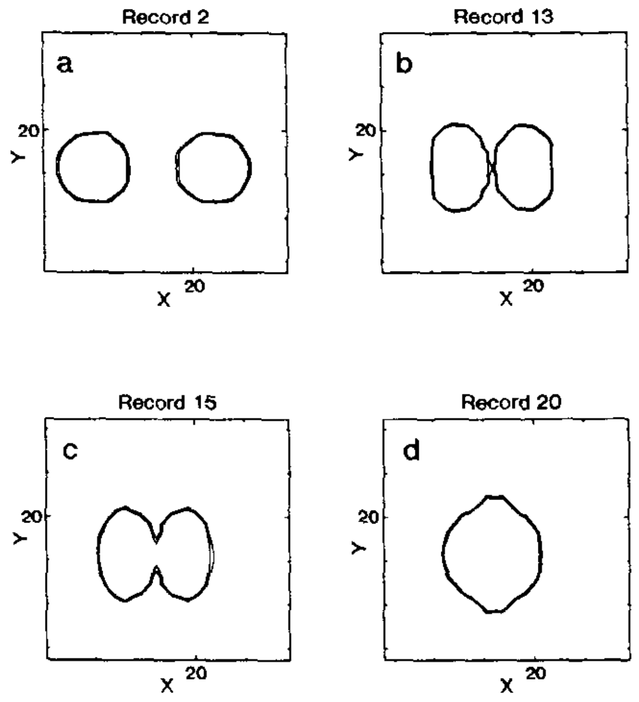

FIG. 15. Collision of two droplets in run A. A coarse (32 × 32) grid is used. The times are, in arbitrary units: (a) 1, (b) 12, (c) 14, and (d) 19.

4.5. *Simulations of Colliding Droplets in 3D*

We have repeated the droplet collisions simulations in 3D. In fact, the three series of images shown in Figs. 17, 18, and 19 are the initial results of a project, dealing with the quantitative validation of SURFER against experimental [21, 22, 20] results, that we have just started. For typographical reasons we can show only some snapshots of these simulations but complete movies will be available on Internet[2] in the near future. The three sequences illustrate drop–drop collision at different Weber numbers and impact parameters. All the parameters and their values are defined in Tables I and II. Run E, the longest simulation, took about 3 h to complete on an IBM RS6000/550. The interface between the two fluid phases is visualized as an isosurface at $C = 0.5$.

The first two sequences (Figs. 17 and 18) show collisions at moderate Weber numbers (13). The first is head-on, i.e., with axisymmetric initial conditions, while the impact parameter of the latter is slightly larger than the droplet radius. As shown in Fig. 17, the two initial droplets coalesce in a single object undergoing large amplitude oscillations that are not, however, large enough to shatter it. These oscillations are eventually dissipated by viscous damping and loss of energy to the surrounding fluid. In run D angular momentum conservation superimposes to the fluctuations a rotation, as can be clearly seen in Fig. 18.

The last case (run E) is qualitatively different. Here the Weber number (133) is large enough so that the final result of the collision is not a single object but rather unbound droplets moving away from each other. On Fig. 19 we can distinguish three main stages in the collision: merging of the two droplets in a single one; transformation of the resulting

TABLE I

Parameters for All Runs

Parameter	Value
D/L^a	0.333
$\mathrm{Re} = (\Delta u)\, D/\nu_1{}^b$	533
ρ_1/ρ_2	2
η_1/η_2	1

[a] D/L is the ratio of the initial diameter of the objects to the box size.
[b] Δu is the relative velocity of the two droplets.

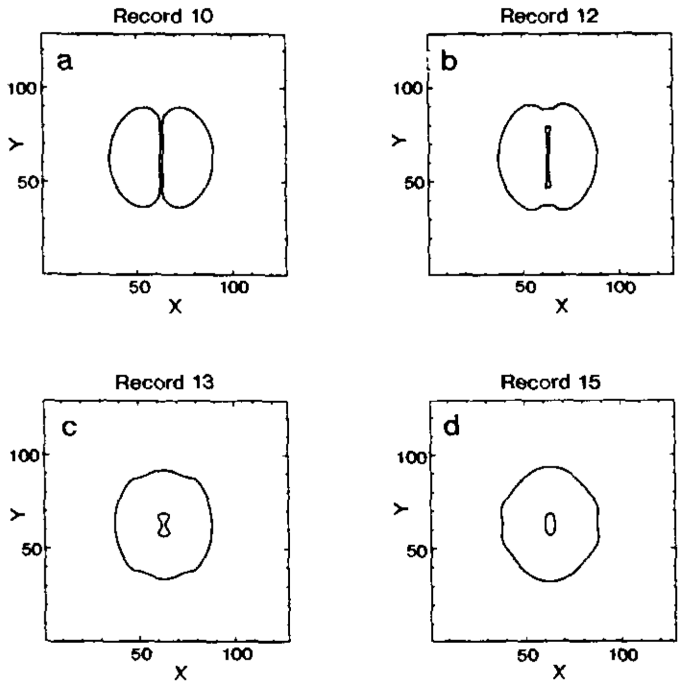

FIG. 16. Droplet collision in run B: (a) 9, (b) 11, (c) 12, (d) 14.

TABLE II

Parameter Values for Each Run

Parameter\Run	A	B	C	D	E
Size	32 × 32	128 × 128	$(32)^3$	$(32)^3$	$(32)^3$
$W = \rho_1(\Delta u)^2\, D/\sigma$	13	13	13	133	133
b/D	0	0	0	0.82	0.27

[2] The movies will be stored as MPEG movies accessible via the CRS4 World Wide Web (WWW) server. The relevant Universal Document Identifier (UDI) address will be: `http://www.crs4.it:/Animate/Animations.html`. WWW software may be retrieved by anonymous FTP from `various ftp sites`.

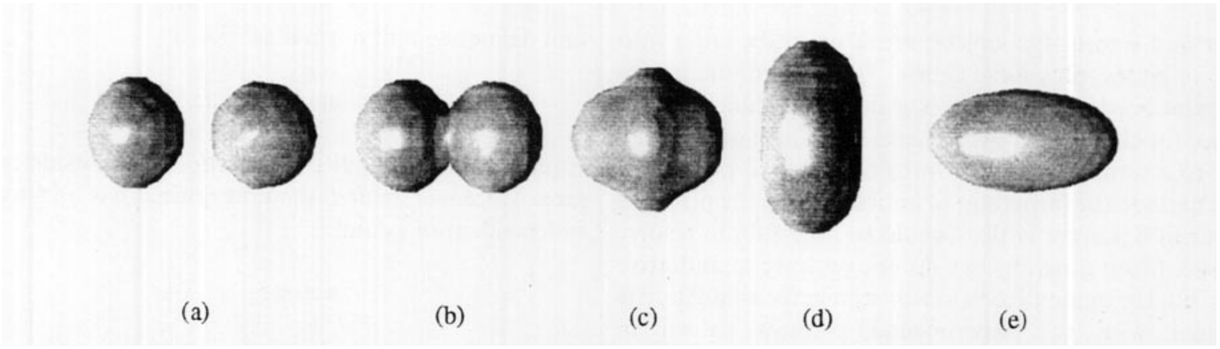

FIG. 17. Droplet collision in run C. Initial conditions are axisymmetric. The snapshots are taken at dimensionless times: (a) 0.30, (b) 0.53, (c) 0.75, (d) 1.05, and (e) 2.1. Times are dimensionless based on the droplet relative velocity Δu and on the droplet diameter D.

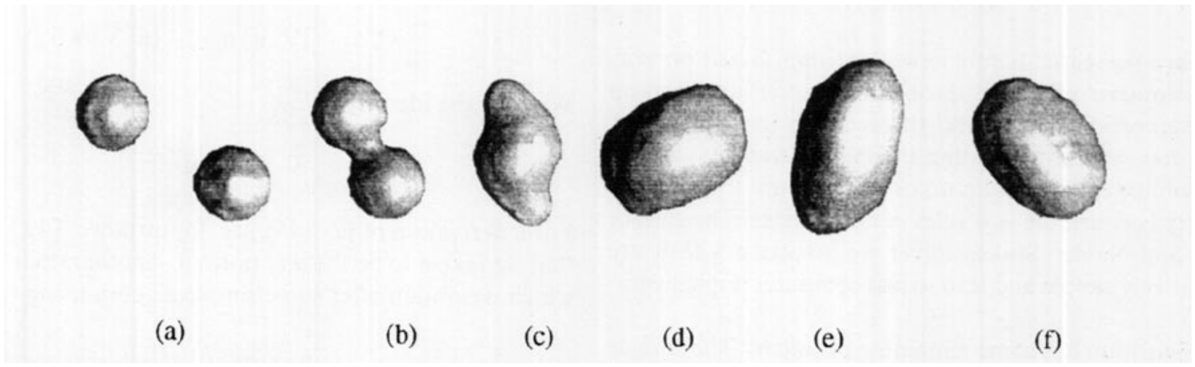

FIG. 18. Droplet collision in run D. The two droplets merge with a non-zero angular momentum. The resulting rotation for the final droplet is clearly seen. Dimensionless snapshots times are: (a) 0.3, (b) 2.0, (c) 2.6, (d) 2.9, (e) 4.4, and (f) 5.9.

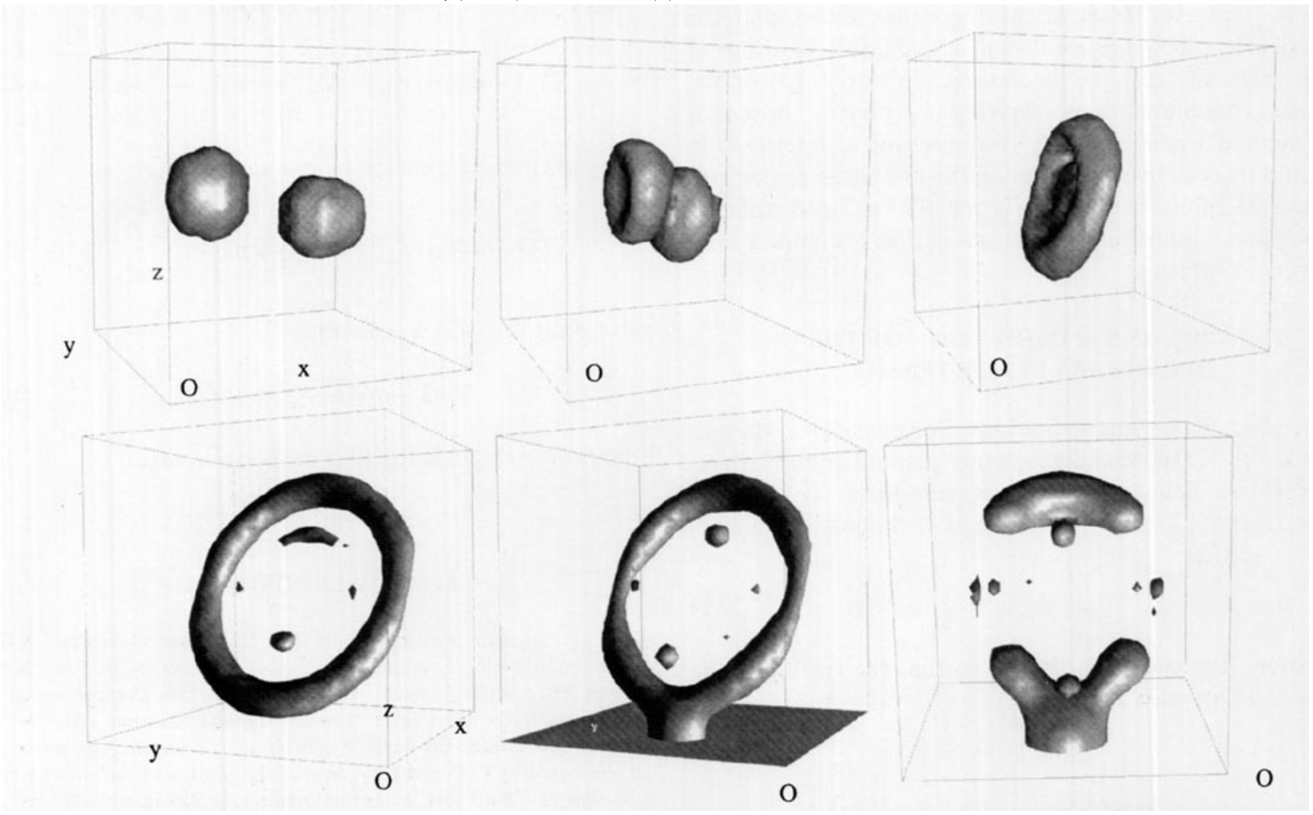

FIG. 19. Droplet collision at large Weber number (run E). We can distinguish three main stages in the collision: merging of the two droplets; transformation of the resulting droplet into a torus-like object; break up of the latter into two main pieces. The snapshots are at times 0.3, 1.1, 2.2, 5.9, 11.9, and 17.9. At time 11.9 we see how the toroidal bubble touches the floor of the simulation box.

droplet into a torus-like object; break up of the latter into two main pieces (plus some debris). Each one of the transitions mentioned above involves a dramatic change of the interface topology: from two spheres to a single one; from a sphere to a torus; and from a torus back to two pieces. We do not exclude the formation of bubbles inside the pieces as seen in run B. Larger lattices would be necessary to resolve this issue. More generally, we do not yet have quantitative results, but the qualitative behavior seen in the simulation is consistent with the experimental pictures shown in [21, 22, 20].

5. CONCLUSION

We have presented here a new algorithm, based on VOF, able to simulate efficiently problems of interface breakup and reconnection in two and three dimensions. This new method has several interesting theoretical features: it conserves volume exactly and it takes into account the inherent symmetry between the two sides of an interface. The surface tension and Navier–Stokes solver we associate with it are also relatively simple and lead to fast computer implementations.

The algorithm has some remaining problems. The surface tension algorithm leads to pathological effects such as parasite currents, especially at high values of the surface tension. However, tests of the algorithm shows that the undesirable effects are small and comparable to those of other methods, such as the Boltzmann lattice gas or the original VOF. Simulations of physical problems show that our method treats correctly reconnection of interfaces in two and three dimensions. The ability of SURFER to perform three-dimensional simulations is one of its main advantages. It thus looks promising for the simulation of complex fluid dynamics problems.

APPENDIX A: DERIVATION OF THE CAPILLARY STRESS TENSOR

We start with some definitions. We consider a smooth surface S in 3D space and a reference point O on the surface. Near O we define a system of curvilinear coordinates $(\xi_1(\mathbf{x}), \xi_2(\mathbf{x}), \xi_3(\mathbf{x}))$. This system is orthogonal and normalised so that

$$\nabla_{\mathbf{x}} \xi_i \cdot \nabla_{\mathbf{x}} \xi_j = \delta_{ij}. \tag{39}$$

Moreover, the system is chosen so that the surface is the locus of points such that

$$\xi_3(\mathbf{x}) = 0. \tag{40}$$

We then define the delta function on the surface as

$$\delta_S(\mathbf{x}) = \delta(\xi_3(\mathbf{x})) \tag{41}$$

and define the unit normal as

$$\mathbf{n}(\mathbf{x}) = \nabla(\xi_3(\mathbf{x})). \tag{42}$$

Equation (42) extends the definition of $\mathbf{n}$ outside the interface. Moreover, near S the unit normal $\mathbf{n}$ is approximately independent of ξ_3 and

$$\mathbf{n} \simeq \mathbf{n}(\xi_1, \xi_2, 0). \tag{43}$$

Then we find that

$$\begin{aligned}\nabla[(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, \delta_S] \\ = \nabla \delta_S - (\nabla \cdot \mathbf{n})\, \mathbf{n}\, \delta_S - (\mathbf{n} \cdot \nabla)(\mathbf{n}\, \delta_S).\end{aligned} \tag{44}$$

We note the identity

$$\nabla \cdot \mathbf{n} = -\kappa. \tag{45}$$

For a derivation of Eq. (45), see, for instance, [12] but note that we take $\mathbf{n}$ to be the *unit* normal. Another useful identity which we obtain after some simple algebra using (39) is

$$\mathbf{n} \cdot \nabla = \partial / \partial \xi_3. \tag{46}$$

Using (44), (45), and (46) one obtains

$$\nabla[(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, \delta_S] = \nabla \delta_S + \kappa \mathbf{n}\, \delta_S - \frac{\partial}{\partial \xi_3}\, \mathbf{n}\, \delta_S. \tag{47}$$

Using (43) we let $\mathbf{n}$ escape from under $\partial / \partial \xi_3$,

$$\nabla[(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, \delta_S] = \nabla \delta_S + \kappa \mathbf{n}\, \delta_S - \mathbf{n} \frac{\partial}{\partial \xi_3}\, \delta_S, \tag{48}$$

and from (39), (42) we obtain

$$\nabla[(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\, \delta_S] = \kappa \mathbf{n}\, \delta_S. \tag{49}$$

Equation (49) yields Eq. (23) in the main text.

ACKNOWLEDGMENTS

The authors acknowledge support from the Region of Sardinia. S. Z. wishes to acknowledge support from Société Européenne de Propulsion and the "Programme de Recherge Coordonnées sur la Combustion dans les moteurs fusées" of Centre National d'Étude Spatiales and CNRS. A travel grant from the NATO collaborative research programme is acknowledged by S. Z. and G. Z. Part of this work was performed while S. Z. was at CNRS and at the Laboratoire de Physique Statistique of École Normale Supérieure, Paris. Support from DRET is gratefully acknowledged. Part of the work was executed at the Centre National de Calcul Parallèle of Institut de Physique du Globe, Paris, which we gratefully acknowledge for the grant of Connection Machine time. S. Z. wishes to thank X. Calimez, J. F. Estrade, F. X. Keller, and J. Li for their very efficient collaboration in testing and using the two-dimensional version of SURFER. We had interesting and helpful conversations with C. Bruno, S. Candel, P. Constantine, D. d'Humières, L. P. Kadanoff, B. Nadiga, A. Quarteroni, D. H. Rothman, and D. Vandromme.

REFERENCES

1. J.-X. Yang, J. Koplik, and J. R. Banavar, *Phys. Rev. Lett.* **67**, 3539 (1991).
2. B. J. Daly, *Phys. Fluids* **10**, 297 (1967).
3. J. M. Hyman, *Physica D* **12**, 396 (1984).
4. J. Glimm, O. McBryan, R. Menikhoff, and D. H. Sharp, *SIAM J. Sci. Stat. Comput.* **7**, 230 (1986).
5. B. Boghosian, *Comput. Phys.* Nov/Dec, 585 (1991).
6. D. H. Rothman and J. M. Keller, *J. Stat. Phys.* **52**, 1119 (1988).
7. A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti, *Phys. Rev. A* **43**, 4320 (1991).
8. A. K. Gunstensen, Ph.D. thesis, MIT, June 1992 (unpublished).
9. A. J. Chorin, *J. Comput. Phys.* **35**, 1 (1980).
10. C. W. Hirt and B. D. Nicholls, *J. Comput. Phys.* **39**, 201 (1981).
11. R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow* (Springer-Verlag, New York/Berlin, 1983).
12. J. U. Brackbill, D. B. Kothe, and C. Zemach, *J. Comput. Phys.* **100**, 335 (1992).
13. S. Chandrasekhar, *Hydrodynamic and Hydromagnetic Stability* (Dover, New York, 1981).
14. W. L. Briggs, *A Multigrid Tutorial* (SIAM, Philadelphia, 1987).
15. A. Brandt, "Guide to Multigrid Development," in *Multigrid Methods*, edited by W. Hackbush and U. Trottenberg (Springer-Verlag, Berlin, 1982).
16. G. K. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge Univ. Press, Cambridge, UK, 1970).
17. W. H. Press and S. A. Teukolsky, *Comput. Phys.* Sep/Oct, 514 (1991).
18. G. B. Whitham, *Linear and Nonlinear Waves* (Wiley-Interscience, New York, 1974).
19. J. Rowlinson and B. Widom, *Molecular Theory of Capillarity*, Clarendon, Oxford, 1982.
20. E. Becker, W. J. Hiller, and T. A. Kowalewski, *J. Fluid Mech.* **231**, 189 (1991).
21. A. Menchaca-Rocha, Preprint, Istituto de Fisica, Universidad Nacional Autonoma de Mexico, 1993.
22. R. W. Park, Ph.D. thesis, Chemical Engineering, University of Wisconsin, Madison, 1970 (unpublished).