from *SIAM News*, Volume 33, Number 10

Parallelized Simulations of Two-fluid Dispersions

By Yuriko Renardy and Jie Li

APPLICATIONS ON ADVANCED ARCHITECTURE COMPUTERS

Greg Astfalk, Editor

Two-fluid dynamics is a challenging subject, rich both in physics and in practical applications, which range from manufacturing to lubricated transport. Mechanisms unique to a flow can be exploited to enhance functionality in applications. For example, density-matching can be used to depress the effect of gravity, or of centripetal acceleration in rotating systems, making it possible to manipulate the locations occupied by the fluid, as well as the shapes of the interfaces between the fluids. Viscosity segregation can be used to promote mixing and demixing, and thus the displacement of one fluid by another (as in the problem of oil recovery), or to segregate one molten plastic from another by encapsulation. The lubrication of one (highly viscous) fluid by another (less viscous) fluid is a particularly important branch of two-fluid dynamics.

Many configurations are possible for the flow of two immiscible fluids: layers, fingers, encapsulated regimes, and drops. One of the difficulties in the study of flows of two immiscible fluids is that the domain contains an interface. The interface moves under the effects of the flow and can sometimes undergo severe deformations, including breakup. In the numerical treatment of these flows, we must answer three questions: (1) How do we represent the interface on a mesh? (2) How will the interface evolve in time? (3) How should we apply the boundary conditions on the interface?

There are many interface-tracking methods, such as the moving-grid method, the front-tracking method, the level-set method, and the volume-of-fluid (VOF) method. The VOF method provides a simple way to treat the topological changes of the interface, as well as ease of generalization to the three-dimensional case. The interested reader is referred to a recent review article [4] and its references.

Our two-fluid code is composed of three parts: a second-order VOF method for tracking the interface, a projection method for solving the Navier–Stokes equations on the MAC grid (to be defined later), and, finally, a continuum method for modeling the interfacial tension.

The Equations of Motion

In our two-fluid model, the density ρ and viscosity μ of each fluid are constant, but a jump across an interface is possible. A concentration, or color, function C is used to represent and track the interface:

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

This concentration function is transported by the velocity field $\mathbf{u}$. The average values of the density and viscosity are given by $\rho = C\rho_1 + (1-C)\rho_2$ and $\mu = C\mu_1 + (1-C)\mu_2$, where the subscripts refer to fluids 1 and 2.

We assume that the flow is incompressible, i.e., $\nabla \cdot \mathbf{u} = 0$, and governed by the Navier–Stokes equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot\mu\mathbf{S} + \mathbf{F}$$

Here, $\mathbf{S}$ is the viscous stress tensor:

$$\mathbf{S}_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)$$

The body force $\mathbf{F}$ includes the gravity and interfacial tension force. The interfacial tension force $\mathbf{F}_s = \sigma\kappa\mathbf{n}_S\,\delta_S$, where σ is the interfacial tension, κ is the mean curvature, and $\mathbf{n}_S$ is the normal to the interface.

Temporal Discretization and Projection Method

The solution of the large system of equations resulting from discretization of the governing equations is very costly. This is especially true in three spatial dimensions. For an efficient approximation, a projection method can be used to decouple the solution of the Navier–Stokes equations from the solution of the continuity equation.

In the projection method, the momentum equations are first solved for an approximate $\mathbf{u}^*$ without the pressure gradient, with the velocity field $\mathbf{u}^n$ assumed to be known at a given time-step:

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

In general, the intermediate flow field $\mathbf{u}^*$ does not satisfy the incompressibility equation. It is corrected by the pressure

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

so that a divergence-free velocity $\mathbf{u}^{n+1}$ will be produced at the next time-step. The pressure field is not known, but when the divergence of the equation is taken, it is found to satisfy a Poisson equation:

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

In the problems addressed here, the boundary conditions for the velocity are periodicity and the Dirichlet condition. Analogously, the boundary conditions for the pressure are periodicity and the Neumann condition, respectively.

Spatial Discretization and Interface Advection

The spatial discretization of the momentum equation and the Poisson equation for the pressure is based on a finite-difference scheme known as the MAC method. An Eulerian mesh of rectangular cells is used, with variable sizes in all three dimensions. The pressure and the concentration function C are given at the node at the center of each rectangular cell. The three velocities are given at the centers of the faces of the cell.

At the discrete level, the concentration function is the volume fraction field C_{ij}. When a cell is filled by fluid 1, $C_{ij} = 1$; when a cell contains no fluid 1, $C_{ij} = 0$. The interfaces are in the cells for which C_{ij} is between 0 and 1.

Given an interface, we can calculate a unique volume fraction field. When we represent the interface by a volume fraction field, however, we lose interface information and cannot determine a unique interface. The interface should be reconstructed. Piecewise linear interface calculation (PLIC) methods have been developed for two- and three-dimensional cases. The essence of these methods is to calculate the approximate normal $\mathbf{n}$ to the interface in each cell; this determines a unique linear interface with the volume fraction of the cell. The discrete gradient of the volume fraction field yields $\mathbf{n} = \nabla^h C/|\nabla^h C|$.

The second step of the VOF method is to evolve the volume fraction field C. The Lagrangian method is the natural choice for interface evolution. In this scheme, once the interface has been reconstructed, the velocity at the interface is interpolated linearly. The new interface position is then calculated via $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{u}(\Delta t)$. Figure 1 illustrates the performance of the Lagrangian method on an arbitrary two-dimensional mesh. In comparison with the Eulerian method, the Lagrangian method has two advantages: (1) When the Courant condition (Max $|\mathbf{u}|$) $\Delta t/h <$ 1/2 is satisfied, the method is stable; and (2) the volume fraction always satisfies the physical constraint $0 \le C \le 1$.

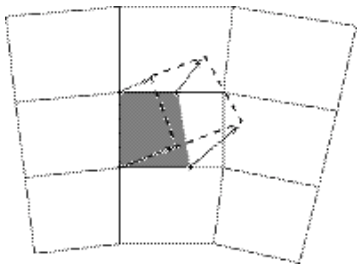

Figure 1. *The Lagrangian method on an arbitrary two-dimensional mesh. The shaded polygon represents the portion of the central cell occupied by the fluid. The broken line shows the polygon position after advection in the local velocity field (represented by arrows). The fluid is redistributed between neighboring cells, which are partially overlapped by the new polygon.*

In the VOF method the interfacial tension condition across the interface is not applied directly, but rather as a body force over the cells that contain the interface. Two such formulations have been implemented in this work. The first is the continuous surface force formulation, in which $\mathbf{f}_s = \sigma\kappa\mathbf{n}_S$ and $\mathbf{F}_s = \mathbf{f}_s\nabla C$. The second is the continuous surface stress formulation, in which $\mathbf{F}_s = \nabla\cdot\mathbf{T} = \sigma\delta_S\kappa\mathbf{n}_S$ and $\mathbf{T} = [(\mathbf{1}-\mathbf{n}_S\otimes\mathbf{n}_S)\sigma\delta_S]$. The latter approach leads to a conservative scheme for the momentum equation.

Semi-implicit Stokes Solver

The characterization of our numerical method is now complete, except for one weakness that needs to be considered: Being an explicit method, it is not suitable for simulations of low-Reynolds-number flows. For an explicit method, the time-step Δt should be smaller than the viscous time scale, $T_\mu = \rho h^2/\mu$, where h denotes the mesh size. This stability limit is much more restrictive than the CFL condition for simulations of low-Reynolds-number flows. In calculations for times of $O(1)$, implicit treatment of the viscous terms is im-perative.

Our approach is an original semi-implicit method. The time integration scheme is constructed to be implicit for the Stokes operator, and explicit otherwise. In the u component of the momentum equation, we treat only the terms related to u (those with upper index *) implicitly, leaving the others (those with upper index ^{n}) in the explicit part:

$$\begin{aligned}
&\frac{u^*-u^n}{\Delta t}=\left(\mathbf{u}^n\cdot\nabla\right)u^n\\
&+\frac{1}{\rho^n}F_1^n+\frac{1}{\rho^n}\frac{\partial}{\partial x}\left(2\mu^n u_x^*\right)\\
&+\frac{1}{\rho^n}\frac{\partial}{\partial y}\left(\mu^n u_y^*+\mu^n v_x^n\right)\\
&+\frac{1}{\rho^n}\frac{\partial}{\partial z}\left(\mu^n u_z^*+\mu^n w_x^n\right)
\end{aligned}$$

This can be expressed as:

$$\left[I - \frac{\Delta t}{\rho}\left(\frac{\partial}{\partial x}\left(2\mu \frac{\partial}{\partial x} \right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial}{\partial y} \right) + \frac{\partial}{\partial z}\left(\mu \frac{\partial}{\partial z} \right) \right) \right] u^* = \text{explicit terms}$$

This procedure decouples the u component from the previous parabolic system. It is still a first-order method, but the effort needed to solve it is significantly less than for the fully implicit treatment. The same idea applies to the other velocity components.

The viscous terms in our semi-implicit scheme are unconditionally stable. In addition, the left-hand side of the equation is factorized as:

$$\left[I - \frac{\Delta t}{\rho}\left(\frac{\partial}{\partial x}\left(2\mu \frac{\partial}{\partial x} \right) \right) \right] \left[I - \frac{\Delta t}{\rho}\left(\frac{\partial}{\partial y}\left(\mu \frac{\partial}{\partial y} \right) \right) \right] \left[I - \frac{\Delta t}{\rho}\left(\frac{\partial}{\partial z}\left(\mu \frac{\partial}{\partial z} \right) \right) \right] u^* = \text{explicit terms}$$

The error of this factorization is $O(\Delta t^3)$. Inversion of the left-hand side of the preceding equation requires only the solution of tridiagonal matrices, which results in significant reductions in computation and memory. Moreover, the scheme involves the solution of a large number of tridiagonal systems, which are independent of each other and can therefore be solved in parallel. The solution of these tridiagonal systems can be done in $O(N)$ operations, where N is the number of grid points. In summary, this scheme is first-order accurate and unconditionally stable. This stability is crucial for the simulation of low-Reynolds-number flow.

Simulation of Drop Breakup in 3D

Study of the deformation and breakup of a drop in shear flow lies at the foundation of dispersion science and multiphase flow. The topic has been evoking the interest of the scientific community since the time of Taylor [5]. More recently, experimental observations of sheared breakup have been recorded [3]: A strong shear is applied to a single drop; the drop elongates, followed by the pinching off of drops at the ends. Such processes, with their yield of daughter drops, are paradigms for theoretical investigations of emulsification and mixing.

The experimental work in [3] focused on a viscous drop suspended in a second immiscible liquid, the matrix liquid, in a cylindrical Couette device. The difference in density between the two liquids is a minor effect, and the flow is sufficiently slow that centrifugal effects in the cylindrical device are not important. A corresponding theoretical model is simply three-dimensional Couette flow with zero gravity; the matrix liquid is undergoing a simple shear flow between two parallel plates, separated by a distance d^*. The liquid drop has an undeformed radius a and viscosity μ_d, and the matrix liquid has viscosity μ_m. The undisturbed velocity field $\mathbf{u} = \dot{\gamma} z \mathbf{i}$, where $\dot{\gamma}$ is the imposed shear rate.

Additional parameters for our numerical simulations are the interfacial tension σ and the spatial periodicities λ_x^* and λ_y^* in the x and y directions, respectively. There are six dimension-less parameters: a capillary number $Ca = a\dot{\gamma}\mu_m\sigma$, where an average shear rate is defined as $\dot{\gamma} = U^* d^*$, the viscosity ratio $\lambda = \mu_d\mu_m$, the Reynolds number $Re = \rho_m \dot{\gamma} a^2 \mu_m$, the dimensionless plate separation $d = d^{*/a}$, and dimensionless spatial periodicities $\lambda_x = \lambda_{x/a}^*$ and $\lambda_y = \lambda_{y/a}^*$. The shear stress induced by the flow competes with the interfacial tension to deform and rupture the drop. The capillary number denotes the ratio between these two competing effects and provides a useful measure of the efficiency of the shear flow in deforming the drop.

Recently, highly accurate computations of drop deformation and breakup for Stokes flow have been performed [1]. We present a numerical exploration of breakup. Periodic conditions are imposed in the x and y directions (streamwise and cross-flow, respectively). The top and bottom plates have constant velocities, and the shear rate is therefore constant during the entire computation.

When the shear rate is increased past a critical value, the drop ruptures. For the case of equal viscosities and densities, the critical capillary number is approximately 0.41. When the capillary number exceeds the critical value, the drop ruptures. Below the critical value, the drop attains a steady unruptured state.

Our computation for $Ca = 0.42$ in a dimensionless $3 \times 1 \times 2$ box, with a $96 \times 32 \times 64$ mesh, is shown in Figure 2. The competition between the externally imposed shear flow and the surface tension-driven flow is clearly evident. The most noticeable initial motion is elongation of the drop, stretched by the viscous shear stress of the external flow (for times less than 20). At time $T = 30.0$, formation of a waist is seen near the center of the drop, and the drop is steadily thinning. As the drop slowly lengthens, a visible neck is seen near the bulbous end. Because of this neck, the ends will eventually pinch off and the liquid thread remaining in the middle will form small satellite droplets.

The deformation and breakup of a drop in shear flow were recently investigated experimentally [3]. Figure 3 shows some of the experimental observations. Figure 4 shows our simulation for parameters close to the experimental case. The largest daughter drops are formed by the pinching off at the elongating end, as shown in both of these figures. Subsequently, there is a sequence of small,

and then larger drops, as shown in both of these figures. The reader is referred to [2] for more details.

Parallelization

For transient 3D Navier–Stokes simulations like those of drop breakup, it is necessary to use parallel computers. When coding, we essentially use large Fortran `DO` loops, which are made to step through the array in the same way the array is laid out in memory. Using the system routine `/bin/time`, we profiled the code to find the time-critical components and, through tuning, to improve the efficiency. We have parallelized the entire code on the Origin2000, using the `C$DOACROSS` directive. For the code dealing with explicit schemes, which consists of independent loops, there is no difficulty in the parallelization. It is not easy to break the data dependencies in the solution of tridiagonal systems. However, there are many such systems to be solved, and we parallelize this component by solving them simultaneously.

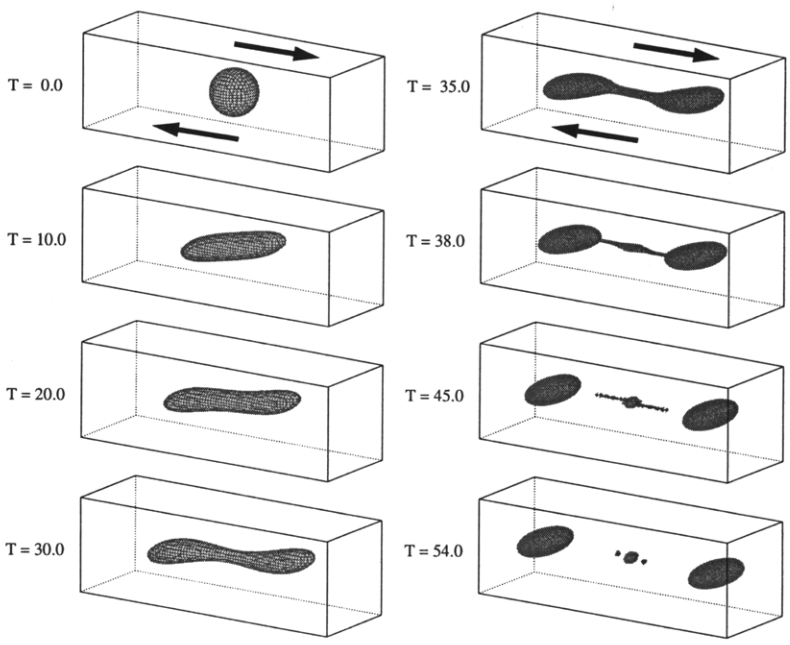

Figure 2. *Evolution of a drop shape for Ca = 0.42 in a dimensionless 3 × 1 × 2 domain, with equal viscosities and densities.*

The numerical solution of the pressure equation is the most time consuming part of our Navier–Stokes solver; consequently, an efficient solution is crucial for overall performance. The multigrid method is known to be an optimal choice. The basic idea of the multigrid method is to combine two complementary procedures: one basic iterative method to reduce the high-frequency error, and one coarse-grid correction step to eliminate the low-frequency error. Our most challenging task has been to parallelize this multigrid solver. We have chosen a two-color Gauss–Seidel iterative method because it breaks the dependencies between the variables and therefore allows for parallelization of the scheme. A Galerkin method provides a good coarse-grid correction.

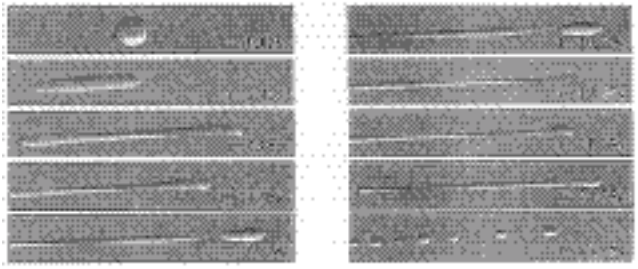

Figure 3. *Reproduction from Marks (1998), showing a typical drop breakup. Here, the viscosities are 7.0 Pa.s for the matrix liquid and 4.3 Pa.s for the drop; the interfacial tension is 10.7 mN/m, the initial drop radius 0.048 cm, and the shear rate 2.17/s; densities are equal.*

On the finite grid, the discretization for the Poisson equation has the form:

$$\begin{aligned} p_{i,j,k} &= h_{i,j,k} \\ &- a_{i,j,k}\,p_{i-1,j,k} - b_{i,j,k}\,p_{i+1,j,k} \\ &- c_{i,j,k}\,p_{i,j-1,k} - d_{i,j,k}\,p_{i,j+1,k} \\ &- e_{i,j,k}\,p_{i,j,k-1} - f_{i,j,k}\,p_{i,j,k+1} \end{aligned}$$

The Jacobi iteration method used to solve this system is

$$\begin{aligned} p^{n+1}_{i,j,k} &= h_{i,j,k} \\ &- a_{i,j,k}p^{n}_{i-1,j,k} - b_{i,j,k}p^{n}_{i+1,j,k} \\ &- c_{i,j,k}p^{n}_{i,j-1,k} - d_{i,j,k}p^{n}_{i,j+1,k} \\ &- e_{i,j,k}p^{n}_{i,j,k-1} - f_{i,j,k}p^{n}_{i,j,k+1} \end{aligned}$$

In this form, the iteration is easily parallelized because the components $p^{n+1}_{i,j,k}$ can be computed independently of each other. In the Gauss–Seidel iteration, on the other hand, the components of p^{n+1} are computed successively; in the computation of each new component, the already computed components of p^{n+1} are used on the right-hand side of the equation instead of p^n. Algorithms of this type cannot be parallelized. The Gauss–Seidel iteration converges much faster than the Jacobi iteration.

In our calculations, we used a two-color scheme, with grid points divided into two categories according to whether $i + j + k$ is even or odd. The right-hand side of the above equation involves only the values of p^n at points with parity opposite to that of (i,j,k). Hence, we can use a two-step method, first updating the values at "odd'' points and then, using the previously updated values at

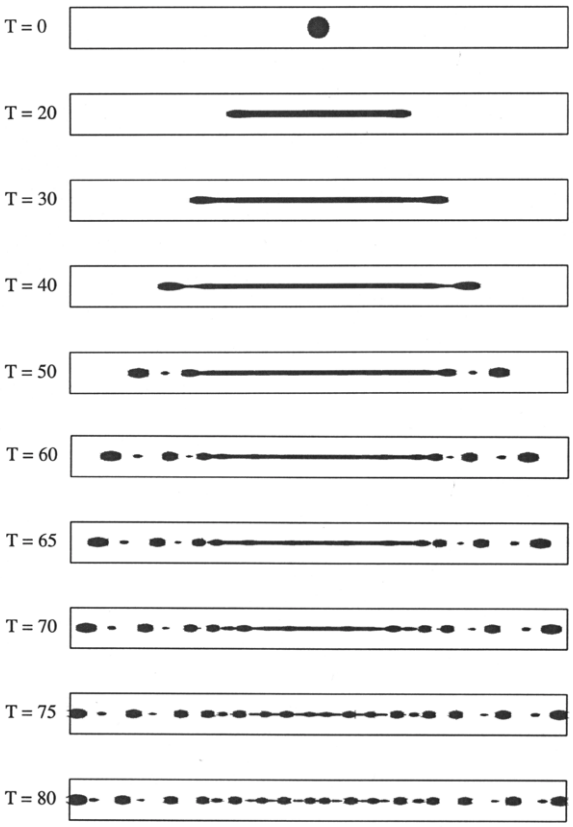

Figure 4. *Interface evolution as viewed from the top of a 12 × 1 × 1 computational box during breakup for Ca = 0.55, λ = 0.77, and equal densities. The grid is 256 × 64 × 64.*

the odd points, updating the values at "even'' points. For each of the two substeps, the calculations at different points are independent of each other and can therefore be carried out in parallel.

The three main components of the code are the interface-tracking scheme, the projection method for solving the Navier–Stokes equations (semi-implicit and explicit terms in time integration, multigrid pressure solver), and the continuous surface force or, alternatively, the continuous surface stress algorithm. All the components have been parallelized. The algorithms that account for the major portion of the computation time are the multigrid scheme for the pressure solver and the semi-implicit scheme for time integration.

We have used up to 32 processors and 512 megabytes of physical memory to run our parallelized code. Large storage capacity is preferable but not essential. Once the code has been parallelized, an important measure of performance is scalability. Scalability describes how much faster the code will run as the number of processors is increased. Figure 5 shows the scalability of the parallelized version of our code. The speedup factor denotes the time required to solve the problem on one processor vs. the time for the same problem on *N* processors. These tests were performed on the 3D Navier–Stokes simulation of drop rupture discussed in the previous section. For each test, 10 time-steps were run on a 128^3 uniform grid. At each time-step, the divergence-free condition was satisfied to $\nabla \cdot \mathbf{u}^n < 10^{-8}$. The timings were recorded by the Unix `/bin/time` command and are given in Table 1.

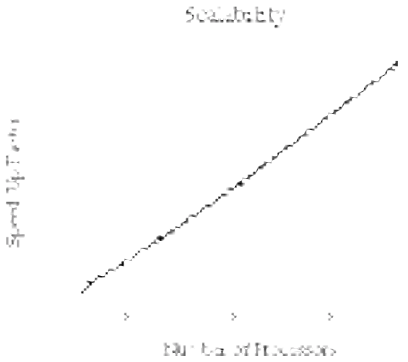

Figure 5. *Scalability curve for 3D Navier–Stokes simulation of drop rupture on up to 32 processors. The drop, with an initial radius of 0.25, is placed at the center of a computational box with a 128^3 mesh.*

Processors	CPU minutes	Speedup
1	22.0	1.0
2	16.0	1.4
4	9.2	2.4
8	5.0	4.4
16	2.5	8.8
32	1.2	18.3

Table 1. *CPU time for 10 time-steps in a 3D simulation of drop breakup in a 128^3-mesh cube.*

Efficiency in multiprocessing is defined as the ratio (time on *N* processors)/((*MN*) × (time on *M* processors)). In going from one to two processors, the efficiency for our code is about 69%, but improvements can be achieved with the use of additional processors. When the number of processors is increased from two to 32, for instance, the efficiency is 83%.

Summary

Direct simulation of two-fluid flows is often limited by the computational effort required and by the available memory, especially in the 3D case. For improved performance, we found three issues to be of the utmost importance: accuracy, stability, and efficiency. We parallelized our entire code, including the semi-implicit Stokes solver; on an Origin2000 with 32 parallel processors, the efficiency of the parallelized code is greater than 57%.

Acknowledgments

Support for the computations described here was provided by a start-up grant from the National Computational Science Alliance (NCSA) Origin2000, grant CTS 990010N, and the Interdisciplinary Center for Applied Mathematics at Virginia Tech.

References

[1] V. Cristini, *Drop dynamics in viscous flow*, PhD thesis, Yale University, 2000.

[2] J. Li, Y. Renardy, and M. Renardy, *Numerical simulation of breakup of a viscous drop in simple shear flow with a volume-of-fluid method*, Phys. Fluids, February 2000 (in press).

[3] C.R. Marks, *Drop breakup and deformation in sudden onset strong flows*, PhD thesis, University of Maryland at College Park, 1998.

[4] R. Scardovelli and S. Zaleski, *Direct numerical simulation of free surface and interfacial flow*, Ann. Rev. Fluid Mech., 31 (1999), 567–604.

[5] G.I. Taylor, *The formation of emulsions in definable fields of flow*, Proc. R. Soc. A, 146 (1934), 501–523.

Yuriko Renardy (renardyy@math.vt.edu) is a professor in the Department of Mathematics at Virginia Polytechnic Institute and State University. Jie Li (jl305@eng.cam.ac.uk) is a lecturer at the BP Institute, University of Cambridge.