JOURNAL OF COMPUTATIONAL PHYSICS 76, 349–384 (1988)

Surface Tension and Viscosity with Lagrangian Hydrodynamics on a Triangular Mesh

D. E. FYFE, E. S. ORAN, AND M. J. FRITTS*

Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory, Washington, D.C. 20375

Received March 20, 1986; revised January 22, 1987

Numerical algorithms for surface tension and viscosity are presented in the context of a Lagrangian treatment of incompressible hydrodynamics with a dynamically restructuring grid. New algorithms are given which update previous Lagrangian approaches in the code SPLISH. Test problems involving internal gravity and capillary waves, an oscillating droplet and a viscous shear layer are described. An example is given of a flow calculated in and around a viscous droplet with surface tension in a shear flow.

I. INTRODUCTION

In principle, a Lagrangian formulation of the hydrodynamics equations is particularly attractive for numerical calculations. Each discretized fluid element is tracked as it evolves through the interaction with its changing environment and with external forces. The local interactions can be represented without nonphysical numerical diffusion. Conservation laws are simple to express since there are no fluxes out of the fluid element boundaries. The paths of the fluid elements are themselves a flow visualization. It thus appears to be the natural approach to transient hydrodynamics with free surfaces, interfaces, or sharp boundaries.

In practice, the use of Lagrangian methods in numerical simulations has generally been restricted to "well-behaved" flows. Shear, fluid separation, or even larger amplitude motion produce severe grid distortion. These distortions arise because grid points can move far enough that their near-neighbors change in the course of a calculation. When differential operators are approximated over a mesh which is distorting, the approximations may become inaccurate. Attempting to regain accuracy through regridding and interpolating physical quantities onto the new grid introduces numerical diffusion into the calculation.

This paper is a summary and update of the latest additions and modifications to a numerical technique for indefinitely extending Lagrangian calculations by using a restructuring triangular mesh, first introduced by Fritts and Boris [1]. The major advance of this approach is that the grid automatically adapts and refines itself to

* Current address: SAIC, Annapolis, MD.

0021-9991/88 $3.00

maintain accuracy for discretized operators in a manner that is nondiffusive. The algorithms have been implemented in the code SPLISH, which has been applied to calculations of nonlinear waves [2, 3], flows over obstacles [4], Kelvin–Helmholtz instabilities [5], Rayleigh–Taylor instabilities [6], Couette flows, and Taylor vortex flows [7].

Work on Lagrangian techniques for grids which do not have fixed connectivity has recently had a renaissance. Early attempts included the PANACEA code [8] and the PAF (particle-and-force) algorithm [9, 10]. In the 1970s, these concepts were improved and extended for triangular grids: triangle reconnection by Crowley [11]; MHD algorithms over a triangular mesh [12]; and adaptive triangular meshes in the work mentioned in the previous paragraph on SPLISH. During the same period work began which used Voronoi meshes for hydrodynamics calculations [13].

Recently this use of general connectivity grids has rapidly expanded, as summarized in the First International Conference on Free-Lagrange Methods [14]. Applications now include finite-difference and finite-element calculations of classic hydrodynamic instabilities, tokamak modelling, high temperature plasma physics, heat conduction, wave-structure interactions, impact deformations and hydrodynamics problems for both compressible and incompressible fluids. Free-Lagrange methods now use quadrilateral, triangular, and mixed meshes in two dimensions, tetrahedral meshes in three dimensions, Voronoi meshes in both two and three dimensions, and methods which are mesh-free.

In this paper we present the latest modifications to SPLISH (Section II). These include the most recent version of the rotation operator, which conserves circulation, and the residual algorithm, which ensures conservation of the area of cells. We also introduce new algorithms for viscosity and surface tension. Including viscosity proved to be straightforward (Section II). However, the search for an algorithm good enough for surface tension (Section III) was more challenging and difficult. The basic problem is defining a proper curvature from a finite number of points. Because of this, the numerical approximation of surface tension forces between two fluids is conceptually quite different from approximations of convection and viscous forces. The final formulation chosen, a series of test problems, and a list of approaches that failed are detailed (Section III). Finally, we combine the convective transport, surface tension, and visocsity algorithms to perform some preliminary calculations of flows in and around a viscous kerosene droplet. These calculations show vortex shedding behind the droplet, distortion of the droplet due to the shear flow, and internal droplet flows.

II. Basic Elements of Lagrangian Triangular Grids

This section is a review of the derivation of low order finite-difference approximations to the equations describing incompressible fluid motion for general triangular grids. Some of the material was originally presented by Fritts and Boris

[1], and the interested reader is referred there for more detail. However, new material brings the previous paper up-to-date. This includes the lastest version of the rotation operator, which conserves circulation, the residual algorithm, which ensures conservation of the area of cells, and the new algorithm for viscosity.

A. *The Triangular Grid*

Consider a two-dimensional space which is divided into triangular cells. A section of this mesh shown in Fig. 1, which shows an interface between fluid type I and fluid type II. In Fig. 1a, a particular triangle j is highlighted by heavy lines and the various components of the triangle are labeled. Three vertices, V_1, V_2, and V_3, are connected consecutively by sides S_1, S_2, and S_3. The direction of labeling around each triangle is counterclockwise and the z axis is directed out of the page. Since the mesh can be irregularly connected, an arbitrary number of triangles can meet at each vertex.

We can define a cell surrounding a vertex, as shown in Fig. 1b, by the shaded region surrounding V_3. The borders of such vertex-centered cells are determined by constructing line sigments joining the centroid of each triangle with the midpoints of the two triangle sides connected to the vertex, for all triangles surrounding that vertex. This definition of a vertex cell equally apportions the area of a triangle to each of its three vertices and provides a simple, efficient way to evaluate the finite difference operators. However, the definition of a vertex cell is arbitrary. Other

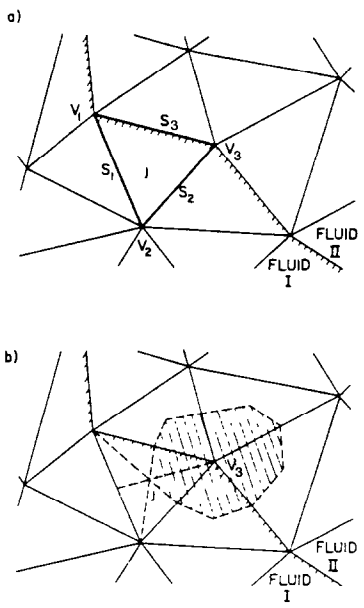

FIG. 1 A section of a triangular grid showing (a) a material interface, (b) a vertex cell.

definitions could be equally well employed, although they generally require additional calculations to determine cell intersection points. The integration of cell quantities may therefore involve more arithmetic operations for other definitions.

B. *Finite Differences on a Triangular Grid*

Finite-difference approximations for derivatives of functions defined on the triangular grid are derived from the expressions for the integral of the gradient of a scalar function, f, and the divergence and curl of a vector field, $\mathbf{v}$, in two Cartesion dimensions.

$$\int_A \nabla f \, dA = \oint_C f \, d\boldsymbol{l} \times \hat{z} \tag{2.1}$$

$$\int_A \nabla \cdot \mathbf{v} \, dA = \oint_C \mathbf{v} \cdot (d\boldsymbol{l} \times \hat{z}) \tag{2.2}$$

$$\int_A \nabla \times \mathbf{v} \, dA = \oint_C \mathbf{v} \cdot d\boldsymbol{l} \, \hat{z}. \tag{2.3}$$

In each of these expressions, A is the region enclosed by the curve C and $d\boldsymbol{l}$ is the vector arc length around C in the counterclockwise direction. The variable $\hat{z}$ is a unit vector in the direction of the ignorable coordinate. By using these definitions in a conservative integral approach, the definitions for spatial derivatives described below can be naturally extended to two-dimensional axisymmetric geometry [7].

Throughout the following discussion a triangle-centered quantity is assumed to be piecewise constant over the triangles with discontinuities occurring at the triangle sides and a vertex-centered quantity is assumed to be piecewise linear over the triangles. If we want to form a triangle-centered derivative, we use the triangles as the area A and the sides of the triangle for the curve C in Eqs. (2.1)–(2.3). We then approximate the area integral by the area of the triangle times the value of the derivative on the triangle, and approximate the line integral using the trapezoidal rule on each side of the triangle. For example, the gradient of a scalar function f defined at the vertices is a triangle-centered quantity, $(\nabla f)_j$, given by

$$A_j(\nabla f)_j = \tfrac{1}{2} \sum_{i(j)} f_i(\mathbf{r}_{i-1} - \mathbf{r}_{i+1}) \times \hat{z}, \tag{2.4}$$

where $\mathbf{r}_i = (x_i, y_i)$ is a vector coordinate for vertex i and A_j is the area of triangle j. We have also used the notation of Fritts and Boris [1] that $\sum_{i(j)}$ is interpreted as the sum over vertices i of triangle j. In the material presented below, the index i designates vertex-centered quantities and the index j designates triangle-centered quantities.

If we want to form a vertex-centered derivative, we use the vertex-centered cell as the area A. We approximate the area integral on the left side of Eqs. (2.1)–(2.3) by

the area of the vertex-centered cell times of value of the derivative at the vertex. We approximate the line integral using the value on each triangle and the appropriate vector length through the triangle. For example, the curl of the vector field $\mathbf{v}$ at a vertex c is approximated by

$$A_c(\nabla \times \mathbf{v})_c = \tfrac{1}{2} \sum_{i(c)} \mathbf{v}_{i+1\,2} \cdot (\mathbf{r}_{i+1} - \mathbf{r}_i)\hat{z}, \tag{2.5}$$

where $A_c = \frac{1}{3} \sum_{j(c)} A_j$ is the vertex-centered cell area, $\sum_{j(c)}$ is a sum over the triangles around the central vertex c, $\sum_{i(c)}$ is a sum over the vertices around vertex c, and $\mathbf{v}_{i+1\,2}$ is the value of the vector field $\mathbf{v}$ on the triangle having vertices $c, i, i+1$. Similarly, the divergence of the vector field $\mathbf{v}$ at a vertex is approximated by

$$A_c(\nabla \cdot \mathbf{v})_c = \tfrac{1}{2} \sum_{i(c)} [\mathbf{v}_{i+1\,2} \times (\mathbf{r}_{i+1} - \mathbf{r}_i)] \cdot \hat{z}. \tag{2.6}$$

C. *The Equations for Incompressible, Inviscid Flow*

The basic equations for inviscid incompressible hydrodynamics are

$$\frac{d\rho}{dt} = 0, \tag{2.7}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{2.8}$$

$$\rho \frac{d\mathbf{v}}{dt} + \nabla p = \mathbf{f}_e. \tag{2.9}$$

In two dimensions the fluid density ρ, pressure p, and velocity $\mathbf{v}$ are assumed to vary with x, y, and t. The term $\mathbf{f}_e$ represents external forces applied to the fluid, for example, forces due to gravity. Equation (2.8), the condition for incompressibility, removes the sound waves.

Since we want our finite difference approximation to preserve the conservation properties for incompressible inviscid fluids, it is important to consider which of the physical variables, ρ, $\mathbf{v}$, and p, should be defined as vertex-centered quantities and which should be defined as triangle-centered quantities. We have found that prescribing velocities as triangle-centered quantities makes the formulation of conservation of circulation straightforward. Prescribing the densities on triangles and pressures at vertices allows conservation of vertex cell areas.

The time integration of velocities uses a second-order implicit split-step algorithm which is solved by iteration. The vertex positions are advanced using a second-order midpoint rule. Specifically, the velocities are advanced a half timestep, the grid is advanced a full timestep, and then the velocities are advanced forward the other half timestep. The complete algorithm is as follows. First compute the half-timestep triangle velocities using

$$\mathbf{v}_j^{1\,2} = \mathbf{v}_j^o - \frac{\delta t}{2\rho_j} (\nabla p)_j^o + \frac{\delta t}{2\rho_j} \mathbf{f}_e, \tag{2.10}$$

where the superscript o designates the values at the old time step. We then make an initial guess for the new triangle velocities

$$v_j^{n,0} = v_j^{1/2}$$

and iterate

$$\mathbf{v}_i^{1/2,k} = \tfrac{1}{2}(\mathbf{v}_i^o + \mathbf{v}_i^{n,k-1}), \tag{2.11}$$

$$\mathbf{x}_i^{n,k} = \mathbf{x}_i^o + \delta t \mathbf{v}_i^{1/2,k}, \tag{2.12}$$

$$\tilde{\mathbf{v}}_j^{1/2,k} = \mathbf{R}(\{\mathbf{x}_i^o\}, \{\mathbf{x}_i^{n,k}\}) \cdot \mathbf{v}_j^{1/2}, \tag{2.13}$$

$$\mathbf{v}_j^{n,k} = \tilde{\mathbf{v}}_j^{1/2,k} - \frac{\delta t}{2\rho_j}(\nabla p)_j^{n,k} + \frac{\delta t}{2\rho_j}\mathbf{f}_e, \tag{2.14}$$

where the second superscript indicates the iteration number. The vertex velocity $\mathbf{v}_i^{n,k}$ in Eq. (2.11) is obtained from a weighted average of the triangle velocities $\mathbf{v}_j^{n,k}$ for those triangles having i as a vertex,

$$\mathbf{v}_i^n = \frac{\sum_{j(i)} w_j \mathbf{v}_j^n}{\sum_{j(i)} w_j}. \tag{2.15}$$

We use $w_j = \theta_j \rho_j A_j$, where θ_j is the angle (in radians) of triangle j at vertex i divided by π. The transformation $\mathbf{R}$ in Eq. (2.13) results from the requirement of conservation of circulation, and is discussed in Section D below.

The pressures $\{p_i^{n,k}\}$ in Eq. (2.14) are derived from the condition that the new velocities $\{\mathbf{v}_j^{n,k}\}$ should be divergence-free at the new timestep, satisfying Eq. (2.8). The pressure Poisson equation is derived from Eq. (2.14) by setting $(\nabla \cdot \mathbf{v}_j)^{n,k} = 0$ to obtain a pressure $p_i^{n,k}$, such that

$$\left(\nabla \cdot \frac{\delta t}{2\rho_j}(\nabla p)_j\right)_i^{n,k} = (\nabla^{n,k} \cdot \mathbf{v}_j^{1/2,k})_i + \left(\nabla^{n,k} \cdot \frac{\delta t}{2\rho_j}\mathbf{f}_e\right)_i. \tag{2.16}$$

Both terms in Eq. (2.16) are straightforward to evaluate, since the divergence is taken over triangle-centered quantities. Note also that the discrete gradient operator ∇ must also carry time advancement superscripts since it depends on the current grid location. (See Eq. (2.4).) Two features of the Poisson equation, Eq. (2.16), are noteworthy. First, it is derived from $\nabla^2\phi = \nabla \cdot \nabla\phi$, as in the continuum case. Second, the left-hand side results in the more familiar second-order accurate templates for the Laplacians (such as the five-point formula) derived for homogeneous fluids and regular mesh geometries.

D. *Conservation of Circulation*

The approach we have outlined is basically a control volume approach which uses an integral formulation to derive the difference algorithms. Equation (2.13),

which produces conservation of circulation over vertex cell volumes, is a consequence of this approach. It reflects numerically the fact that the triangle velocities must be altered as the grid rotates and stretches. This process does not prevent the addition or loss of vorticity due to external forces or changes in density at interfaces. Rather it corrects any numerical errors that may arise because the grid has moved. Thus it guarantees conservation of circulation at those vertices where the circulation theorem applies.

The transformation **R** is derived by considering the circulation about each vertex. Since triangle velocities are constant over the triangle, the circulation taken about the boundary of the vertex cell can be calculated from Eq. (2.5). The conservation of vorticity then takes the form of the operator **R** which preserves the value of the circulation about each vertex as the grid changes.

Conservation of circulation requires that at each timestep, and for each vertex, c,

$$\sum_{l(c)} \tilde{\mathbf{v}}_{l+1/2}^{k} \cdot (\mathbf{r}_{l+1}^{n,k} - \mathbf{r}_{l}^{n,k}) = \sum_{l(c)} \mathbf{v}_{l+1/2}^{o} \cdot (\mathbf{r}_{l+1}^{o} - \mathbf{r}_{l}^{o}). \tag{2.17}$$

For convenience in notation, we now drop the superscript $\frac{1}{2}$ for the velocities and the iteration superscript k appearing in Eq. (2.17). Since there are two components of velocity on each triangle, but only one constraint at each vertex, the form of the rotator is undetermined. Fritts and Boris [1] provided the additional constraints by making each term in the circulation integral associated with a given triangle a conserved quantity, and hence the sum in Eq. (2.17) remains unchanged. This means that for each triangle j,

$$\tilde{\mathbf{v}}_j \cdot (\mathbf{r}_{l+1}^{n} - \mathbf{r}_{l}^{n}) = \mathbf{v}_j \cdot (\mathbf{r}_{l+1}^{o} - \mathbf{r}_{l}^{o}), \qquad i = 1, 2, 3. \tag{2.18}$$

Although this approach conserves circulation, the following example shows that it is much too restrictive.

Consider an inviscid shear flow on the grid shown in Fig. 2a. Triangles above $y = 0$ have a velocity $v_x = -1$, and those below have a velocity $v_x = +1$. If after one step the vertices have moved as in Fig. 2b, conservation of circulation through Eq. (2.18) imparts a y-component to the velocities for those triangles bordering the shear. Although the circulation integral about each vertex in the grid is conserved, the flow is now no longer independent of y.

To obtain a better formulation of the transformation **R** we must consider Eq. (2.17) more carefully. Since Eq. (2.17) is linear in the unknowns $\{\mathbf{v}_l\}$, we can obtain the change in triangle velocities by considering the change produced by the movement of a single vertex c, with coordinates $\mathbf{r}_c$, and sum the resultant expression over all vertices. It is reasonable to assume that the rotator should change only the velocities of the triangles which have c as a vertex. As a result, conservation of circulation gives

$$\begin{aligned} &\tilde{\mathbf{v}}_{l+1/2} \cdot (\mathbf{r}_c^{n} - \mathbf{r}_{l+1}) + \tilde{\mathbf{v}}_{l-1/2} \cdot (\mathbf{r}_{l-1} - \mathbf{r}_c^{n}) \\ &\quad = \mathbf{v}_{l+1/2} \cdot (\mathbf{r}_c^{o} - \mathbf{r}_{l+1}) + \mathbf{v}_{l-1/2} \cdot (\mathbf{r}_{l-1} - \mathbf{r}_c^{o}) \end{aligned} \tag{2.19}$$

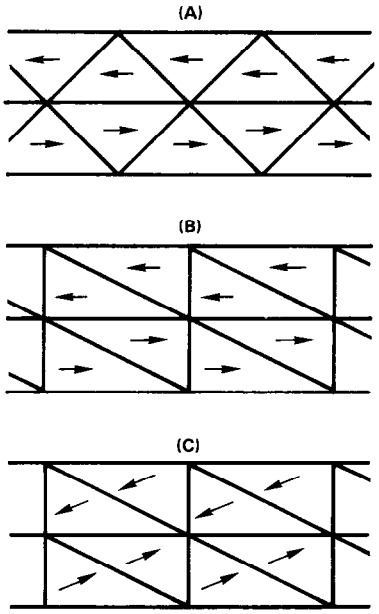

FIG. 2. A test problem for conservation of circulation: (a) the initial flow pattern; (b) the velocities after a half-time step, (c) the velocities after the old rotator operator is applied.

for each vertex i about c. We have used $\mathbf{r}_i = \mathbf{r}_i^n = \mathbf{r}_i^o$ for those vertices which are stationary. If only vertex c moves, the cell area at vertex c is constant, so that vorticity is conserved about vertex c as well. However, at all neighboring vertices, circulation, not vorticity, is conserved. By introducing the notation

$$\delta\mathbf{v}_{i+1/2} \equiv \tilde{\mathbf{v}}_{i+1/2} - \mathbf{v}_{i+1/2}$$

and

$$\delta\mathbf{r}_c \equiv \mathbf{r}_c^n - \mathbf{r}_c^o,$$

Eq. (2.19) may be rewritten as

$$\delta\mathbf{v}_{i+1/2} \cdot (\mathbf{r}_c^n - \mathbf{r}_{i+1}) + \delta\mathbf{v}_{i-1/2} \cdot (\mathbf{r}_{i-1} - \mathbf{r}_c^n) = (\mathbf{v}_{i-1/2} - \mathbf{v}_{i+1/2}) \cdot \delta\mathbf{r}_c. \tag{2.20}$$

Let us also decompose $\delta\mathbf{v}_{i+1/2}$ into a component, $t_{i+1/2}$, parallel to the side opposite vertex c, and a component, $n_{i+1/2}$, normal to the side opposite vertex c by writing

$$\delta\mathbf{v}_{i+1/2} \equiv n_{i+1/2} \frac{\hat{z} \times (\mathbf{r}_{i+1} - \mathbf{r}_i)}{|\mathbf{r}_{i+1} - \mathbf{r}_i|} + t_{i+1/2} \frac{\mathbf{r}_{i+1} - \mathbf{r}_i}{|\mathbf{r}_{i+1} - \mathbf{r}_i|}. \tag{2.21}$$

With this notation and using the equation for the area, $A_{i+1/2}$, of triangle $i+1/2$.

$$2A_{i+1/2} = \hat{z} \cdot [(\mathbf{r}_{i+1} - \mathbf{r}_i) \times (\mathbf{r}_c^n - \mathbf{r}_{i+1})], \tag{2.22}$$

Eq. (2.20) becomes

$$\frac{2A_{i+1/2}}{|\mathbf{r}_{i+1}-\mathbf{r}_i|}n_{i+1/2}+\frac{(\mathbf{r}_{i+1}-\mathbf{r}_i)\cdot(\mathbf{r}_c^n-\mathbf{r}_{i+1})}{|\mathbf{r}_{i+1}-\mathbf{r}_i|}t_{i+1/2}$$
$$+\frac{-2A_{i-1/2}}{|\mathbf{r}_i-\mathbf{r}_{i-1}|}n_{i-1/2}+\frac{(\mathbf{r}_i-\mathbf{r}_{i-1})\cdot(\mathbf{r}_{i-1}-\mathbf{r}_c^n)}{|\mathbf{r}_i-\mathbf{r}_{i-1}|}t_{i-1/2}$$
$$=(\mathbf{v}_{i-1/2}-\mathbf{v}_{i+1/2})\cdot\delta\mathbf{r}_c. \tag{2.23}$$

Let N_c denote the number of triangles (vertices) about an interior vertex c. The N_c equations given by Eq. (2.23) for the $2N_c$ unknowns $\{t_{i+1/2}\}$ and $\{n_{i+1/2}\}$ are linearly dependent. This can be seen by summing the equations, which produces the equation for the change in circulation about vertex c. The equation for the change in circulation at vertex c is a linear combination of the $t_{i+1/2}$'s, which is equal to zero. Since we want the $t_{i+1/2}$'s to be linearly independent, we can set $t_{i+1/2}=0$ for all i. We still need another equation to determine the normal component for the change in velocities on the triangles.

Let us for the moment write that equation as

$$\sum_{i=1}^{N_c} c_{i+1/2}n_{i+1/2}=b. \tag{2.24}$$

Using Eq. (2.23) with $t_{i+1/2}=0$ for all i, we can successively eliminate each $n_{i+1/2}$ for $i=1,\ldots,N_c-1$ in Eq. (2.24) until we arrive at an equation for $n_{N_c+1/2}$. Since the numbering of the triangles and vertices is arbitrary, this expression is valid for each triangle $i+1/2$ by replacing $n_{N_c+1/2}$ with $n_{i+1/2}$ and $\mathbf{v}_{N_c+1/2}$ with $\mathbf{v}_{i+1/2}$. The result is that

$$\delta\mathbf{v}_{i+1/2}=\frac{\hat{z}\times(\mathbf{r}_{i+1}-\mathbf{r}_i)}{2A_{i+1/2}}$$
$$\times\left[b-\sum_{k(c)}\frac{c_{k+1/2}\,|\mathbf{r}_{k+1}-\mathbf{r}_k|}{2A_{k+1/2}}(\mathbf{v}_{k+1/2}-\mathbf{v}_{i+1/2})\cdot\delta\mathbf{r}_c\right]\Bigg/\sum_{k(c)}\frac{c_{k+1/2}\,|\mathbf{r}_{k+1}-\mathbf{r}_k|}{2A_{k+1/2}}. \tag{2.25}$$

Several alternatives are possible for Eq. (2.24). If we conserve divergence about the vertex c, then

$$c_{i+1/2}=|\mathbf{r}_{i+1}-\mathbf{r}_i|,$$
$$b=0. \tag{2.26}$$

The transformation **R** prescribed by Eq. (2.25) is time-reversible, hence Eqs. (2.10)–(2.14) are also reversible. The entire algorithm advances vertex positions and velocities reversibly while evolving the correct circulation about every

interior vertex. This technique is unique for Lagrangian codes, which usually either ignore conservation of circulation completely or conserve circulation through an iteration performed simultaneously with the pressure iteration. With this method the circulation is conserved exactly regardless of whether the pressures have iterated to their final values.

E. *Viscous Flows*

Viscosity modifies Eq. (2.9), so that now

$$\rho \frac{d\mathbf{v}}{dt} + \nabla p = \mathbf{f}_e + \mu \nabla^2 \mathbf{v}. \tag{2.27}$$

Discretization of the additional term in the momentum equation follows the same approach as the discretization of the other terms. Since the velocity is a triangle-centered quantity, we need a discrete vertex-centered gradient operator, and a discrete triangle-centered divergence operator. Employing the same techniques as above we have

$$A_i(\nabla f)_i = \tfrac{1}{2} \sum_{t(i)} f_{t+1/2}(\mathbf{r}_{t+1} - \mathbf{r}_t) \times \hat{z}, \tag{2.28}$$

and

$$A_j(\nabla \cdot \mathbf{v})_j = \tfrac{1}{2} \sum_{t(j)} [\mathbf{v}_t \times (\mathbf{r}_{t+1} - \mathbf{r}_{t-1})] \cdot \hat{z}. \tag{2.29}$$

The Laplacian is found by taking the divergence of the gradient.

The finite difference equations, Eqs. (2.10) and (2.14), can be modified to account for the additional term in the momentum equation by

$$\mathbf{v}_j^{1/2} = \mathbf{v}_j^o - \frac{\delta t}{2\rho_j}(\nabla p)_j^o + \frac{\delta t}{2\rho_j}\mathbf{f}_e + \frac{\mu_j \delta t}{2\rho_j}(\nabla^2 \mathbf{v})_j^o, \tag{2.30}$$

$$\mathbf{v}_j^{n,k} = \tilde{\mathbf{v}}_j^{1/2,k} - \frac{\delta t}{2\rho_j}(\nabla p)_j^{n,k} + \frac{\delta t}{2\rho_j}\mathbf{f}_e + \frac{\mu_j \delta t}{2\rho_j}(\nabla^{n,k} \cdot \nabla^{n,k} \mathbf{v}^{n,k-1})_j. \tag{2.31}$$

These equations are implicit in the velocities, just as the original Eqs. (2.10)–(2.14) are. As in the inviscid case, we solve by iteration.

This algorithm was tested by calculating the spreading of a shear layer of initially zero thickness given by

$$\mathbf{v}(x, y, t=0) = \begin{cases} (v_x, 0), & \text{for } y > y_0, \\ (0, 0), & \text{for } y = y_0, \\ (-v_x, 0), & \text{for } y < y_0, \end{cases} \tag{2.32}$$

where y_0 is the original location of the vortex sheet. The velocity distribution across this layer evolves as

$$\mathbf{v}(x, y, t) = \hat{x}\, v_x \operatorname{erf}\left[\frac{(y - y_0)}{(4\nu t)^{1/2}}\right], \tag{2.33}$$

where $\nu = \mu/\rho$. The width Δy of the layer grows as

$$\Delta y \approx 8(\nu t)^{1/2}. \tag{2.34}$$

For the test calculation the grid was initialized to center a vortex sheet in a grid 16 cells wide with an initial layer width of zero. The two opposing streams had initially constant velocity profiles. The evolution of the interface between the streams was governed by the same algorithms as the interior of either fluid, so that no special interface boundary condition was used. The boundary conditions on the sides of the computational region were periodic, and the top and bottom had free-slip boundary conditions.

At the end of the calculation, the layer width agreed to within numerical round-off with the theory and the layer extended over the whole mesh. The velocity profile for each stream coincided with that given by Eq. (2.33) to within round-off error. The y-components of the velocity remained zero, indicating that the algorithm was working well for the grid distortions presented by the problem.

F. *Conservation of Vertex Cell Areas*

Equations (2.10)–(2.14) are implicit in the triangle velocities $\{\mathbf{v}_j\}$. Because these equations must be solved iteratively to produce a divergence free velocity field, a small residual error may remain. In addition, vertex velocities are derived from the divergence-free triangle velocities. In practice this means that vertex cell areas may not be conserved. Furthermore, as the flow progresses, the triangle sides distort. Yet at any given time we compute using straight triangle sides, which does not produce the equivalent cell area about any given vertex. However, since we know what the triangle area should be, it is possible to at least make a correction to the known error. Our approach, then, is to perform an ad hoc correction step after all the vertices have been advanced in time. This correction step moves the vertices in order to conserve vertex cell area. After this vertex correction step, the rotator is applied to ensure that the circulation has not been changed.

To expand or contract a vertex cell area, we must expand or contract the surrounding triangles areas. Suppose we wish to expand a triangle j with area A_j and vertex coordinates $\mathbf{r}_i$ by an amount δA_j. To do this we will move each vertex $\mathbf{r}_i$ an amount

$$\mathbf{r}_i^{\text{new}} - \mathbf{r}_i = \delta\mathbf{r}_i = d_j[\hat{z} \times (\mathbf{r}_{i-1} + \mathbf{r}_{i+1})]; \tag{2.35}$$

that is, the vertices of the triangle are moved normally to the opposite side by a

distance prescribed by the triangle expansion factor, d_j. If d_j is positive, the triangle area increases. Using the vector definition for the area of a triangle, we have

$$\begin{aligned} 2\delta A_j &= 2A_j^{\text{new}} - 2A_j \\ &= [(\mathbf{r}_{i+1}^{\text{new}} - \mathbf{r}_i^{\text{new}}) \times (\mathbf{r}_{i-1}^{\text{new}} - \mathbf{r}_{i+1}^{\text{new}})] \cdot \hat{z} - [(\mathbf{r}_{i+1} - \mathbf{r}_i) \times (\mathbf{r}_i - \mathbf{r}_{i-1})] \cdot \hat{z} \\ &= [(\mathbf{r}_{i+1} - \mathbf{r}_i) \times (\delta\mathbf{r}_{i-1} - \delta\mathbf{r}_{i+1})] \cdot \hat{z} \\ &\quad + [(\delta\mathbf{r}_{i+1} - \delta\mathbf{r}_i) \times (\mathbf{r}_{i-1} - \mathbf{r}_{i+1})] \cdot \hat{z} \\ &\quad + [(\delta\mathbf{r}_{i+1} - \delta\mathbf{r}_i) \times (\delta\mathbf{r}_{i-1} - \delta\mathbf{r}_{i+1})] \cdot \hat{z} \\ &= s^2 d_j + 6A_j d_j^2, \end{aligned} \tag{2.36}$$

where s^2 is the sum of the squares of the sides of the triangle. This quadratic in the expansion factor, d_j, can be solved to yield

$$d_j = \frac{-s^2 + \sqrt{s^4 + 48A_j \delta A_j}}{12A_j}. \tag{2.37}$$

The sign in front of the square root was chosen to ensure d_j has the same sign as δA_j.

We relate the change in triangle area, δA_j, to the conservation of vertex cell areas through

$$\delta A_j = \frac{A_j}{3} \sum_{i(j)} \frac{A_i' - A_i}{A_i} = \sum_{i(j)} \frac{A_j/3}{A_i} \delta A_i, \tag{2.38}$$

where the sum is over the three vertices of the triangle, A_i is the current area about vertex i and A_i' is the original area about vertex i. Basically, the change in vertex cell areas is apportioned to each contributing triangle according to that triangle's contribution to the vertex cell area.

Although this residual correction is a small numerical effect, we have found that it improves the overall results of a calculation. Because this algorithm expands triangles, it has potential for modelling other physical processes. In a compressible algorithm involving energy release and fluid flows with transit times which are small compared to the energy release times, this algorithm could be used to produce the required expansion of the vertex cells.

G. *Grid Restructuring*

In Lagrangian calculations the grid may distort to the point where grid restructuring is necessary. The derivations of the reconnection and vertex addition and deletion algorithms are done through the control volume approach and the use of triangle velocities. For all the algorithms used, the area-weighted divergence and curl taken about each vertex are both identically conserved for grid reconnections and vertex addition.

The accuracy of a general triangular mesh is diminished by large obtuse angles within triangles. With reconnections, accuracy can be recovered by ensuring that large obtuse angles are preferentially eliminated. There are many ways of formulating a reconnection algorithm. The one we have chosen is based on requirements for solving the pressure Poisson equation. The pressure Poisson equation is formally equivalent to that obtained by a piece-wise linear Rayleigh–Ritz–Galerkin finite element procedure on a triangular grid. (See, for example, [15].) Since we solve the equation by iteration, we want the iteration to converge as rapidly as possible. Mathematically, convergence is assured if the finite difference equation has a maximum principle; that is, all the off-diagonal terms are negative, the diagonal term is positive and greater than or equal to the absolute value of the sum of the off-diagonal terms, with strict inequality for at least one equation. (That one equation typically involves boundary conditions. Our boundary condition prescribes the integrated pressure along the upper boundary.)

To see how large angles affect the maximum principle, consider the difference equation for vertex l of Fig. 3a. The off-diagonal coefficient relating vertex l to vertex j is

$$a = -\tfrac{1}{2}(\cot\theta^- + \cot\theta^+), \tag{2.39}$$

where θ^+ and θ^- are the angles opposite the line from the vertex j to the vertex l as shown in Fig. 3a. The other off-diagonal terms are determined in a similar manner from the remaining edges emanating from vertex l. The diagonal coefficient is the negative of the sum of the off-diagonal terms. For positive area triangles, θ^+ and θ^- are both between 0° and 180°. Hence, each term in Eq. (2.39) is negative only when $\theta^+ + \theta^- > 180°$, since

$$a = \frac{\sin(\theta^+ + \theta^-)}{2\sin\theta^+ \sin\theta^-}. \tag{2.40}$$

The reconnection algorithm ensures that the angles subtended by any given edge sum to no more than 180°. If $\theta^+ + \theta^-$ is greater than 180°, the grid line is reconnected as shown in Fig. 3b. The new angles, θ'^+ and θ'^-, must sum to less than 180° since $(\theta^+ + \theta^- + \theta'^+ + \theta'^-)$ is the sum of the interior quadrilateral angles,

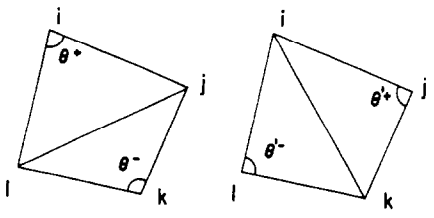

FIG 3. (a) Definition of the angles θ^+ and θ^- for the diagonal line drawn from j to l. (b) The angles θ'^+ and θ'^- formed by connecting the other quadriliateral diagonal.

which must be 360°. By chosing the diagonal which divides the largest opposing angles, the reconnection algorithm preferentially eliminates large angles in triangles.

Interface sides are never allowed to reconnect. In such cases vertex addition algorithms are needed. Vertex addition algorithms are also needed where the flow naturally depletes vertices. For vertex addition, satisfaction of conservation integrals is particularly simple. The vertex added at the centroid of a triangle subdivides that triangle into three smaller triangles. A vertex added to the midpoint of a side subdivides the two adjacent triangles into four smaller triangles. If the new triangle velocities are all the same as the velocity of the subdivided triangles, all conservation laws are satisfied. Since the reconnection algorithm is also conservative, subsequent reconnections to other vertices ensure that the only effect of the addition is an increase in resolution.

The case is not as obvious for vertex deletion. Reconnections can be used to surround any interior vertex whithin a triangle. The vertex is then removed and the new larger triangle given a velocity which is the area-weighted sum of the old velocities,

$$A_l \mathbf{v}_l = A_i \mathbf{v}_i + A_j \mathbf{v}_j + A_k \mathbf{v}_k. \tag{2.41}$$

Such a substitution redistributes circulation in accordance with area coordinates. Figure 4 illustrates the triangles before and after vertex removal. If ζ_4 is the vorticity about vertex 4 before it is removed, then the vorticity about each of the other three vertices is increased by an amount ζ_l' given by

$$\begin{aligned} \zeta_1' &= A_i \zeta_4 / A_l, \\ \zeta_2' &= A_k \zeta_4 / A_l, \\ \zeta_3' &= A_i \zeta_4 / A_l, \end{aligned} \tag{2.42}$$

where

$$\zeta_1' + \zeta_2' + \zeta_3' = \zeta_4,$$

since

$$A_i + A_j + A_k = A_l.$$

Therefore, total vorticity is conserved and redistributed in a reasonable and natural manner.

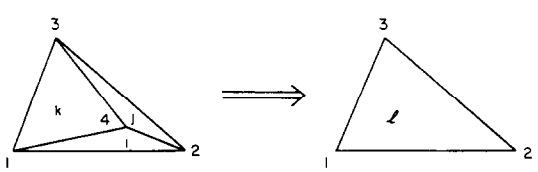

FIG. 4. (a) Vertex 4 isolated within a larger triangle before its removal. (b) The larger triangle remaining after deletion of vertex 4 and three associated sides and triangles.

III. Surface Tension

A. *The Algorithm*

The surface tension at an interface between two materials depends on the curvature of the interface. In the conventional numerical representation of surface tension, it is cast into a finite-difference form by fitting vertices on the material interface to some parametric function. This function is then used to find an estimate of local curvature. Once the curvature is known, a surface tension force is evaluated and used to accelerate interface vertices.

This scheme fails in SPLISH for two reasons. First, the interface vertices are accelerated directly by surface tension forces evaluated on the vertices. Since velocities are centered on triangles in SPLISH, the velocity field sees the effect of the acceleration a half timestep later, unless a secondary calculation is made. As a result, the pressure calculated within the droplet is inconsistent with that found from the surface tension formula. Second, since the pressure gradient forces and surface tension forces are not calculated in the same manner, numerical errors result which grow with each timestep.

Both of these problems are eliminated by a different formulation of surface tension, in which a surface tension potential is used to generate the forces. The surface tension force is formulated as a gradient of a potential present only at the surfaces. With this method, the pressure gradient forces are calculated in the same manner and on the same grid as the forces derived from the surface tension potential. Therefore both the surface tension potential and the pressure are dynamically similar, and the physical pressure drop across the interface must exactly cancel the surface tension forces. Preliminary aspects of this work were described by Fritts *et al.* [16, 17].

The finite-difference algorithms for surface tension are straightforward. The surface tension forces are included through Laplace's formula for the pressure jump across an interface [18],

$$p_i - p_o = \sigma/R, \tag{3.1}$$

where p_i is the pressure just inside the droplet at the interface, p_o is the pressure just outside the droplet at the interface, σ is the surface tension coefficient associated with the two media which define the interface, and R is the radius of curvature in the two-dimensional plane. The radius of curvature is positive at points on the interface where the droplet surface is convex (a circle is convex everywhere) and negative when the surface is concave. These pressure jumps are included in the Poisson equation for the pressure. The average pressure, $(p_i + p_o)/2$, is computed at the interface vertices. From the average pressure and the pressure jump, we can compute a pressure gradient centered on triangles, both inside and outside the surface. This pressure gradient is used in the momentum equation.

The radius of curvature is computed from a parametric cubic spline interpolant to the interface vertices. Past calculations of droplets oscillating due to surface tension forces [19, 20] also use cubic spline interpolation. However, they divided the surface into at least four segments (the top, bottom, right, and left sides of the droplet) to produce an interpolant on each segment. Each interpolant was matched at the joints to produce an overall curve. The parametric interpolant used here does not require this special matching.

The parametric spline is produced in the following manner. Denote the interface vertices by $\mathbf{r}_i = (x_i, y_i)$, $i = 1, ..., N$, with $\mathbf{r}_N = \mathbf{r}_1$. Also define a pseudo arc length parameter, s, such that the spline knots occur at the points

$$\begin{aligned} s_1 &= 0, \\ s_i &= s_{i-1} + |\mathbf{r}_i - \mathbf{r}_{i-1}|, \qquad i = 2, ..., N. \end{aligned} \tag{3.2}$$

We generate the twice differentiable periodic spline interpolants, $\mathbf{r}(s) = (x(s), y(s))$ from the data $\{s_i\}$, and $\{\mathbf{r}_i\}$, $i = 1, ..., N$, as prescribed by DeBoor [21]. The curvature is then given by

$$K \equiv R^{-1} = \frac{|\mathbf{r}'' \times \mathbf{r}'|}{|\mathbf{r}'|^3}, \tag{3.3}$$

where the prime indicates differentiation with respect to the parameter s. The sign of R at an interface vertex, $\mathbf{r}_i$, is given by the sign of $\hat{z} \cdot [(\mathbf{r}_{i+1} - \mathbf{r}_i) \times (\mathbf{r}_{i-1} - \mathbf{r}_i)]$.

We can iterate the process if necessary. From the spline fit we can generate new values for the $\{s_i\}$ by integrating the expression for arc length along a parametrically prescribed curve. For symmetrically placed vertices on a symmetric droplet, however, we have found that the iteration on arc length parameter is unnecessary.

The parametric spline fit is also used for regridding. When the regridding algorithm calls for the bisection of a triangle side which borders the two media, a new vertex is added on the spline interpolant between the vertices. This is done rather than bisecting the straightline segment, since a straightline bisection introduces spurious interface oscillations. Bisecting the spline maintains a better overall shape for the interface.

B. *Test Results*

We tested the algorithm for surface tension in SPLISH using two test problems. The first test problem consists of internal capillary waves. In the second test problem we calculated the oscillation of a droplet due to surface tension. For completeness we also present calculations of internal gravity waves as a test of the overall hydrodynamic algortithms in SPLISH.

1. *Internal Gravity and Capillary Waves*

The linear theory for the small amplitude oscillation of an interface between two fluids, bounded above and below by solid walls, gives the frequency ω as a function of wavenumber k,

$$\omega^2 = \frac{(\rho - \rho')\, gk + \sigma k^3}{\rho \coth kh + \rho' \coth kh'}.$$

Here the upper fluid is of depth h' and density ρ', the lower fluid is of depth h and density ρ, g is the acceleration due to gravity and σ is the coefficient of surface tension for the two media. Following the free-surface wave calculations of Fritts and Boris [1], we take $k = 2\pi/\lambda$, $\lambda = 2.5$ cm, $h = h' = 1.0$ cm, $\rho = 2$g/cc, and $\rho' = 1$g/cc. For an internal gravity wave, we have $g = 980$ cm^2/s and $\sigma = 0$ dynes/cm. For an internal capillary wave, we have $g = 0$ cm^2/s and $\sigma = 30$ dynes/cm. These values give a period $\tau = 2\pi/\omega = 0.22073$ s for the internal gravity wave and $\tau = 0.50196$ s for the internal capillary wave. The amplitude of the oscillation is taken as $A = 0.0672h$. For this amplitude the free-surface oscillations of Fritts and Boris [1] showed negligible non-linear effects. Figure 5 shows the initial grid for the mesh size $\delta s = 0.125$ cm.

Figure 6 shows the wave period as a function of mesh size for the internal gravity wave problem. The ratio of timesteps for any two calculations was the same as that for the mesh sizes. Each data point on the curve is an average over several periods and is accurate to three digits. If we extrapolate to zero mesh size using a parabolic least-squares fit, $\tau = \tau_0 + b\delta s + a(\delta s)^2$ to the data points, we obtain $\tau_0 = 0.2214$, $b = 0.0726$, and $a = 0.1549$ for this problem. The extrapolated value, τ_0, is accurate to 0.3%. The finite-difference derivatives given in Section II are accurate to second order in the mesh size for triangular grids in which the centroid of a vertex cell is the vertex itself. The truncation error is linear in the distance between th vertex and the centroid of the vertex cell. This truncation error can occur in this problem for vertex cells near the interface in our discretization and hence the linear term in δs in the above quadratic expression. This linear term has a coefficient on the order of the wave amplitude which is the approximate distortion of the grid. The order of convergence for the algorithm is essentially quadratic with a small linear contribution.

Figure 7 shows the wave period as a function of mesh size for the internal capillary wave problem. Here the least-squares fit to the data gives $\tau_0 = 0.4995$, $b = 0.2198$, and $a = 0.0640$. The extrapolated period is accurate to 0.5%. With surface tension included, the convergence is primarily linear in the mesh size. The reduction in rate of convergence is due to the use of cubic splines to calculate the curvatures. The cubic spline curve itself is fourth-order accurate, and theorems exist showing the second-order accuracy of its second derivatives. However, we know of no theorem giving the accuracy of the combination of derivatives needed to produce the curvature in Eq. (3.3).

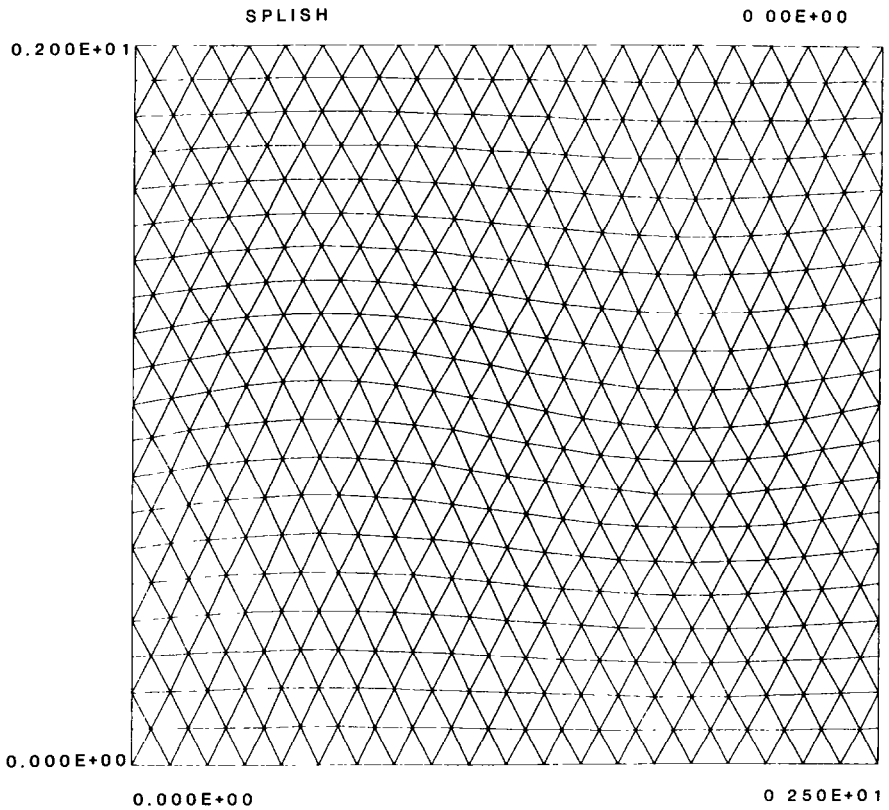

FIG. 5. The initial grid for the internal wave test problems.

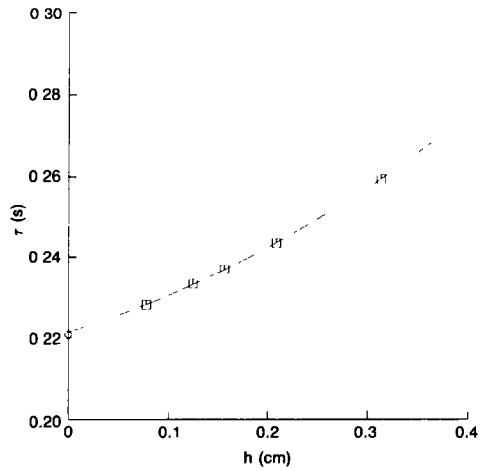

FIG. 6. The period τ as a function of mesh size for the internal gravity wave test problem.

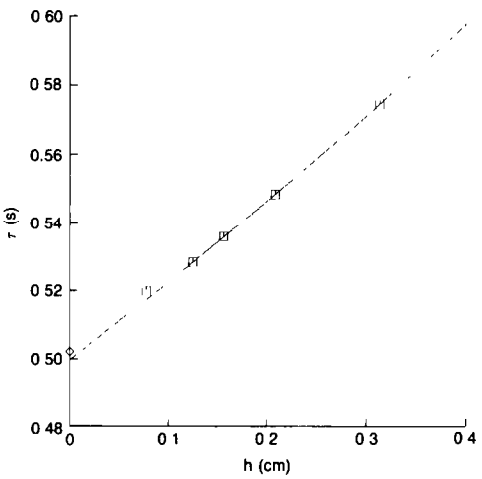

FIG 7. The period τ as a function of mesh size for the internal capillary wave test problem.

2. *Droplet Oscillation*

As a further test of the algorithm for surface tension in SPLISH, we calculated the oscillation of a droplet due to surface tension. Rayleigh [22] derived a linear theory for small amplitude oscillations on cylindrical jets that applies to the cylindrical droplets we are discussing. He concluded that when the perturbation is totally in the plane perpendicular to the axis of the cylinder, the frequency, ω, for the oscillation is given by

$$\omega_n^2 = (n^3 - n)\frac{\sigma}{\rho a^3}, \tag{3.4}$$

where the surface of the droplet is given in polar coordinates by

$$r = a + \varepsilon \cos(n\theta), \tag{3.5}$$

where ρ is the density of the jet, a is the unperturbed radius of the jet, and n prescribes the mode of oscillation in the plane with amplitude ε. For large amplitude oscillations, Rayleigh found that the experimental frequency diverged from that predicated by the linear theory, and he attributed these difference to nonlinear effects.

We have extended Rayleigh's theory to include the presence of an external fluid. Equation (3.4) then becomes

$$\omega_n^2 = (n^3 - n)\frac{\sigma}{(\rho_d + \rho_e)a^3}, \tag{3.6}$$

where ρ_d is the droplet density and ρ_e is the density of the external fluid.

The tests of the surface tension algorithm consisted of a series of calculations of oscillations initiated in the lowest oscillating mode, $n = 2$ in Eq. (3.6). Also, we have chosen

$$a = 0.0125 \text{ cm}$$

$$\sigma = 30 \text{ dynes/cm},$$

values which are typical for many practical droplet problems. We discuss results for two different sets of conditions. First we consider a droplet density of 2 g/cc in a background external fluid density of 1 g/cc. If we use the definition of the period as $2\pi/\omega$, Eq. (3.6) gives a period

$$\tau = 1.13 \times 10^{-3} \text{s}.$$

The second set of conditions are for a kerosene droplet, with density 0.82 g/cc, in a background of air, with density 0.0013 g/cc. This second case, with the 650:1 density ratio, is a stringent test of the numerical approximations.

Figure 8 is a composite of frames from a calculation in which $\varepsilon = 0.2a =$ 0.0025 cm for the 2:1 density ratio case. In this calculation there are 17 vertices in each direction along the exterior boundaries, 12 vertices on the droplet interface, and a total of 313 vertices initially in the calculation. The computational domain is 0.1 cm on a side. The left and right boundaries are periodic while the top and bottom boundaries are solid walls. The timestep is $\delta t = 2.5 \times 10^{-5}$ s. The figures show four and a half oscillations of the droplet. We can see that as the calculation

This was the case because the initial gridding was adequate to represent the droplet shape. From these calculations, the period of oscillation is

$$\tau_{12} = 1.35 \times 10^{-3} \text{ s}.$$

Similar calculations with 20 vertices surrounding the droplet (a 21×21 grid) show a period of

$$\tau_{20} = 1.33 \times 10^{-3} \text{ s},$$

for 24 vertices surrounding the droplet (a 25×25 grid) we have a period of

$$\tau_{24} = 1.31 \times 10^{-3} \text{ s},$$

and for 28 vertices surrounding the droplet (a 33×33 grid) the period is

$$\tau_{28} = 1.27 \times 10^{-3} \text{ s}.$$

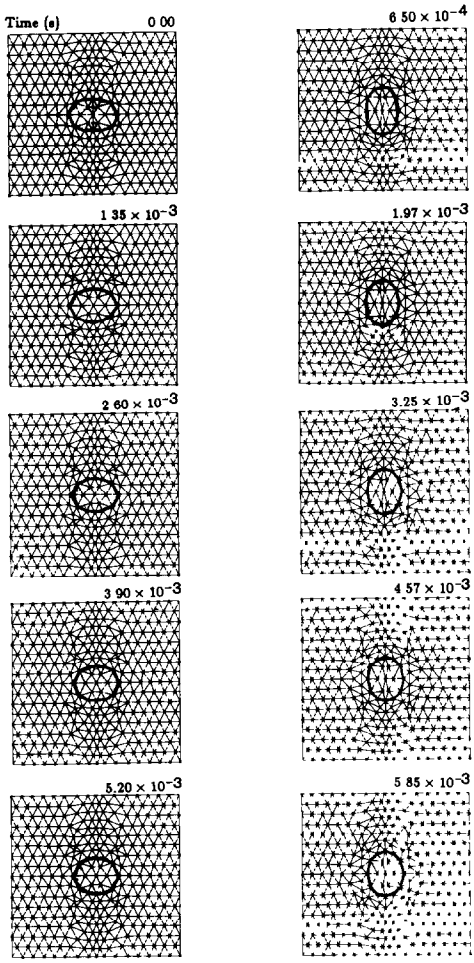

FIG. 8. A composite of frames from a calculation of an $n = 2$ normal mode droplet oscillation with 12 vertices around the droplet· $\rho_c = 1$ g/cc, $\rho_d = 2$ g/cc, $\sigma = 30$ dynes/cm, $a = 0.0125$ cm. Each frame is $0\,1 \times 0.1$ cm^2.

In each case, the period does not change during the calculation. Figures 9 and 10 show the initial oscillation for the more resolved cases. For these calculations, it was necessary to decrease the timestep, as discussed below. The time step for the calculation with 12 vertices surrounding the droplet is such that the period cannot be resolved to better than two digits. It appears that the calculations are not converging to the theoretical value, but to a value of 1.19 ± 0.06s, based on the graph of the computed period as a function of mesh size shown in Fig. 11. The convergence is essentially linear as it was in the internal capillary wave test problem, but with a numerical error of about 5.5% for this calculation.

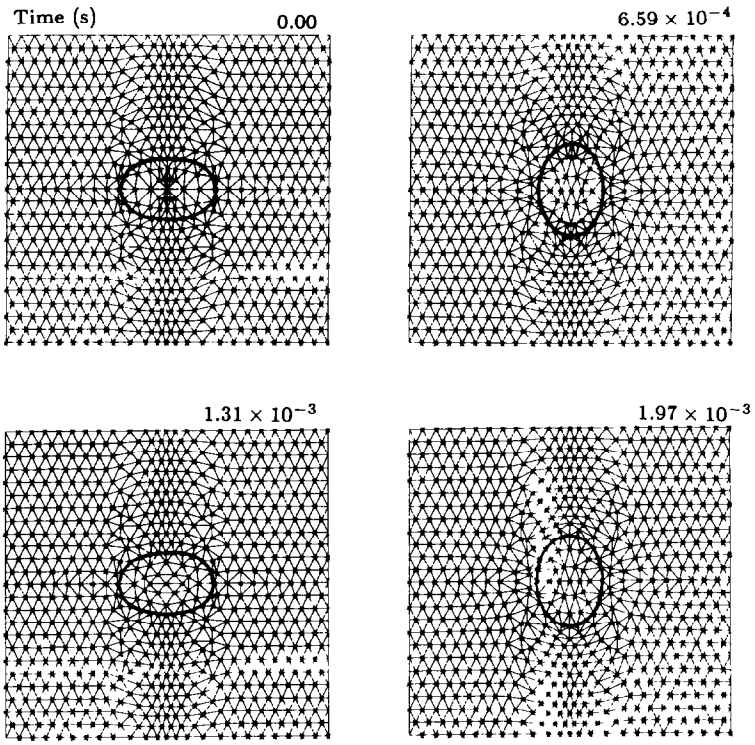

FIG. 9. A composite of frames from a calculation of an $n = 2$ normal mode droplet oscillation with 24 vertices around the droplet. Same conditions as in Fig. 7.

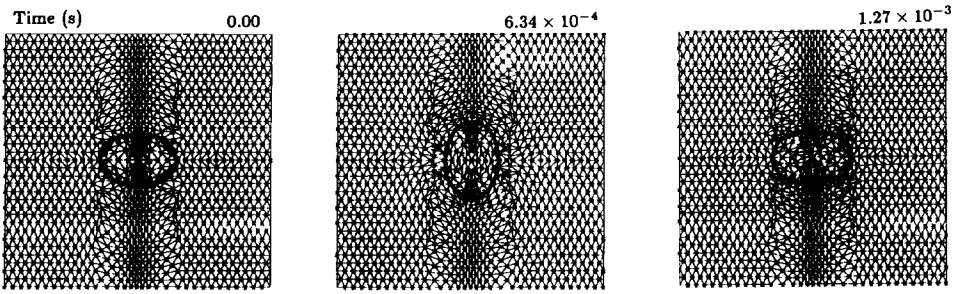

FIG. 10. A composite of frames from a calculation of an $n = 2$ normal mode droplet oscillation with 28 vertices around the droplet. Same conditions as in Fig. 7.

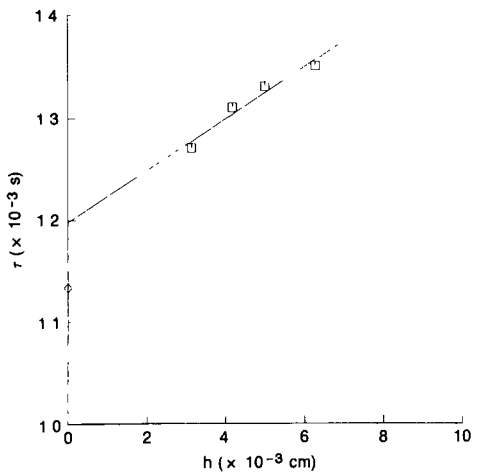

FIG. 11 The period as a function of mesh size for the droplet oscillation problem

Since the internal wave tests show much better convergence properties for the algorithm, as do previous free-surface wave calculations [1, 2], than the droplet oscillation test problem, we performed several other numerical tests on the droplet oscillation problem to determine if the poorer convergence properties were due to other numerical parameters.

First, we tested whether the presence of boundaries a finite distance away could alter the calculated period by performing calculations in a larger domain of length 0.2 cm. Here there were twice as many vertices on the boundary, but still only 12 vertices surrounding the droplet which was the same size as the droplets in the tests described above. These calculations showed no change in period, so we conclude that the effects of periodic boundaries and reflecting walls are negligible.

It was also important to evaluate the possible effects of nonlinearity in the solution. The theoretical value is from a linear analysis, and the calculation is a full nonlinear calculation. It is possible that this could account for part of the discrepancy. To test this, we performed calculations with smaller amplitudes, ε, to see if there was any difference in calculated period. The result was that the numerical value of the period was the same for $\varepsilon = 0.01a = 0.000125$ cm over the course of two oscillations as it was for $\varepsilon = 0.2a$. Our conclusion is that the calculations were in a range in which the linear theory is valid.

We used two diagnostics to determine the period of the computed droplet oscillation. One is the time history of the position of the rightmost vertex on the droplet interface, denoted by x_r. The other diagnostic is the quadratic moment, defined by

$$\langle x^2 \rangle = \int x^2 \, dx \, dy, \tag{3.7}$$

TABLE I

16 × 16 Grid

Time	Last period	$\langle x^2 \rangle$
0.0000		$0.3061E-07$
$0.6500E-03$		$0.1426E-07$
$0.1300E-02$	0.0013	$0.2929E-07$
$0.1975E-02$		$0.1497E-07$
$0.2600E-02$	0.0013	$0.2821E-07$
$0.3250E-02$		$0.1563E-07$
$0.3900E-02$	0.0013	$0.2711E-07$
$0.4550E-02$		$0.1626E-07$
$0.5200E-02$	0.0013	$0.2627E-07$
$0.5850E-02$		$0.1677E-07$
$0.6500E-02$	0.0013	$0.2556E-07$
$0.7150E-02$		$0.1718E-07$
$0.7775E-02$	0.001275	$0.2493E-07$
$0.8425E-02$		$0.1757E-07$
$0.9075E-02$	0.0013	$0.2435E-07$
$0.9725E-02$		$0.1793E-07$
$0.1037E-01$	0.0013	$0.2387E-07$
$0.1100E-01$		$0.1827E-07$
$0.1165E-01$	0.001275	$0.2349E-07$

TABLE II

16 × 16 Grid

Time	Last period	x_r
0.0000		$0.1500E-01$
$0.6250E-03$		$0.9974E-02$
$0.1325E-02$	0.001325	$0.1486E-01$
$0.1975E-02$		$0.1046E-01$
$0.2600E-02$	0.001275	$0.1463E-01$
$0.3275E-02$		$0.1064E-01$
$0.3900E-02$	0.0013	$0.1423E-01$
$0.4550E-02$		$0.1078E-01$
$0.5225E-02$	0.001325	$0.1400E-01$
$0.5825E-02$		$0.1104E-01$
$0.6525E-02$	0.0013	$0.1392E-01$
$0.7150E-02$		$0.1131E-01$
$0.7775E-02$	0.00125	$0.1378E-01$
$0.8450E-02$		$0.1140E-01$
$0.9050E-02$	0.001275	$0.1355E-01$
$0.9725E-02$		$0.1146E-01$
$0.1037E-01$	0.001325	$0.1335E-01$
$0.1100E-01$		$0.1159E-01$
$0.1167E-01$	0.0013	$0.1332E-01$

TABLE III

24 × 24 Grid

Time	Last period	$\langle x^2 \rangle$
0.0000		0.3513E−07
0.6594E−03		0.1628E−07
0.1306E−02	0.00131	0.3420E−07
0.1966E−02		0.1637E−07

where the integral is performed over the triangles which define the droplet. Tables I–IV give the values of $\langle x^2 \rangle$ and x_r as a function of time for the resolutions of 12 and 24 vertices surrounding the droplet. From the maxima and minima in Tables I and III, we can determine the period of oscillation. It is less well defined from the values of x_r in Tables II and IV. However, it never differs by more than two timesteps from that given by the moments.

Finally, we examined the oscillations of a kerosene droplet in air. This calculation tests the effects of the external fluid density on the numerical convergence of the pressure algorithm as well as any role the external fluid may have in introducing higher frequency modes. Here the theoretical value of the period is 5.9×10^{-4} s. Using a resolution of 12 vertices around the droplet surface, we find a computed value of 7.1×10^{-4} s. The ratio between the theoretical and numerical results is 0.83, compared to a ratio of 0.84 for the 2:1 density ratio calculation at the same resolution. Since changing the density ratio from 2:1 to 650:1 did not alter the relative error, we conclude that only minor errors arise by including of the external fluid in the calculation.

C. *Some Difficulties and Limitations*

We now believe the that the inability of the method to produce as accurate a solution for the droplet oscillation test problem as for the internal wave test problem is a combination of the physical problem itself and the spline approximations to curvature.

TABLE IV

24 × 24 Grid

Time	Last period	x_r
0.0000		0.1500E−01
0.6594E−03		0.1038E−01
0 1250E−02	0 00125	0.1503E−01
0.1931E−02		0.1015E−01

The surface tension algorithm discussed above suffers a basic problem in curve fitting. We are trying to approximate an unknown continuous function by a known curve through a finite number of points or computational cells. For example, we are trying to represent the droplet interface or capillary wave interface by a spline fit to a finite number of points. Whereas an accurate interpolant can be found that goes through a set of points, it is not always clear that the other properties of the curve calculated at the points, e.g., the curvature, are well represented by this interpolant. Splines are notorious for introducing spurious oscillations between the points defining them initially.

Figure 12 shows the curvature at each vertex around the droplet. The exact curvature for the initial drop is compared to the curvature produced by the spline interpolant and to curvatures produced after one oscillation is completed. The initial curvature, defined by splines on the interface vertices, is reasonable. However, by the end of a cycle, there are spurious oscillations even though the curvature has the same basic shape.

In the internal capillary wave problem, the range of values for the interface curvature was a factor of 15 smaller than for the droplet oscillation problem. As a result the interface curvature for the internal capillary wave is determined with greater accuracy. In the droplet oscillation problem where the interface "bends" more sharply, the spline has a greater difficulty approximating the curvatures accurately.

Interpolations can cause other problems in the calculations. Our calculations have shown that the final result can be affected by the location of additional vertices used to obtain a better initial approximation of the droplet interface. The grid initialization procedure involves two phases: a first phase to generate a course grid,

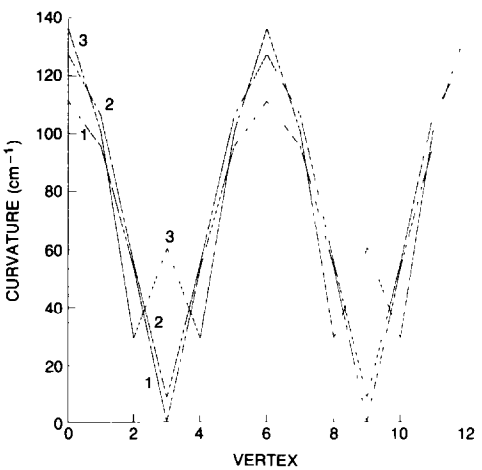

FIG. 12. Curvature as a function of vertex index around the drop in the 16 × 16 calculation: (1) exact solution; (2) initial spline; (3) after one oscillation.

and a second phase which refines the grid produced in the first phase. During the refinement phase, we have two chices for the location of new interface vertices. The initial grid produced in Fig. 8 placed new vertices on the droplet defined by the Rayleigh oscilation mode. We could also add the vertex on the existing spline interpolant. Figure 13 shows $\langle x^2 \rangle$ as a function of time for the two types of initialization. The curve labelled 1 is the calculation in which the additional vertices were placed on the Rayleigh drop. The curve labelled 2 placed the additional vertices on the spline fit. After one oscillation, the value of $\langle x^2 \rangle$ differs by 8%. After one oscillation the value of $\langle x^2 \rangle$ on the curve with label 1 is lower than the initial value of $\langle x^2 \rangle$ and the value of $\langle x^2 \rangle$ on the curve with label 2 is higher than the initial value of $\langle x^2 \rangle$. Notice also that the period, as well as the amplitude, is affected by the type of initialization.

In Fig. 8 we see that the amplitude of the droplet oscillation decays as a function of time even though the period is not changing. The damping rate is about 18% per oscillation. The shape of the droplet at the end of the calculation is notably different than it was at that same place in an earlier oscillation cycle. In the ideal case, this would not occur.

The decay of the oscillation is also apparent in the moment $\langle x^2 \rangle$ and the variation in the location of x_r from oscillation to oscillation. It is apparent from Fig. 14 that the $\langle x^2 \rangle$ moment is dissipating, and it is apparent from Fig. 15 that the overall shape of the droplet is changing. Energy associated with this lowest-mode oscillation is going into other modes, which is reflected in the reduction of the timestep required to keep the computations stable. In general, to carry out these droplet oscillation calculations it was necessary to reduce the timestep to the point where we could resolve the highest mode of oscillation the droplet could support at

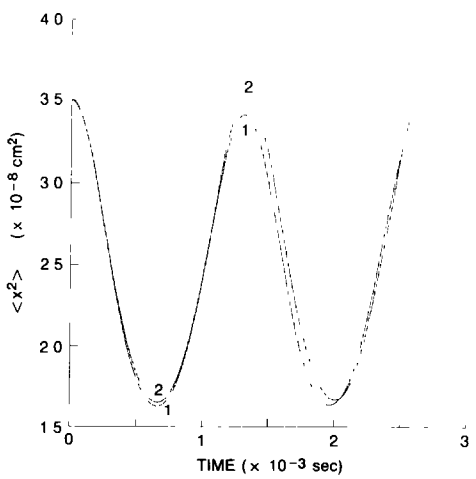

FIG. 13. The moment $\langle x^2 \rangle$ as a function of time for two initializations: (1) all initial vertices on the Rayleigh drop; (2) initial refining vertices on the spline interpolant.

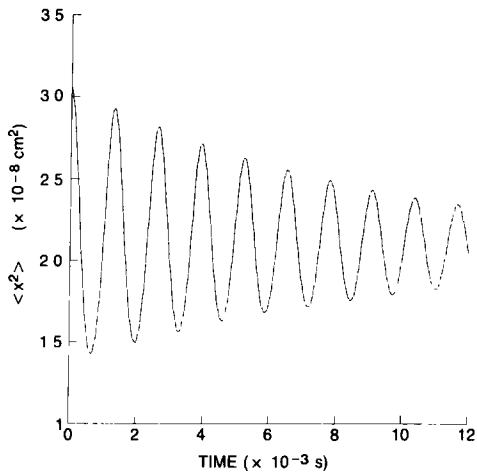

FIG. 14 The moment $\langle x^2 \rangle$ as a function of time in the 16×16 calculation.

a given resolution. When we doubled the resolution around the droplet, we found that the timestep had to be decreased by a factor of about 2.8. This is consistent with the analysis which says that since the period is inversely proportional to $n^{3/2}$, where n is the mode of oscillation. Increasing the resolution of the droplet interface by a factor of 2 means that the timestep must decrease by a factor of $2^{3/2} \approx 2.8$.

A physical mechanism for the observed decay in the $n = 2$ normal mode oscillation is in the existence of a resonance between the $n = 2$ and the $n = 3$ normal modes; that is, $\omega_3 = \pm 2\omega_2$. A similar behavior in three dimensions has been analyzed by Natarajan and Brown [23]. In their three-dimensional analysis,

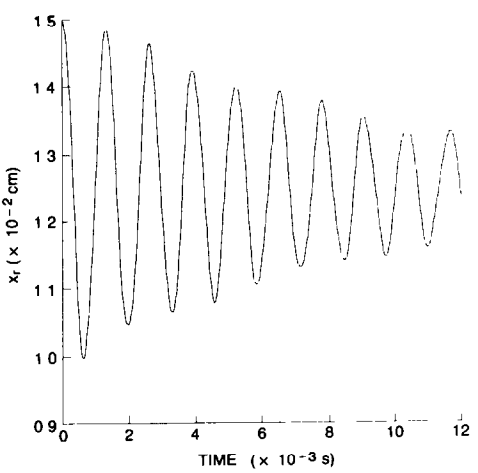

FIG. 15. The position of the rightmost vertex, x_r, as a function of time in the 16×16 calculation.

significant energy can be transferred from one resonant mode to another within ten oscillations.

In summary, the total damping rate for the droplet oscillation calculation is roughly 18% for the $n=2$ mode. Much of the energy from this mode is transferred to higher harmonics, as evidenced by the calculated droplet shapes, curvatures, and the numerical timestep limitations. The difference in initialization procedures alone produced an 8% change in amplitude. Since the total numerical error is 5.5%, we conclude that the majority of this error arise from the inability of the spline fit to approximate large curvatures accurately. This error is large enough to mask other error contributions, so that we cannot evaluate additional error terms other than to indicate that they are apparently much smaller than that due to the spline fit.

However, despite all the problems with spline fits, we found that they provided a good way to calculate curvature. In the search for better curvatures, we have also tried other methods, none of which produced better results. We enumerate these attempts both for completeness and to emphasize that better numerical approximations for curvature are still needed to permit more accurate calculations of surface tension.

(1) We averaged the curvature between the vertices, such that

$$K_i = \frac{1}{s_{i+1} - s_{i-1}} \int_{s_{i-1}}^{s_{i+1}} K(s)\, ds.$$

The results were found to depend sensitively how how the integral was actually performed. However, integration produced results which were no better than the pointwise curvatures discussed above.

(2) We smoothed the curvatures K_i with a least squares linear spline. This method worked well for one period, but the method failed on subsequent oscillations.

(3) We used a circular arc to calculate the curvatures. A circle was placed through the three adjoining vertices. The radius of that circle was used as the radius of curvature for the interface at the center vertex of the three vertices. This method did not work at all. The droplet interface distorted wildly within the first oscillation.

(4) We used splines under tension. This approach introduced a free parameter which could not be consistently determined.

(5) Based on the experiences of Foote [20], we tried producing an interpolant through every other vertex and averaging the result. The motivation was that fewer points could introduce fewer oscillations, and that averaging the interpolants could damp the oscillations. This produced poor results. The calculation is really the average of two calculations with half the original accuracy.

(6) We considered but did not implement nonlinear splines [24]. Although these splines produce differentiable curvatures, there is no guarantee that there

exists such a spline through a given set of points and, if such a spline does exist, there is no guarantee that it is unique.

(7) We considered several methods for calculating an interpolant based on the Rayleigh modes. The high mode oscillations could then be eliminated. None of the schemes we considered gave better results than the spline interpolant, and all introduced arbitrary parameters into the calculation. These parameters could be well determined for a particular known shape, but could not be determined for a general unknown shape.

IV. Incompressible Flow about a Droplet

In this section we present some preliminary calculations of forced, asymmetric drop oscillations induced by flow around a droplet. These calculations include both the effects of viscosity and surface tension. The capability of studying such flows for highly viscous droplets in shear flows, in two and eventually in three dimensions, is the motivation for developing the viscosity and surface tension algorithms.

The initial conditions we used specified an initially steady-state potential flow about a periodic series of cylinders. Again, the boundary conditions on the left and right sides are periodic, and the upper and lower boundary conditions are reflecting walls. Initially, a perfectly circular droplet is at rest in a background flow. A physical situation modelled by such an initialization might occur if the flow velocity were ramped up to its final value before any significant structure could develop in the flow, and before the droplet could pick up any substantial velocity. Basically, it is a smooth start for the calculation. Previously we had performed calculations which began with an impulsive start, but found that as a result there was a large amount of momentum transferred across the droplet interface early in the calculation.

The calculations presented here model the forced fluid flow due to a fast air stream about an initially stationary kerosene droplet. The physical parameters, given in Table V, are appropriate for a combustor environment. A total of 309 vertices were used to initialize the problem, with 12 vertices at the droplet interface. Figure 16 follows the evolution of pathlines in the internal and external flow fields

TABLE V

Density of kerosene	0.82 g/cc
Density of air	0.0013 g/cc
Surface tension (STP)	30 dynes/cm
Viscosity of kerosene	1.8 cp
Viscosity of air	0.018 cp
Air velocity	100 or 120 m/s
Initial droplet velocity	0.0 m/s
Droplet radius	125 μm

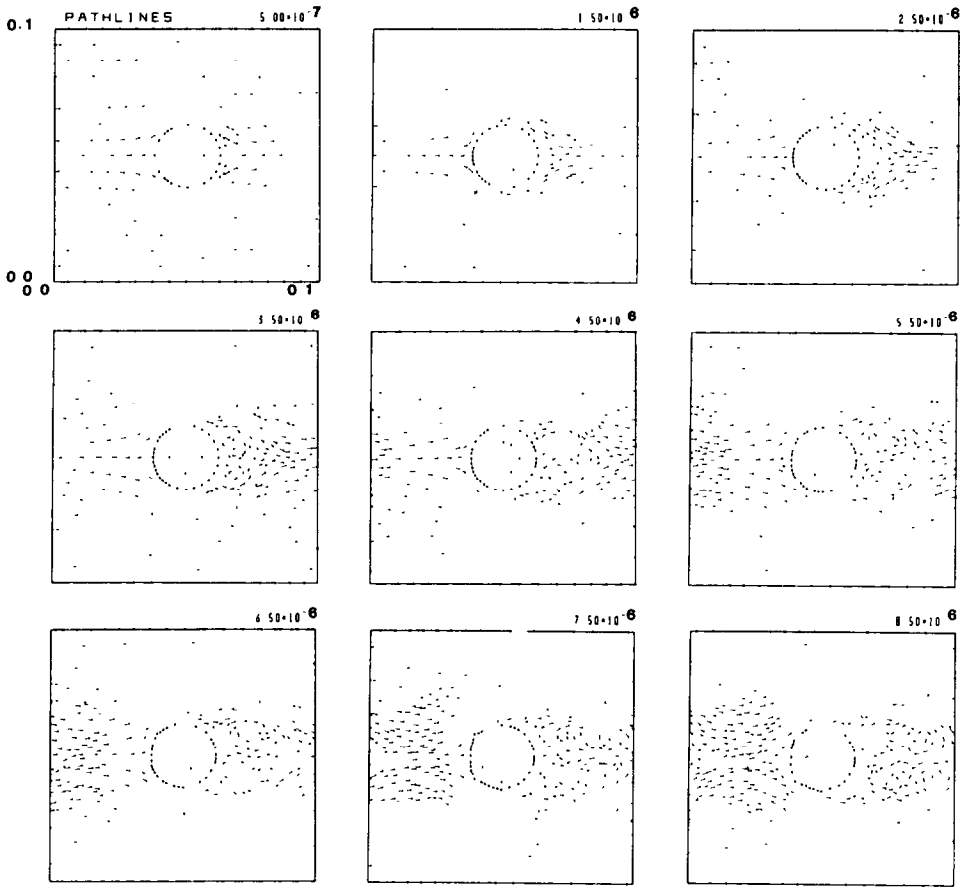

FIG. 16. Pathlines from a calculation of air flowing past a deforming, viscous kerosene droplet. Surface tension forces are included at the material interface. Heads of pathlines are the current vertex positions and the tails are made up of the previous five positions. The flow speed is 100 m/s and $Re \approx 1600$

through a series of timesteps. For an air velocity of 100 m/s and a droplet radius of 125 μm, the corresponding Reynolds number is roughly 1600. The pathlines are defined by the paths of vertices over five timesteps. By the last frame of Fig. 16, the fluid originally to the left of the droplet has progressed through the mesh and interacted with the face of the (next) droplet.

The first clear indication of the development of the recirculation region is seen in the fourth frame of Fig. 16, which shows a pair of counter-rotating vortices. The recirculation zone continues to develop throughout the calculation, although at times the vortex pair is not as evident due to the deletion and addition of vertices, which interrupt the continuity of the pathlines. By the last frame, another pair of vortices is forming near the droplet, and the original pair has been shed. The

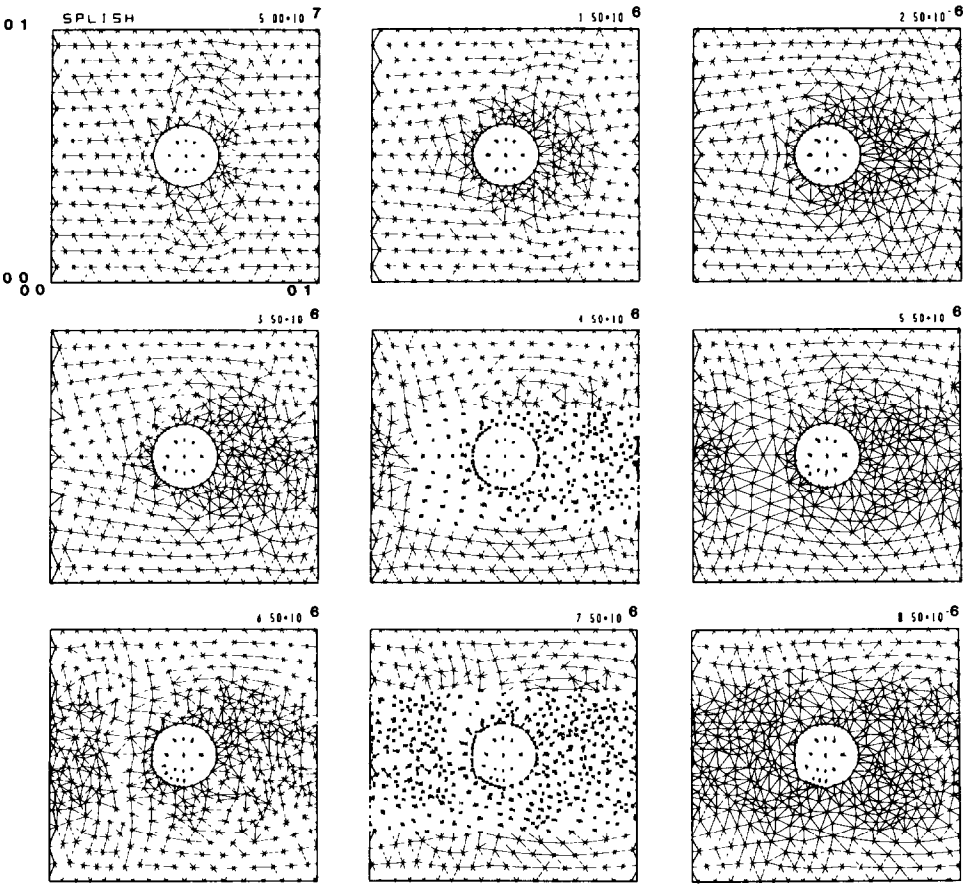

FIG. 17. Frames showing in triangular grid at the same times as shown for the pathlines in Fig. 14.

leading face of the droplet is now quite distorted, and the droplet is about to enter the wake of the preceding droplet.

Distortions in the face of the droplet are evident in at least the seventh frame. These distortions occur because the curvature has increased and the streamlines in the external flow are condensed by the approaching wake. The internal velocities are small compared to the external flow rates and therefore cannot be distinguished as pathlines. However, indication of the (small) internal recirculation may be obtained by comparing internal vertex positions at various timeteps.

Figure 17 shows the grid at times in the calculation corresponding to those in Fig. 16. During the course of the calculation, a great deal of vertex addition and deletion has occured. Vertex addition, however, is most noticeable in the wake of the droplet and around the droplet interface. Whereas there were 300 vertices at the beginning of the calculation, there are 450 at the end.

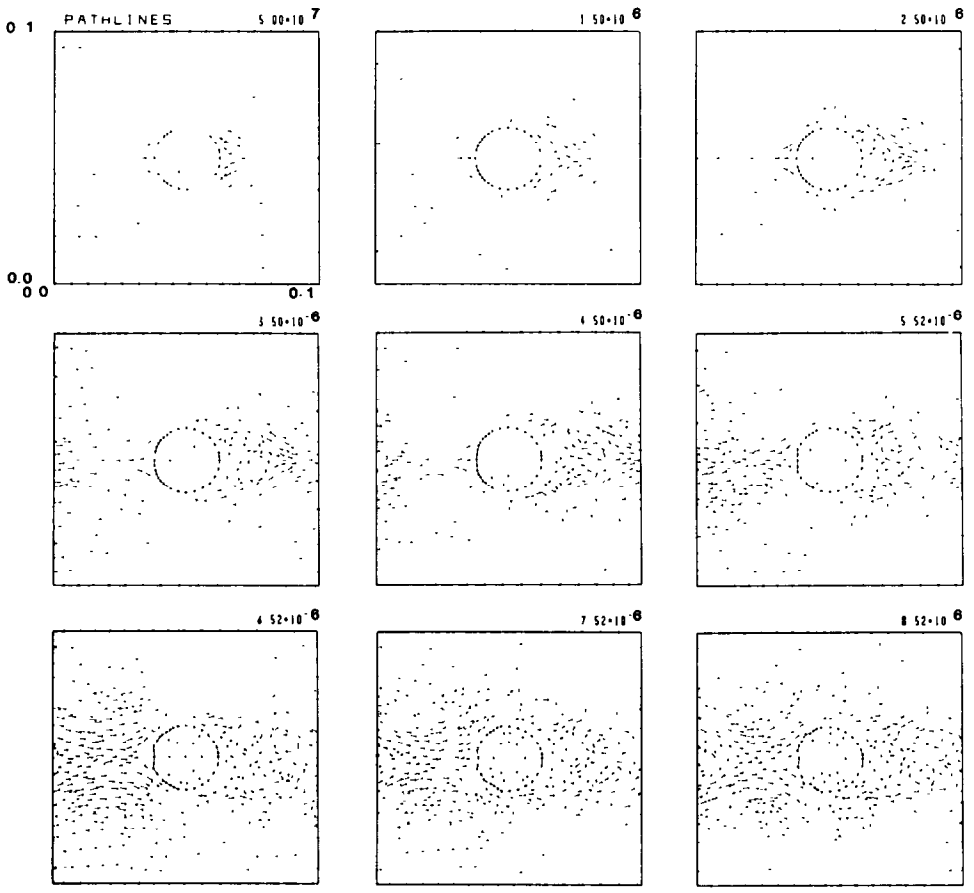

FIG 18. Pathlines from a calculation similar to that shown in Fig. 14, but with a flow velocity of 120 m/s and Re ≈ 2000.

Figure 18 shows the pathlines for a simulation with the air speed increased to 120 m/s, corresponding to a Reynolds number of 2000. The fluid now completely passes through the mesh. The fluid initially near the droplet has completely passed the next droplet by the time the calculation was terminated. The initial flow about the droplet is similar to that shown above, except for a more pronounced flattening at the face of the droplet due to the higher flow speed. The wake develops in much the same manner, but it now interacts strongly with the flow at the forward stagnation point on the droplet. Oscillations in the flow due to the wake are transmitted to the forward face of the droplet and give rise to fairly large perturbations.

As seen in Fig. 19, the computational grid needs further refinement at this time because the perturbations cannot be resolved by the limits set on minimum triangle size originally chosen for the calculation. A sign that the calculation is under-

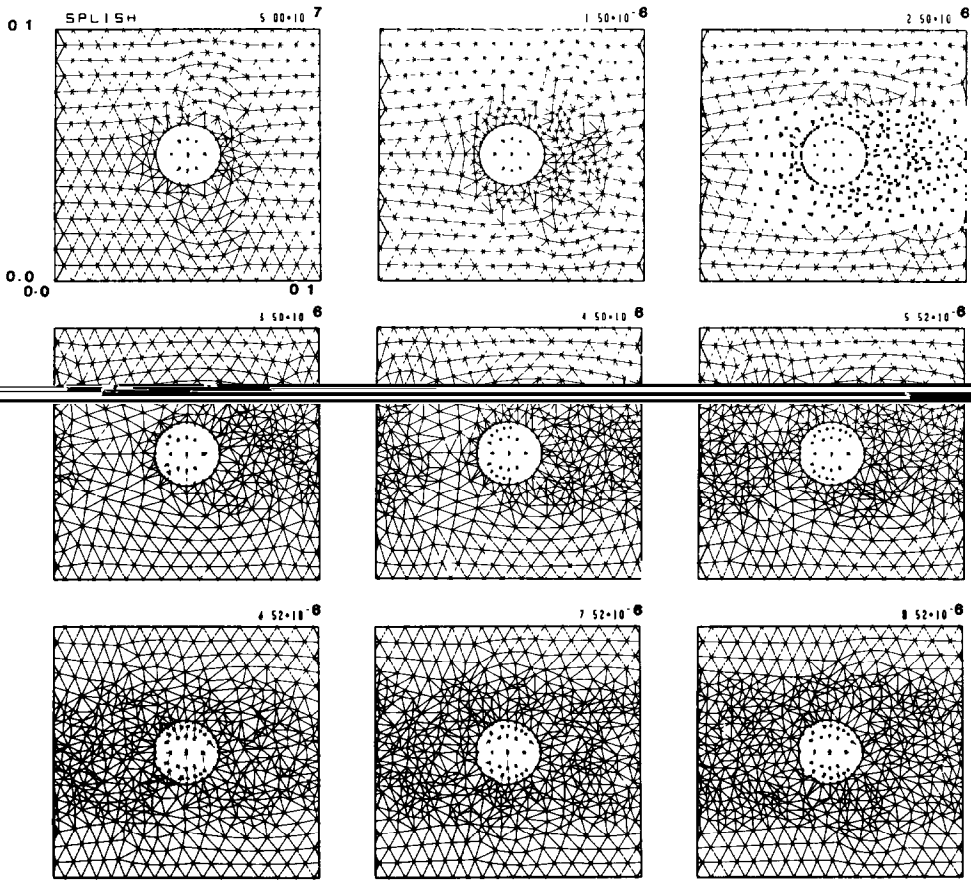

FIG. 19. Frames showing the triangular grid at the same times as shown for the pathlines in Fig. 16.

resolved is that one of the crests of the surface wave is spanned by a single triangle, a situation which allows no communication of that surface fluid with the interior of the droplet. In order to continue the simulation, better resolution must be obtained about the droplet surface. Another algorithm is currently being included to allow higher resolution near points of large curvature at material interfaces.

V. SUMMARY AND CONCLUSIONS

This paper presented the current algorithms included in the code SPLISH, a two-dimensional Cartesian Lagrangian treatment of incompressible flows with a dynamically restructuring grid. The new rotator algorithm is an improvement on the one previously used for conserving circulation. The residual algorithm ensures

conservation of the area of cells. These algorithms together with the original SPLISH framework constitute an extremely flexible code for calculating incompressible flows in highly distorted geometries or with obstacles in the flow.

New algorithmms for modelling the physical effects of viscosity and surface tension have been added. Whereas adding the viscosity algorithm was relatively straightforward, adding surface tension caused a number of numerical problems. Detailed benchmarks of the final algorithm selected were presented using internal capillary waves and a Rayleigh oscillating droplet as test problems. The surface tension algorithm, based on spline fits to determine curvature, allowed the droplet to oscillate many times and still maintain a constant period. The numerical tests on the internal capillary waves indicate that the surface tension algorithm produced a convergence rate which is linear in the mesh size, whereas the basic hydrodynamic algorithms are quadratic in the mesh size for ideal meshes. The droplet oscillation test problem, however, indicated some difficulties with the spline fits for curvature when the interface becomes highly distorted.

Previous numerical calculations of oscillating spherical droplets with surface tension and viscosity using a marker-and-cell method showed only one oscillation of a water droplet in air [20], and thus did not give any information about the subsequent behavior of the mode amplitudes. These calculations used 2.5 times the resolution of our most resolved calculations. Their calculated period differed from the theoretical period by 6% compared to our 12% for a similar initial deformation. Their viscous calculations failed to damp as quickly as required by theory which may indicate that they suffer from a similar problem of approximating curvatures.

We presented calculations showing how a kerosene droplet deforms and sheds vortices in the wake of a shear flow. Calculations of fluid flow in and around fuel droplets are important in the study of spray combustors. The flow patterns influence droplet breakup, evaporation, and burning rates.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Air Force Office of Scientific Research, the Air Force Wright Aeronautical Laboratories, and the Naval Research Laboratory through the Office of Naval Research. The authors acknowledge the helpful suggestions of Jay Boris.

REFERENCES

1. M. J FRITTS AND J. P. BORIS, *J. Comput. Phys.* **31**, 173 (1979).
2. M J. FRITTS, SAIC Report SAI-76-528-WA, Science Applications International, Inc., March, 1976 (unpublished).
3. M. J. FRITTS AND J. P. BORIS, in *Second International Conference on Numerical Ship Hydrodynamics*, edited by J. V Wehausen and N. Salvesen (University of California, Berkeley, 1977), p. 319.
4. M. J. FRITTS, E. W MINER, AND O M. GRIFFIN, in *Computer Methods in Fluids*, edited by K Morgan, C. Taylor, and C. A. Brebbia (Pentech Press, London, 1980), p. 1.

5. M J. FRITTS, SAIC Report SAI-76-632-WA, Science Applications International Inc., September, 1976 (unpublished).
6. M. H. EMERY, S. E BODNER, J. P BORIS, D. G. COLOMBANT, A. L. COOPER, M J. FRITTS, AND M. J HERBST, NRL Memorandum Report 4947, Naval Research Laboratory, Washington DC 1982 (unpublished).
7. M. H. EMERY, M J FRITTS, AND R. C. SHOCKLEY, NRL Memorandum Report 4569, Naval Research Laboratory, Washington, 1981 (unpublished).
8. R. GENTRY, private communication (1965).
9. B. J DALY AND F. H. HARLOW, Los Alamos National Laboratory Report LA-3144, April, 1965 (unpublished).
10. F. H. HARLOW, Los Alamos National Laboratory Report LA-2806, November, 1962 (unpublished).
11. W. P. CROWLEY, in *Proceedings, Second International Conference on Numerical Methods in Fluid Dynamics*, Lecture Notes in Physics Vol. 8, edited by M. Holt (Springer-Verlag, New York, 1971)
12. A. BARCILON, D. BOOK, J. BORIS, A. COOPER, K. HAIM, P LIEWER, A. ROBSON, R. SHANNY, P. TURCHI, AND N. WINSOR, in *Plasma Physics and Controlled Nuclear Fusion Research*, IAEA-CN-33 (IAEA, Vienna, 1974), p 567
13. C. S. PESKIN, *J. Comput Phys.* **25**, 220 (1977)
14. M. J FRITTS, W P CROWLEY, AND H. TREASE (Eds.), *The Free-Lagrange Method*, Lecture Notes in Physics Vol. 238 (Springer-Verlag, New York, 1985).
15. G STRANG AND G. FIX, *An Analysis of the Finite Element Method* (Prentice–Hall, Englewood Cliffs, NJ, 1973), pp. 77–78
16. M. J. FRITTS, D. E. FYFE, AND E. S. ORAN, "Numerical Simulations of Droplet Flows with Surface Tension," Proceedings, ASME Workshop on Two-Phase Flows, Phoenix, Arizona, 1982.
17. M. J. FRITTS, D. E. FYFE, AND E. S. ORAN, NASA CR-168263, 1983 (unpublished).
18. L. D. LANDAU AND E. M. LIFSHITZ, *Fluid Mechanics* (Pergammon, New York, 1959), pp 230–234.
19. B. J. DALY, *J. Comput. Phys.* **4**, 97 (1969).
20. G B. FOOTE, *J Comput. Phys.* **11**, 507 (1973)
21. C. DE BOOR, *A Practical Guide to Splines* (Springer-Verlag, New York, 1978), pp. 316–322.
22. LORD RAYLEIGH, *Proc. R. Soc. London* **29**, 71 (1879).
23. R. NATARAJAN AND R BROWN, *Phys. Fluids* **29**, 2788 (1986).
24. M. A. MALCOLM, *SIAM J Numer Anal.* **14**, 320 (1977).