JOURNAL OF COMPUTATIONAL PHYSICS 39, 201–225 (1981)

Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries*

C. W. HIRT AND B. D. NICHOLS

Los Alamos Scientific Laboratory, Los Alamos, New Mexico 87545

Received November 1, 1979

Several methods have been previously used to approximate free boundaries in finite-difference numerical simulations. A simple, but powerful, method is described that is based on the concept of a fractional volume of fluid (VOF). This method is shown to be more flexible and efficient than other methods for treating complicated free boundary configurations. To illustrate the method, a description is given for an incompressible hydrodynamics code, SOLA-VOF, that uses the VOF technique to track free fluid surfaces.

1. INTRODUCTION

In structural dynamics, it is customary to employ Lagrangian coordinates as the basis for numerical solution algorithms. In fluid dynamics, however, both Lagrangian and Eulerian coordinates have been used with considerable success. Because each coordinate representation has unique advantages and disadvantages, the choice of which representation to use depends on the characteristics of the problem to be solved. In this paper the emphasis is on Eulerian formulations for problems involving free boundaries, in particular, problems where the free boundaries undergo such large deformations that Lagrangian methods cannot be used.

Free boundaries are here considered to be surfaces on which discontinuities exist in one or more variables. Examples are free surfaces, material interfaces, shock waves, or interfaces between fluid and deformable structures. Three types of problems arise in the numerical treatment of free boundaries: (1) their discrete representation, (2) their evolution in time, and (3) the manner in which boundary conditions are imposed on them. In Section II, a short review is given of different methods that have been used for embedding free boundaries in finite-difference or finite-element grids. A comparison of the relative advantages and disadvantages of these methods leads to a new technique that is simple yet powerful. This method, the volume of fluid (VOF) method, is described in Section III. In Section IV, details of the VOF method are described as it has been implemented in an Eulerian hydrodynamics code. The new code, SOLA-VOF, is illustrated in Section V with various examples that show the

* The U. S. Government's right to retain a nonexclusive royalty-free license in and to the copyright covering this paper, for governmental purposes, is acknowledged. This work was performed under the auspices of the U. S. Department of Energy.

0021-9991/81/010201-25$02.00/0

strength of the VOF technique for treating problems involving highly complicated free surface flows. Finally, in Section VI, a short summary is provided that emphasizes the advantages of the new code.

II. Free Boundary Methods

Discrete Lagrangian representations for a fluid are conceptually simple because each zone of a grid that subdivides the fluid into elements remains identified with the same fluid element for all time. Body and surface forces on these elements are easy to define, so it is relatively straightforward to compute the dynamic response of the elements. In an Eulerian representation the grid remains fixed and the identity of individual fluid elements is not maintained. Nevertheless, it is customary to view the fluid in an Eulerian mesh cell as a fluid element on which body and surface force may be computed, in a manner competely analogous to a Lagrangian calculation. The two methods differ, however, in the manner in which the fluid elements are moved to nex positions after their new velocities have been computed. In the Lagrangian case the grid simply moves with the computed element velocities, while in an Eulerian or Arbitrary Lagrangian–Eulerian [1] calculation it is necessary to compute the flow of fluid through the mesh. This flow, or convective flux calculation, requires an averaging of the flow properties of all fluid elements that find themselves in a given mesh cell after some period of time. It is this "averaging process," inherent in convective flux approximations, that is the biggest drawback of Eulerian methods. Convective averaging results in a smoothing of all variations in flow quantities, and, in particular, a smearing of surfaces of discontinuity such as free surfaces. The only way to overcome this loss in resolution for free boundaries is to introduce some special treatment that recognizes a discontinuity and avoids averaging across it.

As already noted, the process of embedding a discontinuous surface in a matrix of computational cells involves three separate tasks. First, it is necessary to devise a means of numerically describing the location and shape of the boundary. Second, an algorithm must be given for computing the time evolution of the boundary. Finally, a scheme must be provided for imposing the desired surface boundary conditions on the surrounding computational mesh. The first two problems are related because the method of description will govern the choice of evolution algorithm. On the other hand, the application of boundary conditions is largely independent of how the surface is defined.

In the remainder of this section, we shall concentrate on the representation and evolution problems. We shall also restrict this discussion to two-dimensional situations, except for a few remarks concerning analogous three-dimensional methods.

A. *Height Functions*

A simple means of representing a free boundary is to define its distance from a reference line as a function of position along the reference line. For example, in a

rectangular mesh of cells of width δx and height δy one might define the vertical height, h, of the free boundary above the bottom of the mesh in each column of cells. This would approximate a curve $h = f(x, t)$ by assigning values of h to discrete values of x. This method does not work well when the boundary slope, dh/dx, exceeds the mesh cell aspect ratio $\delta y/\delta x$, and does not work at all for multiple-valued surfaces having more than one y value for a given x value. This is a severe limitation because many simple shapes, such as bubbles or drops, cannot be treated. However, when it can be used, this representation is extremely efficient, requiring only a one-dimensional storage array to record the surface height values. Likewise, the evolution of the surface only requires the updating of the one-dimensional array (see, for example, Ref. [2]).

In the case of a free fluid boundary, the time evolution of the height function is governed by a kinematic equation expressing the fact that the surface must move with the fluid,

$$\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} = v, \tag{1}$$

where (u, v) are fluid velocity components in the (x, y) coordinate directions. It should be noted that Eq. (1) is Eulerian in the horizontal direction, but Lagrangian-like in the vertical direction, which is more or less normal to the surface. Finite-difference approximations to this equation are easily made [2].

The height function method is directly extendable to three-dimensional situations [3] for single-valued surfaces describable by, e.g., $h = f(x, y, t)$.

B. *Line Segments*

A generalization of the height function method uses chains of short line segments, or points connected by line segments (e.g., Ref. [4]). Coordinates for each point must be stored and for accuracy it is best to limit the distance between neighboring points to less than the minimum mesh size δx or δy. Therefore, slightly more storage is required for this method, but it is not limited to single-valued surfaces.

The evolution of a chain of line segments is easily accomplished by simply moving each point with the local fluid velocity determined by interpolation in the surrounding mesh. In this sense the line segment method resembles a Lagrangian mesh line. It is more flexible, however, because individual segments may be readily deleted or added as required for optimal resolution. Since the segments are linearly ordered, the deletion–addition process presents no logical problems.

Unfortunately, there is one serious difficulty with the line segment method. When two surfaces intersect, or when a surface folds over on itself, segment chains must be reordered, possibly with the addition or removal of some chains. If such intersections are anticipated, the reordering process may not be difficult. In the general case, however, the detection of intersections and determining how a reordering should be done is not a trivial task.

The extension of the line segment method to three-dimensional surfaces is also nontrivial [5]. Linear ordering used for two-dimensional lines does not work for three-dimensional surfaces. Thus, the determination of neighboring points defining the local surface configuration requires a major effort. Similarly the determination of surface intersections and addition–deletion algorithm is considerably more complex.

C. *Marker Particles*

Instead of defining a free surface directly, one can also work with the regions occupied by fluid. For example, marker particles can be spread over all fluid occupied regions with each particle specified to move with the fluid velocity at its location [6]. Clearly, storage requirements increase significantly with this method because of the large increase in the number of point coordinates that must be stored. Surfaces are defined as lying at the "boundary" between regions with and without marker particles. More specifically, a mesh cell containing markers, but having a neighboring cell with no markers, is defined as containing a free surface. The actual location of the free surface must be determined by some additional computation based on the distribution of markers within the cell.

Marker particle methods offer the distinct advantage of eliminating all logic problems associated with intersecting surfaces. This is primarily a consequence of the fact that while particles have to be ordered with well-defined neighbors when marking surfaces they do not have to be well ordered when marking regions. The marker particle method is also readily extendable to three-dimensional computations, provided the increased storage requirements can be tolerated [7].

In retrospect, it appears that a method that defines fluid regions rather than interfaces offers the advantage of logical simplicity for situations involving interacting multiple free boundaries. While the marker particle method provides this simplicity, it suffers from a significant increase in required computer storage. It also requires additional computational time to move all the points to new locations. It is natural, therefore, to seek an alternative that shares the region defining property without an excessive use of computer resources. Such a method is described in the next section.

III. The Volume of Fluid (VOF) Method

In each cell of a mesh it is customary to use only one value for each dependent variable defining the fluid state. The use of several points in a cell to define the region occupied by fluid, therefore, seems unnecessarily excessive. Suppose, however, that we define a function F whose value is unity at any point occupied by fluid and zero otherwise. The average value of F in a cell would then represent the fractional volume of the cell occupied by fluid. In particular, a unit value of F would correspond to a cell full of fluid, while a zero value would indicate that the cell contained no fluid. Cells with F values between zero and one must then contain a free surface. Thus, the

fractional volume of fluid (VOF) method [5] provides the same coarse interface information available to the marker particle method. Yet the VOF method requires only one storage word for each mesh cell, which is consistent with the storage requirements for all other dependent variables.

In addition to defining which cells contain a boundary, marker particles also define where fluid is located in a boundary cell. Similar information can be obtained in the VOF method. The normal direction to the boundary lies in the direction in which the value of F changes most rapidly. Because F is a step function, however, its derivatives must be computed in a special way, as described below. When properly computed, the derivatives can then be used to determine the boundary normal. Finally, when both the normal direction and the value of F in a boundary cell are known, a line cutting the cell can be constructed that approximates the interface there. This boundary location can then be used in the setting of boundary conditions.

Although the VOF technique can locate free boundaries nearly as well as a distribution of marker particles, and with a minimum of stored information, the method is worthless unless an algorithm can be devised for accurately computing the evolution of the F field. The time dependence of F is governed by the equation,

$$\frac{\partial F}{\partial t} + u\frac{\partial F}{\partial x} + v\frac{\partial F}{\partial y} = 0. \tag{2}$$

This equation states that F moves with the fluid, and is the partial differential equation analog of marker particles. In a Lagrangian mesh, Eq. (2) reduces to the statement that F remains constant in each cell. In this case, F serves solely as a flag identifying cells that contain fluid. In an Arbitrary Lagrangian–Eulerian mesh, the flux of F moving with the fluid through a cell must be computed, but as noted in Section II, standard finite-difference approximations would lead to a smearing of the F function and interfaces would lose their definition. Fortunately, the fact that F is a step function with values of zero or one permits the use of a flux approximation that preserves its discontinuous nature. This approximation, referred to as a donor–acceptor method [8], is described in more detail in Section IV (Subsection D).

In summary, the VOF method offers a region-following scheme with minimum storage requirements. Furthermore, because it follows regions rather than surfaces, all logic problems associated with intersecting surfaces are avoided with the VOF technique. The method is also applicable to three-dimensional computations, where its conservative use of stored information is highly advantageous.

Thus, the VOF method provides a simple and economical way to track free boundaries in two- or three-dimensional meshes. In principle, the method could be used to track surfaces of discontinuity in material properties, in tangential velocity, or any other property. The particular case being represented determines the specific boundary condition that must be applied at the location of the boundary. For situations where the surface does not remain fixed in the fluid, but has some additional relative motion, the equation of motion, Eq. (2), must be modified.

Examples of such applications are shock waves, chemical reaction fronts, and boundaries between single-phase and two-phase fluid regions.

In the next section a program is presented for using the VOF method to define free surfaces in an Eulerian hydrodynamics code.

IV. SOLA-VOF

Eulerian finite-difference methods for computing the dynamics of incompressible fluids are well established. The first method to successfully treat problems involving complicated free surface motions was the marker-and-cell (MAC) method [6]. This method was also the first technique to use pressure and velocity as the primary dependent variables. MAC employed a distribution of marker particles to define fluid regions, and simply set free surface pressures at the centers of cells defined to contain the surface. No attempt was made to apply the pressure boundary condition at the actual location of the boundary within the surface containing cell. This crude approximation was later improved [9], and marker particles were eliminated in favor of particle chains on the free surfaces [4].

A simplified version of the basic solution algorithm (SOLA) used in the MAC method is available [10] in a user-oriented code called SOLA. Although SOLA does not treat free surfaces, an extended version, SOLA-SURF, is also available [10] that uses the surface height function method (see Section II.A). The basic simplicity and flexibility of the SOLA codes make them excellent foundations for the development of more sophisticated codes. For this reason, a variable mesh version of the SOLA code, SOLA-VM, was chosen as a basis for illustrating the VOF technique. An experimental version of this new code, SOLA-VOF, was first reported in Ref. [5]. Since that time, many improvements have been made and the basic technique has matured through applications to a wide class of problems. In a related development [11], McMaster and co-workers have recently combined the SOLA-SURF code with a different interface tracking technique based on a VOF-like concept [12].

The following subsections provide a description of the SOLA-VM solution algorithm with particular attention devoted to the special considerations needed in making finite-difference approximations in nonuniform meshes. Subsequent subsections describe the VOF algorithms for advection and for locating interfaces.

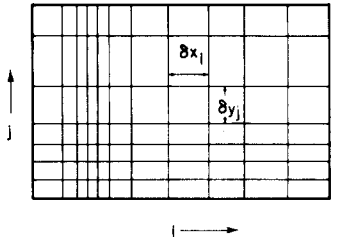

FIG. 1. Schematic of finite-difference mesh with variable rectangular cells.

A. *Outline*

SOLA-VM uses an Eulerian mesh of rectangular cells having variables sizes, δx_i for the ith column and δy_j for the jth row, as shown in Fig. 1. While not as flexible as a mesh composed of arbitrary quadrilaterals, the variable mesh (VM) capability of SOLA-VM gives it a considerable advantage over methods using equal-sized rectangles.

The fluid equations to be solved are the Navier–Stokes equations,

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial p}{\partial x}+g_x+\nu\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\xi\left(\frac{1}{x}\frac{\partial u}{\partial x}-\frac{u}{x^2}\right)\right],$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{\partial p}{\partial y}+g_y+\nu\left[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\xi}{x}\frac{\partial v}{\partial x}\right]. \tag{3}$$

Velocity components (u, v) are in the Cartesian coordinate directions (x, y) or cylindrical coordinate directions (r, z), respectively. The choice of coordinate system is governed by the value of ξ, where $\xi=0$ corresponds to Cartesian and $\xi=1$ to cylindrical geometry. Body accelerations are denoted by (g_x, g_y) and ν is the coefficient of kinematic viscosity. Fluid density has been normalized to unity. For an incompressible fluid, the momentum equations, Eqs. (3), must be supplemented with the incompressibility condition,

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\xi u}{x}=0. \tag{4}$$

Sometimes, it is desirable to allow limited compressibility effects [13] (e.g., acoustic waves) in which case Eq. (4) must be replaced with

$$\frac{1}{c^2}\frac{\partial p}{\partial t}+\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\xi u}{x}=0, \tag{5}$$

where c is the adiabatic speed of sound in the fluid (and the mean density is unity). Since Eq. (5) adds more flexibility with little additional complexity, it is used in the remainder of this discussion.

Discrete values of the dependent variables, including the fractional volume of fluid (F) variable used in the VOF technique, are located at cell positions shown in Fig. 2.

The volume of fluid function F is used to identify mesh cells that contain fluid. A free surface cell (i, j) is defined as a cell containing a nonzero value of F *and* having at least one neighboring cell, $(i \pm 1, j)$ or $(i, j \pm 1)$, that contains a zero value of F. Cells with zero F values are called empty cells, and cells with nonzero F values and no empty neighbors are treated as full or interior fluid cells. The SOLA-VOF code also has provisions for defining any cell or combination of cells in the mesh to be obstacle cells into which fluid cannot flow.

Briefly, the basic procedure for advancing a solution through one increment in time, δt, consists of three steps:

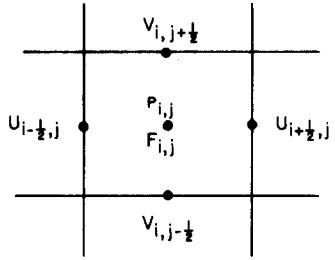

FIG. 2. Location of variables in a typical mesh cell.

(1) Explicit approximations of Eq. (3) are used to compute the first guess for new time-level velocities using the initial conditions or previous time-level values for all advective, pressure, and viscous accelerations.

(2) To satisfy the continuity equation, Eq. (5), pressures are iteratively adjusted in each cell and velocity changes induced by each pressure change are added to the velocities computed in step (1). An iteration is needed because the change in pressure needed in one cell to satisfy Eq. (5) will upset the balance in the four adjacent cells.

(3) Finally, the F function defining fluid regions must be updated to give the new fluid configuration.

Repetition of these steps will advance a solution through any desired time interval. At each step, of course, suitable boundary conditions must be imposed at all mesh and free-surface boundaries. Because most of these details have been presented elsewhere (Refs. [2, 4, 6, 13]), they will not be repeated here. In the remaining sections emphasis will be on those features that represent extensions of the previously reported methodology. In particular, special considerations required when dealing with nonuniform meshes and the details of the VOF method for defining and tracking interfaces are discussed.

B. *Variable Mesh Approximations*

In the following, the notation $Q^n_{i,j}$ stands for the value of $Q(x, y, t)$ at time $n\,\delta t$ and at a location centered in the ith cell in the x-direction and jth cell in the y-direction. Half-integer subscripts refer to cell boundary locations. For example, $Q^n_{i,j+1/2}$, refers to the value of Q on the boundary between the j and $j+1$ cells in the y-direction.

A generic form for the finite-difference approximation of Eq. (3) in MAC-type methods is

$$u^{n+1}_{i+1/2,j} = u^n_{i+1/2,j} + \delta t \left[-\left(p^{n+1}_{i+1,j} - p^{n+1}_{i,j}\right) \Big/ \delta x_{i+1/2} + g_x - \mathrm{FUX} - \mathrm{FUY} + \mathrm{VISX} \right],$$
$$v^{n+1}_{i,j+1/2} = v^n_{i,j+1/2} + \delta t \left[-\left(p^{n+1}_{i,j+1} - p^{n+1}_{i,j}\right) \Big/ \delta y_{j+1/2} + g_y - \mathrm{FVX} - \mathrm{FVY} + \mathrm{VISY} \right]. \tag{6}$$

Here $\delta x_{i+1/2} = 1/2(\delta x_i + \delta x_{i+1})$ and $\delta y_{j+1/2} = 1/2(\delta y_j + \delta y_{j+1})$. The advective and viscous acceleration terms have an obvious meaning; e.g., FUX means the advective flux of u in the x-direction, etc. These terms are all evaluated using the old time level (n) values for velocities.

As far as the basic solution procedure is concerned, the specific approximations chosen for the advective and viscous terms in Eq. (6) are relatively unimportant, provided they lead to a numerically stable algorithm. Special care must be exercised, however, when making aproximations in a variable mesh like that of Fig. 1. The problem is best illustrated by considering the procedure used in the original MAC method for Cartesian coordinates. In the MAC method, Eqs. (3) and (4) were first combined so that the convective flux terms could be written in a divergence form (i.e., $\nabla \cdot \mathbf{uu}$ instead of $\mathbf{u} \cdot \nabla \mathbf{u}$). Thus, FUX would be, for example, $\partial u^2/\partial x$ rather than $u(\partial u/\partial x)$. The divergence form was preferred in MAC because it provided a simple way to ensure conservation of momentum in the difference approximations. This may be seen by considering the control volume used for $u_{i+1/2,j}$ that is indicated by dashed lines in Fig. 3. With the divergence form, Gauss' theorem may be used to convert the integrated value of FUX over the control volume to boundary fluxes at its sides. Then, the flux leaving one control volume will automatically be gained by the adjacent volume and conservation during advection is guaranteed.

Unfortunately, conservation in a variable mesh does not automatically imply accuracy. To see this, suppose an upstream or donor-cell difference approximation is used for $\text{FUX} = \partial u^2/\partial x$, which is known to provide a conditionally stable algorithm. Assuming the u velocity is positive, the donor cell approximation is

$$\text{FUX} = [u_{i+1,j}\langle u_{i+1,j}\rangle - u_{i,j}\langle u_{i,j}\rangle]\,\delta x_{i+1/2}, \tag{7}$$

where, for example,

$$u_{i,j} = (u_{i-1/2,j} + u_{i+1/2,j})/2;$$

$$\langle u_{i,j}\rangle = u_{i-1/2,j}, \qquad \text{if} \quad u_{i,j} \geqslant 0,$$

$$= u_{i+1/2,j}, \qquad \text{if} \quad u_{i,j} < 0.$$

Expanding Eq. (7) in a Taylor series about the location, $x_{i+1/2}$, where the u-equation is evaluated, yields

$$\text{FUX} = \frac{1}{2}\left(\frac{3\delta x_i + \delta x_{i+1}}{\delta x_i + \delta x_{i+1}}\right)\frac{\partial u^2}{\partial x} + 0(\delta x). \tag{8}$$

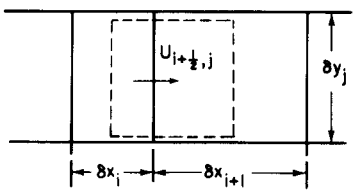

FIG. 3. Control volume (dashed rectangle) used for constructing a finite-difference approximation for the u momentum equation at location $(i + 1/2, j)$.

Thus, the zeroth order term is incorrect unless the cell widths are equal, $\delta x_i = \delta x_{i+1}$. In other words, the variable mesh reduces the order of approximation by one, and in this case leads to an incorrect zeroth order result. If a centered rather than a donor-cell approximation had been used, the result would have been first order accurate and not second order as it is in a uniform mesh.

It does not follow, however, that variable meshes are necessarily less accurate because they do allow finer zoning in localized regions where flow variables are expected to vary most rapidly. Nevertheless, variable meshes must be used with care. It is best, for example, to allow for gradual variations in cell sizes to minimize the reduction in approximation order. It is also worthwhile to look for other approximations that do not lose their accuracy in a variable mesh. In this regard, it should be noted that the reason the conservation form of the advective terms lose accuracy is that the control volumes are not centered about the positions where variables are located. Because of this the advective terms should be corrected to account for the difference in locations of the variables being updated and the centroids of their control volumes. When this is not done a lower order error is introduced.

The stability advantages of the donor-cell method can be retained in a variable mesh with no reduction in formal accuracy, if the $\mathbf{u} \cdot \nabla \mathbf{u}$ form is used for the advection flux. At the same time, it is also possible to combine the donor-cell and centered-difference approximations into a single expression with a parameter, α, that controls the relative amount of each one. The general form at $(i + \frac{1}{2}, j)$ is

$$\mathrm{FUX} = (u_{i+1/2,j}/\delta x_\alpha)[\delta x_{i+1}\,\mathrm{DUL} + \delta x_i\,\mathrm{DUR} + \alpha\,\mathrm{sgn}(u)(\delta x_{i+1}\,\mathrm{DUL} - \delta x_i\,\mathrm{DUR})], \tag{9}$$

where

$$\mathrm{DUL} = (u_{i+1/2,j} - u_{i-1/2,j})/\delta x_i,$$

$$\mathrm{DUR} = (u_{i+3/2,j} - u_{i+1/2,j})/\delta x_{i+1},$$

$$\delta x_\alpha = \delta x_{i+1} + \delta x_i + \alpha\,\mathrm{sgn}(u)(\delta x_{i+1} - \delta x_i),$$

and where $\mathrm{sgn}(u)$ means the sign of $u_{i+1/2,j}$. When $\alpha = 0$, this approximation reduces to a second order accurate, centered-difference approximation. When $\alpha = 1$, the first order donor-cell form is recovered. Thus, using the approximation defined in Eq. (9), there is no loss of formal accuracy when a variable mesh is used.

The basic idea used in Eq. (9) is to weight the upstream derivative of the quantity being fluxed more than the downstream value. The weighting factors are $1 + \alpha$ and $1 - \alpha$, for the up and downstream derivatives, respectively. The derivatives are also weighted by cell sizes in such a way that the correct order of approximation is maintained in a variable mesh. This type of approximation is used in SOLA-VOF for all convective flux terms appearing in Eq. (6). Viscous accelerations are approximated with standard centered approximations.

C. *Continuity Equation Approximation*

Velocities computed from Eq. (6) must satisfy the continuity equation, Eq. (5). In order to satisfy this equation the pressures (and velocities) must be adjusted in each computational cell occupied by fluid. This is done by an iterative process. In each cell full of fluid the pressure is changed to either draw in or force out fluid as necessary to satisfy Eq. (5). Because adjustments in one cell affect neighboring cells a number of passes through the mesh are necessary to satisfy continuity everywhere.

In cells containing a free surface a different procedure is required because the pressure is assumed specified at a surface. In this case the surface cell pressure, $p_{i,j}$, is set equal to the value obtained from a linear interpolation (or extrapolation) between the desired pressure at the surface, p_s, and a pressure, p_N, inside the fluid,

$$p_{i,j} = (1 - \eta)\, p_N + \eta p_s,$$

where $\eta = d_c/d$ is the ratio of the distance between cell centers to the distance between the free surface and the center of the neighbor cell (see Fig. 4). For this scheme to work, the adjacent cell chosen for the interpolation should be such that the line connecting its center to the center of the surface cell is closest to the normal to the free surface. The cell selected in this way is referred to as the interpolation neighbor of the surface cell.

D. *Approximations for Volume of Fluid Function*

1. *Advancing F in Time*

The VOF function F is governed by Eq. (2). For an incompressible fluid, Eq. (4) may be combined with Eq. (2) to yield the equation

$$\frac{\partial F}{\partial t} + \frac{1}{r}\frac{\partial rFu}{\partial x} + \frac{\partial Fv}{\partial y} = 0, \tag{10}$$

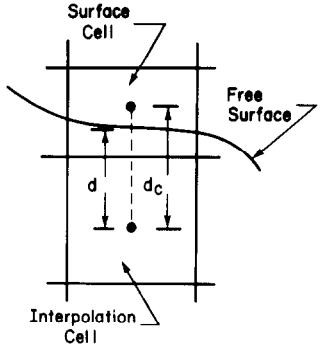

FIG. 4. Sketch showing definition of quantities used in defining free surface pressure boundary condition.

where $r = x$ when $\xi = 1$ and $r = 1$ when $\xi = 0$. Even when the fluid is slightly compressible and Eq. (5) replaces Eq. (4), this equation for F is still an acceptable approximation. Equation (10), which is in divergence form, is here more convenient for numerical approximation and is the form used in the following discussion. When Eq. (10) is integrated over a computational cell, the changes in F in a cell reduce to fluxes of F across the cell faces. As previously noted, special care must be taken in computing these fluxes to preserve the sharp definition of free surfaces. The method employed in SOLA-VOF uses a type of donor–acceptor flux approximation [8]. The essential idea is to use information about F downstream as well as upstream of a flux boundary to establish a crude interface shape, and then to use this shape in computing the flux. Several researchers have previously used variations of this approach for tracking material interfaces (see, e.g., Refs. [8, 14, 15]). The VOF method differs somewhat from its predecessors in two respects. First, it uses information about the slope of the surface to improve the fluxing algorithm. Second, the F function is used to define a surface location and orientation for the application of various kinds of boundary conditions, including surface tension forces.

The basic advection method as developed for use in the VOF technique may be understood by considering the amount of F to be fluxed through the right-hand face of a cell during a time step of duration δt. Fluxes across other cell faces are completely analogous. The total flux of fluid volume and void volume crossing the right cell face per unit cross sectional area is $V = u\,\delta t$, where u is the normal velocity at the face. The sign of u determines the donor and acceptor cells, i.e., the cells losing and gaining fluid volume, respectively. For example, if u is positive the upstream or left cell is the donor and the downstream or right cell the acceptor. The amount of F fluxed across the cell face in one time step is δF times the face cross-sectional area, where

$$\delta F = \text{MIN}\{F_{\text{AD}}\,|V| + CF, F_{\text{D}}\,\delta x_{\text{D}}\},$$

and where

$$CF = \text{MAX}\{(1.0 - F_{\text{AD}})\,|V| - (1.0 - F_{\text{D}})\,\delta x_{\text{D}}, 0.0\}. \tag{11}$$

Single subscripts denote the acceptor (A) and donor (D) cells. The double subscript, AD, refers to either A or D, depending on the orientation of the interface relative to the direction of flow as explained below.

Briefly, the MIN feature in Eq. (11) prevents the fluxing of more fluid from the donor cell than it has to give, while the MAX feature accounts for an additional fluid flux, CF, if the amount of void to be fluxed exceeds the amount available. Figure 5 provides a pictorial explanation of Eq. (11). The donor and acceptor cells are defined in Fig. 5a for fluxing across a vertical cell face. When AD = D, the flux is an ordinary donor-cell value,

$$F = F_{\text{D}}\,|V|,$$

in which the F value in the donor cell is used to define the fractional area of the cell

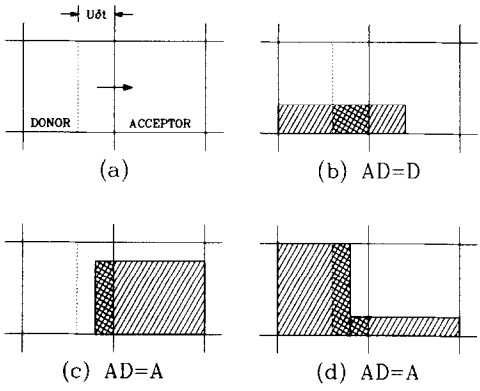

FIG. 5. Examples of free surface shapes used in the advection of F. The donor–acceptor arrangement is shown in (a), where the dashed line indicates the left boundary of the total volume being advected. The cross-hatched regions shown in (b–d) are the actual amounts of F fluxed.

face fluxing fluid; see Fig. 5b. As discussed in Section IV.F, numerical stability requires that $|V|$ be less than δx, so that it is not possible to empty the donor cell in this case.

When AD = A, the value of F in the acceptor cell is used to define the fractional area of the cell face across which fluid is flowing. In case (c) of Fig 5, all the fluid in the donor cell is fluxed because everything lying between the dashed line and the flux boundary moves into the acceptor cell. This is an example exercising the MIN test in Eq. (11). In case (d) of Fig. 5, more fluid than the amount $F_A|V|$, must be fluxed, so this is an example exercising the MAX test. In particular, the extra fluid between the dashed line and the flux boundary is equal to the CF value in Eq. (11).

Whether the acceptor or donor cell is used to determine the fractional area for fluid flow depends on the mean surface orientation. The acceptor cell is used when the surface is convected mostly normal to itself; otherwise, the donor cell value is used. However, if the acceptor cell is empty, or if the cell upstream of the donor cell is empty, then the acceptor cell F value is used to determine the flux regardless of the orientation of the surface. This means that a donor cell must fill before any fluid can enter a downstream empty cell.

The reason for testing on surface orientation is that an incorrect steepening of surface waves will occur if the acceptor cell is always used to compute fluxes. Consider, for example, a horizontal surface with a small wave moving in the positive x-direction. A flux based on the downstream (acceptor) value of F will eventually steepen the wave into a step discontinuity. In effect, the acceptor method is numerically unstable because it introduces a negative diffusion of F (i.e., a diffusion-like transport with a negative coefficient). Instabilities do not grow unbounded, however, because of the MIN and MAX tests used in the flux definition. In contrast, when the surface is advecting normal to itself, a steepening that keeps the step-function character of F is exactly what is wanted.

Once the flux has been computed by the above method, it is multiplied by the flux boundary area to get the amount of fluid to be subtracted from the donor cell and added to the acceptor cell. When the process is repeated for all cell boundaries in the mesh, the resulting F values correspond to the time-advanced values satisfying Eq. (10) and still sharply define all interfaces.

2. *Bookkeeping Adjustments*

The new F values determined by the above method may occasionally have values slightly less than zero or slightly greater than unity. Therefore, after the advection calculation has been completed, a pass is made through the mesh to reset values of F less than zero back to zero and values of F greater than one back to one. Accumulated changes in fluid volume introduced by these adjustments during a calculation are recorded and may be printed out at any time.

There is one other adjustment needed in F in order that it may be used as a surface cell flag. Surface cells have values of F lying between zero or one; however, in a numerical solution F values cannot be tested against exact numbers such as zero and one because roundoff error would cause spurious results. Instead, a cell is defined to be empty when F is less than ε_F and full when F is greater than $1-\varepsilon_F$, where ε_F is typically 10^{-6}. If, after advection, a cell has an F value less than ε_F, this F is set to zero and all neighboring full cells become surface cells by having their F values reduced from unity by an amount $1.1\varepsilon_F$. These changes in F are also included in the accumulated volume change. Volume errors after hundreds of cycles are typically observed to be a fraction of a per cent of the total fluid volume.

3. *Determining Interfaces Within a Cell.*

For the accurate application of boundary conditions, knowledge of the boundary location within a surface cell is required. In the VOF technique, it is assumed that the boundary can be approximated by a straight line cutting through the cell. By first determining the slope of this line, it can be moved across the cell to a position that intersects the known amount of fluid volume in the cell.

To determine the surface slope, it must be recognized that the surface can be represented either as a single-valued function $Y(x)$ or as $X(y)$, depending on its orientation. If the surface is representable as $Y(x)$, we must compute dY/dx. A good approximation to $Y(x)$ is

$$Y_i = Y(x_i) = F(i, j-1)\,\delta y_{j-1} + F(i,j)\,\delta y_j + F(i, j+1)\,\delta y_{j+1},$$

where $Y=0$ has been taken as the bottom edge of the $j-1$ row of cells. Then,

$$(dY/dx)_i = 2(Y_{i+1} - Y_{i-1})/(\delta x_{i+1} + 2\delta x_i + \delta x_{i-1}). \tag{12}$$

A similar calculation can be made for dX/dy,

$$X_j = X(y_j) = F(i-1, j)\,\delta x_{i-1} + F(i,j)\,\delta x_i + F(i+1, j)\,\delta x_{i+1},$$

and

$$(dX/dy)_j = 2(X_{j+1} - X_{j-1})/(\delta y_{j+1} + 2\,\delta y_j + \delta y_{j-1}). \tag{13}$$

If $|dY/dx|$ is smaller than $|dX/dy|$, the surface is more nearly horizontal than vertical, otherwise it is more nearly vertical. In any case, the derivative with the smallest magnitude gives the best approximation to the slope because the corresponding Y or X approximation is most accurate in that case.

Suppose $|dY/dx|$ is smallest so the interface is more horizontal than vertical. If dX/dy is negative, fluid lies below the surface, and cell $(i, j-1)$ is used as the interpolation neighbor for surface cell (i, j). Had dX/dy been positive, cell $(i, j+1)$ would be chosen for the neighboring interpolation cell because fluid would then be above the surface.

Once the surface slope and the side occupied by fluid have been determined, a line can be constructed in the cell with the correct amount of fluid volume lying on the fluid side. This line is used as an approximation to the actual surface and provides the information necessary to calculate η for the application of free surface pressure boundary conditions, as described in Section IV.C.

For cylindrical coordinates, the above computations are more complex because of the volume dependence on radius. Except for cells on the axis, however, the exact results differ little from the simpler Cartesian coordinate results; consequently, the latter are used in both cases.

Surface tension effects may be included in SOLA-VOF with little additional effort [16]. The essential step is to compute a local curvature in each surface cell using the $Y(x)$ or $X(y)$ definitions, Eqs. (12)–(13), and from this an effective surface tension pressure, p_s, to be applied at the surface.

E. *Boundary Conditions*

1. *Mesh Boundaries*

In addition to the free surface boundary conditions, it is necessary to set conditions at all mesh boundaries and at surfaces of all internal obstacles. At the mesh boundaries, a variety of conditions may be set using the layer of fictitious cells surrounding the mesh. The basic idea is to set values for the dependent variables in the fictitious cells such that the desired boundary conditions are met at the boundaries. Using this technique the SOLA-VOF program has provisions for rigid-free or no-slip walls, for continuative outflow boundaries, for periodic boundaries, and for specified pressure boundaries. These conditions, as well as those to be set at free surfaces, are set in the same manner as in all previous SOLA codes [2].

F. *Numerical Stability Considerations*

Numerical calculations often have computed quantities that develop large, high frequency oscillations in space, time, or both. This behavior is usually referred to as a

numerical instability, especially if the physical problem being studied is known not to have unstable solutions. When the physical problem does have unstable solutions and if the calculated results exhibit significant variations over distances comparable to a cell width or over times comparable to the time increment, the accuracy of the results cannot be relied on. To prevent this type of numerical instability or inaccuracy, certain restrictions must be observed in defining the mesh increments δx_i and δy_j, the time increment δt, and the upstream differencing parameter α.

For accuracy, the mesh increments must be chosen small enough to resolve the expected spatial variations in all dependent variables. When this is impossible because of limitations imposed by computing time or memory requirements, special care must be exercised in interpreting calculational results. For example, in computing the flow in a large chamber it is usually impossible to resolve thin boundary layers along the confining walls. In many applications, however, the presence of thin boundary layers is unimportant and free-slip boundary conditions can be justified as a good approximation.

Once a mesh has been chosen, the choice of the time increment necessary for stability is governed by two restrictions. First, material cannot move through more than one cell in one time step because the difference equations assume fluxes only between adjacent cells. Therefore, the time increment must satisfy the inequality

$$\delta t < \min \left\{ \frac{\delta x_i}{|u_{i,j}|}, \frac{\delta y_j}{|v_{i,j}|} \right\},$$

where the minimum is with respect to every cell in the mesh. Typically, δt is chosen equal to one-fourth to one-third of the minimum cell transit time. Second, when a nonzero value of kinematic viscosity is used, momentum must not diffuse more than approximately one cell in one time step. A linear stability analysis shows that this limitation implies

$$\nu \, \delta t < \frac{1}{2} \frac{\delta x_i^2 \, \delta y_j^2}{\delta x_i^2 + \delta y_j^2}.$$

With δt chosen to satisfy the above two inequalities, the last parameter needed to ensure numerical stability is α. The proper choice for α is

$$1 \geqslant \alpha > \max \left\{ \left| \frac{u_{i,j} \, \delta t}{\delta x_i} \right|, \left| \frac{v_{i,j} \, \delta t}{\delta y_j} \right| \right\}.$$

As a rule of thumb, an α approximately 1.2 to 1.5 times larger than the right-hand member of the last inequality is a good choice. If α is too large an unnecessary amount of numerical smoothing (diffusion-like truncation errors) may be introduced [17].

V. Sample Problems

Six calculational examples have been chosen to illustrate the accuracy and capabilities of the SOLA-VOF code. In all these examples, either experimental or analytical information is available for comparison with the calculated results. These examples offer a substantial challenge to any free boundary method.

A. *Broken Dam Problem*

In this example, a rectangular column of water, in hydrostatic equilibrium, is confined between two vertical walls, Fig. 6. The water column is 1.0 units wide and 2.0 units high. Gravity is acting downward with unit magnitude. At the beginning of the calculation, the right wall (dam) is removed and water is allowed to flow out along a dry horizontal floor. Experimental results for this problem have been reported [18] for the position vs time of the leading edge of the water as it flows to the right (Fig. 7).

This is a good test problem because it has simple boundary conditions and a simple initial configuration. The appearance of both a vertical and horizontal free surface, however, provides a check on the capability of SOLA-VOF to treat free surfaces that are not single valued with respect to x or y. Results from two calculations are presented in Fig. 7 with the experimental data. In both cases, the mesh consisted of 40 uniformly spaced columns ($\delta x = 0.1$) and 22 nonuniformly spaced rows. The smallest δy values are located at the bottom of the mesh, where resolution is needed to define the thin leading edge of the advancing water. In the first calculation, the smallest δy was 0.05, while in the second it was 0.025. We see from

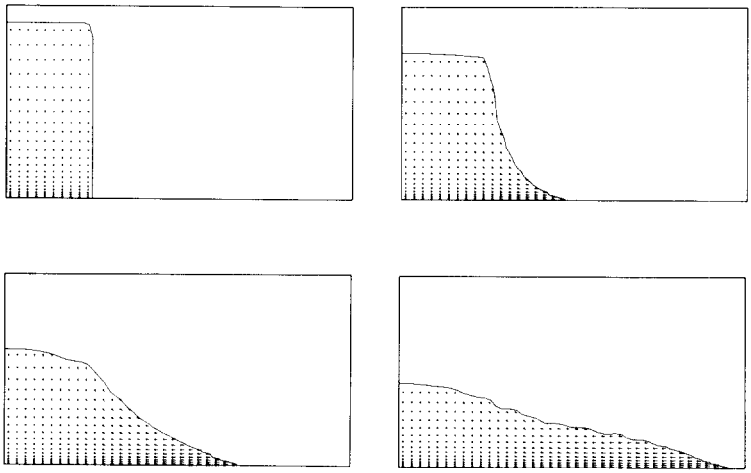

FIG. 6. Velocity vectors and fluid configurations for broken dam problem at times 0.0, 0.9, 1.4, and 2.0. Vectors are drawn from cell centers, which are marked by + signs. The free surface is drawn as an $F = \frac{1}{2}$ contour line, which is why the top right corner at $t = 0.0$ is not 90°.

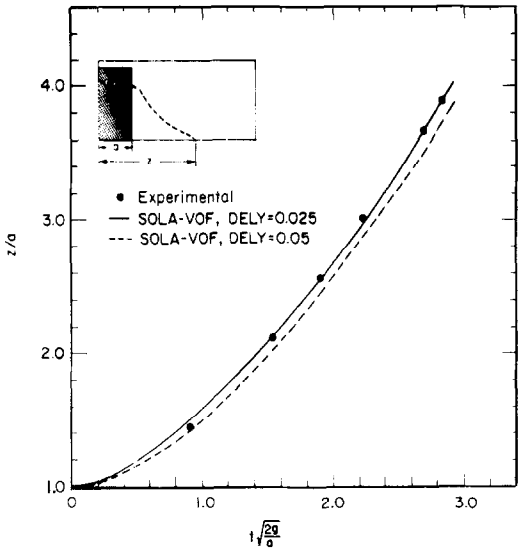

FIG. 7. Comparison of calculated results with experimental data for the broken dam problem.

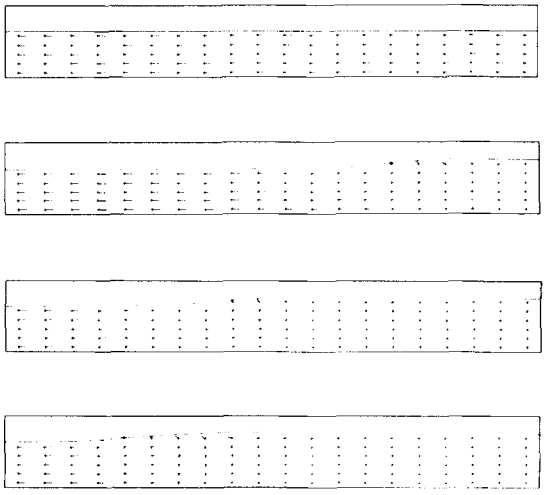

FIG. 8. Velocity vectors and fluid configuration for undular bore problem at times 0.0, 4.05, 7.02, and 10.08.

Fig. 7 that the best results are obtained with the smallest δy case, but both results are still quite good. The greatest deviation from the experimental results is everywhere less than one cell width.

The smallest δy calculation required 460 time cycles to get the water to the right wall ($x = 4.0$) and used 328 sec of CDC-7600 computer time (which included a considerable amount of numerical and graphical output).

B. *Undular Bore*

If a horizontal layer of water is pushed into a rigid, vertical wall there will be a step wave, or bore, produced that runs away from the wall. If the incident velocity is not too great, the bore front will have a well-behaved, undular shape, but at sufficiently high velocities the bore front will break and be highly irregular. In either case, conservation of mass and momentum principles may be used to derive "jump" conditions that should exist across the bore transition [19].

SOLA-VOF was used to compute the undular bore evolution shown in Fig. 8. The initial configuration in Fig. 8a consists of a uniform mesh of 20 cells in the horizontal direction ($\delta x = 0.6$) and 8 cells in the vertical direction ($\delta y = 0.2$). Fluid initially fills the lowest five rows (depth 1.0) and is uniformly moving to the right with unit velocity. The right, bottom, and top walls are rigid, free-slip boundaries. At the left boundary, fluid is continuously input to prevent any waves from being generated there. Gravity acts downward with unit magnitude.

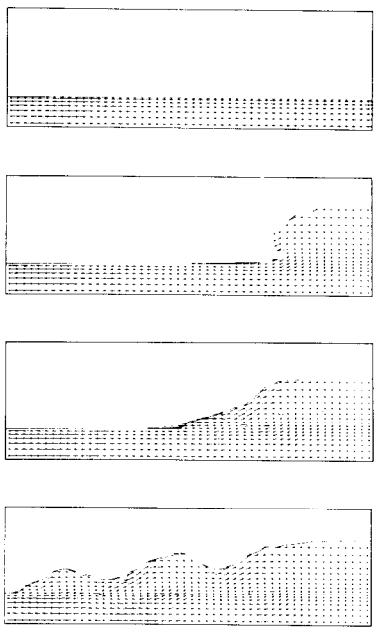

FIG. 9. Velociy vectors and fluid configuration for breaking bore problem at times 0.0, 6.50, 8.51, and 14.01.

Although this problem is very coarsely resolved, the results are remarkably good and provide a nice check on mass and momentum conservation. The computed jump height at the right wall is 1.201, while theory predicts 1.209. A more finely resolved calculation using a mesh consisting of 60 by 12 cells yielded a height of 1.203, which is converging to the theoretical answer.

The coarse mesh calculation took 14 sec of computer time to complete 48 cycles of calculation.

C. *Breaking Bore*

A more interesting example is produced by decreasing the gravitational acceleration in the above example from unity to 0.4548. In this case, the bore transition is turbulent and involves a water elevation change from 1.0 to 2.8. Fluid configurations and velocity fields at selected times, showing the development of the bore, are shown in Fig. 9. In this case the computational mesh consisted of 50 equally spaced cells in the x-direction ($\delta x = 0.25$) and 20 cells with variable spacing in the y-direction. The variable spacing was chosen to give finer resolution around $y = 1.0$, where a shear layer is formed as the incoming water flows into the bore front.

Experimental evidence indicates that turbulent bore transitions have widths that are typically equal to about five times the change in elevation ($2.8 - 1.0 = 1.8$). This is consistent with the calculational results, even though the calculation is not computing true turbulence. A better measure of the accuracy of the calculation is the final height at the right wall, which is 2.91 and is in good agreement with the theoretical value, 2.8.

No special considerations were needed to maintain the resolution of the free surface as it continually folds over on itself, the VOF technique handles this automatically. This calculation required 292 sec of CDC-7600 computer time for 457 cycles of computation.

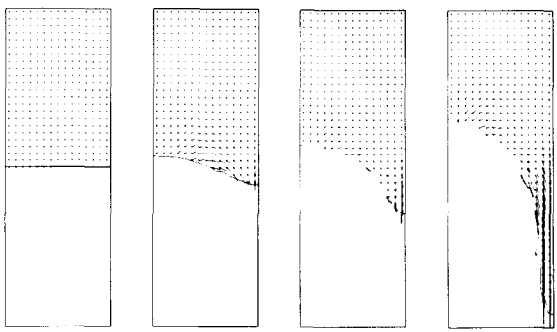

FIG. 10. Evolution of a Rayleigh–Taylor instability started by a pressure perturbation. Times are 0.0, 0.4, 0.8, and 1.6.

D. *Rayleigh–Taylor Instability*

Because the success of the VOF technique is based on the ability to numerically advect a step-function distribution (F) without numerical smoothing, it is worthwhile to investigate the sensitivity of SOLA-VOF to changes in the F-advection algorithm. A good problem for this purpose is the nonlinear development of a Rayleigh–Taylor instability. During the early stages of the instability the fluid surface moves normal to itself, but during the later stages there are regions along the sides of the growing liquid fingers where the flow is mostly tangential to the surface. Thus, this problem offers a good test of the particular combination of donor- and acceptor-cell fluxing used in the code.

The initial fluid configuration consists of an inviscid fluid occupying the top half of a box that has a width of 1.0 and height of 3.0. Gravity is acting downward with unit magnitude. The free surface is given an applied pressure pulse, $p_s = \cos(\pi x)$, that acts only during the first cycle of calculation. This pulse perturbs the unstable fluid surface, causing it to flow down along the right edge of the box in the form of a fluid spike, while a bubble moves up along the left box edge; see Fig. 10. During the earliest stages of growth, the amplitudes of the bubble and spike displacements follow linear theory [20], but nonlinear effects quickly take over with the spike growing significantly more rapidly than the bubble.

To check the sensitivity of the F-advection algorithm used in SOLA-VOF this problem was repeated with F advective fluxes determined entirely by the downstream or acceptor cell F values. This pure acceptor-cell method, which differs from the mixture of donor–acceptor fluxing used in the SOLA-VOF code, has been used in some previous work (see, e.g., Ref. [14]). The consequences of using pure acceptor-cell fluxing are obvious from a comparison of Fig. 11 with Fig. 10. The acceptor-cell method develops large irregularities in the free surface, particularly where it is flowing parallel to itself. This does not occur in the SOLA-VOF method because it uses donor cell fluxing in such regions.

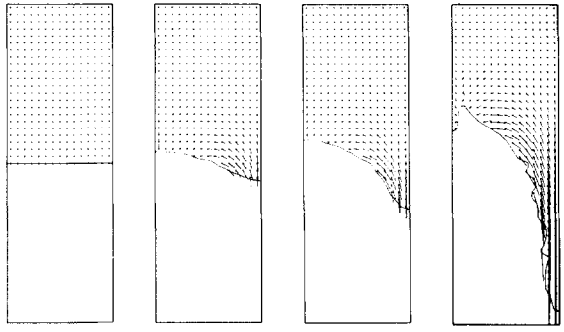

FIG. 11. Repeat of calculation shown in Fig. 10 using pure acceptor cell advection for F. Note the considerably more irregular surface in the last frame.

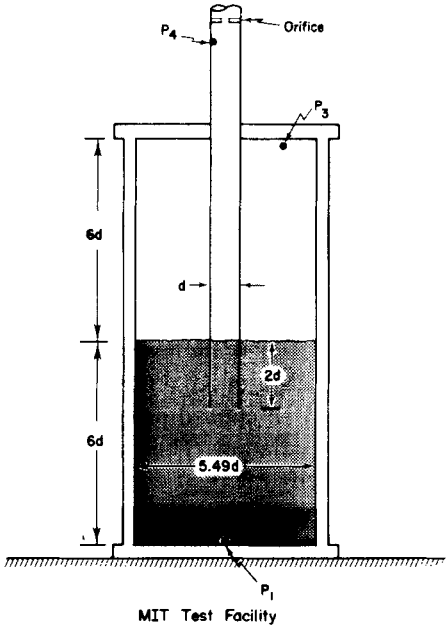

FIG. 12. Schematic of MIT single-vent test apparatus.

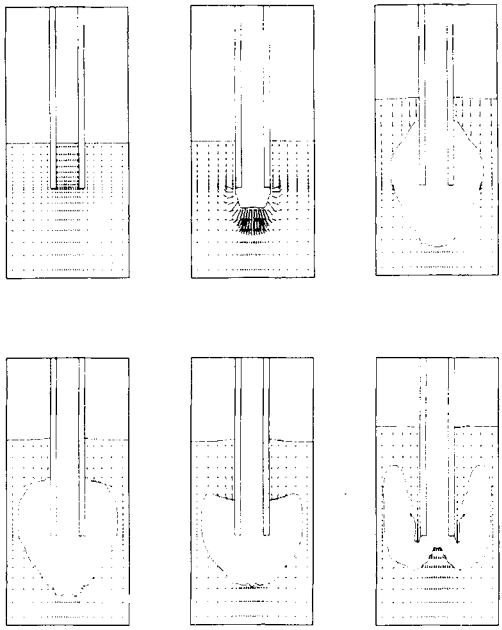

FIG. 13. Velocity vectors and free surface configurations computed when air is forced through submerged vent pipe.

From this simple example, it is evident that the particular combination of donor–acceptor advection used in the VOF technique does an exceedingly good job. It is all the more remarkable because the algorithm uses a single pass through the mesh with relatively few calculations required for the flux at each cell boundary.

E. *A Reactor Safety Application*

Many boiling water reactors use a large pool of water to condense stream should a major steam leak occur. In some designs, steam would be forced into the pool through long vertical pipes extending several pipe diameters below the surface of the pool. Before steam enters the pool, however, air initially in the pipes must be pushed out. The ejection of this noncondensable air forms large bubbles in the pool and displaces the pool surface upward. Safety considerations require an understanding of the hydrodynamic forces generated during this process. For this purpose, several small-scale experimental programs have been conducted and several groups have attempted supporting theoretical analysis.

A cross section of a single-pipe apparatus used at the Massachusetts Institute of Technology [21] is shown in Fig. 12. It consists of a cylindrical vessel approximately half filled with water and with an axisymmetric pipe extending down into the pool from above. At the beginning of a test, a valve is opened at the top end of the central pipe exposing it to a constant pressure plenum. Gas in the plenum flows through an orifice in the pipe and then into the lower pressure cylindrical tank by displacing water initially in the pipe.

To model this test apparatus with the SOLA-VOF code, it is necessary to supplement the code with calculations for the gas pressure in the pipe and for the pressure in the space above the pool surface. These pressures are then used as free surface boundary pressures. A sequence of calculated results illustrating the fluid dynamics associated with the air clearing process are contained in Fig. 13. The free boundaries obviously undergo severe distortion, but the SOLA-VOF algorithm has no

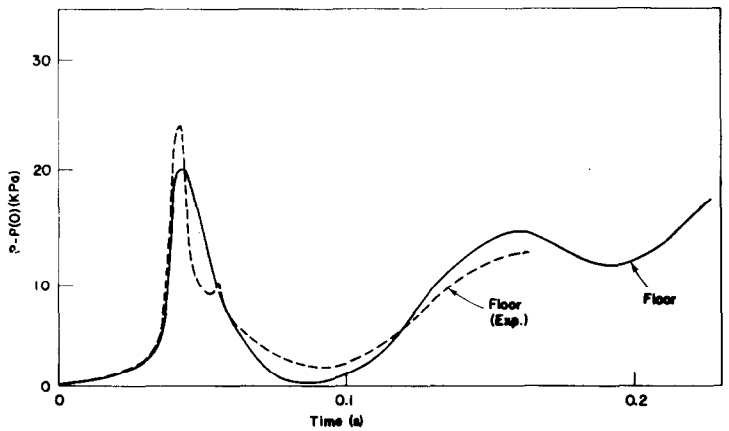

FIG. 14. Comparison of calculated and measured pressure history on floor of pool chamber.

difficulty in following the fluid motion. Pressures measured at the center of the floor are compared with the corresponding calculated pressures in Fig. 14. The agreement is reasonably good, except for some of the details associated with the initial pressure spike. There is some experimental evidence that the higher first spike and subsequent small second spike are results of elastic flexibility in the apparatus, which was not included in the calculation. Similar results have also been obtained for many other test conditions and for other measured quantities [22]. Since these results have been reported in detail in the quoted references, they are not reproduced here. Neverthesless, these results serve to further validate the SOLA-VOF code as a powerful and useful research tool.

VI Summary

The volume of fluid (VOF) technique has been presented as a simple and efficient means for numerically treating free boundaries embedded in a calculational mesh of Eulerian or Arbitrary Lagrangian–Eulerian cells. It is particularly useful because it uses a minimum of stored information, treats intersecting free boundaries automatically, and can be readily extended to three-dimensional calculations.

The VOF technique was described in detail as it has been used to follow free surfaces in an incompressible hydrodynamics code. Sample calculations with the new code, SOLA-VOF, show that it works extremely well for a wide range of complicated problems.

Acknowledgments

The authors wish to thank R. S. Hotchkiss for numerous useful discussions and for his efforts in adding a surface tension capability to the SOLA-VOF program. This work was supported by the Electric Power Research Institute, Contract RP-965-3, and in part by the Office of Naval Research, Contract NA-onr-6-75, NR 062-455, with the cooperation of the U. S. Department of Energy.

References

1. C. W. Hirt, A. A. Amsden, and J. L. Cook, *J. Comput. Phys.* **14** (1974), 227.
2. C. W. Hirt, B. D. Nichols, and N. C. Romero, "SOLA—A Numerical Solution Algorithm for Transient Fluid Flows," Los Alamos Scientific Laboratory report LA-5852, 1975.
3. B. D. Nichols and C. W. Hirt, *J. Comput. Phys.* **12** (1973), 234.
4. B. D. Nichols and C. W. Hirt, *J. Comput. Phys.* **8** (1971), 434.
5. B. D. Nichols and C. W. Hirt, "Proceedings First Intern. Conf. Num. Ship Hydrodynamics, Gaithersburg, Md, October 1975."
6. F. H. Harlow and J. E. Welch, *Phys. Fluids* **8** (1965), 2182; J. E. Welch, F. H. Harlow, J. P. Shannon, and B. J. Daly, "The MAC Method: A Computing Technique for Solving Viscous, Incompressible, Transient Fluid Flow Problems Involving Free Surfaces," Los Alamos Scientific Laboratory report LA-3425, 1966.
7. F. H. Harlow, A. A. Amsden, and J. R. Nix, *J. Comput. Phys.* **20** (1976), 119.

8. W. E. JOHNSON, "Development and Application of Computer Programs Related to Hypervelocity Impact," Systems Science, and Software report 3SR-353, 1970.
9. R. K.-C. CHAN AND R. L. STREET, *J. Comput. Phys.* **6** (1970), 68.
10. SOLA codes may be obtained from the National Energy Software Center, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.
11. W. H. MCMASTER AND E. Y. GONG, "PELE-IC User's Manual," Lawrence Livermore Laboratory report UCRL-52609, 1979.
12. W. H. MCMASTER, private communication.
13. C. W. HIRT AND B. D. NICHOLS, *J. Comput. Phys* **34** (1980), 390.
14. J. D. KERSHNER AND C. L. MADER, "2DE: A Two-Dimensional Continuous Eulerian Hydrodynamic Code for Computing Multicomponent Reactive Hydrodynamic Problems," Los Alamos Scientific Laboratory report LA-4846, 1972.
15. W. F. NOH AND P. WOODWARD, "The SLIC (Simple Line Interface Calculation) Method," Lawrence Livermore Laboratory report UCRL-52111, 1976.
16. B. D. NICHOLS, C. W. HIRT, AND R. S. HOTCHKISS, "Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries," Los Alamos Scientific Laboratory report, LA–8355, 1980.
17. C. W. HIRT, *J. Comput. Phys.* **2** (1968), 339.
18. J. C. MARTIN AND W. J. MOYCE, *Philos. Trans. Roy. Soc. London Ser. A* **244** (1952), 312.
19. J. J. STOKER, "Water Waves," Interscience, New York, 1957.
20. F. H. HARLOW AND J. E. WELCH, *Phys. Fluids* **9** (1966), 842.
21. W. G. ANDERSON, P. W. HUBER, AND A. A. SONIN, "Small Scale Modeling of Hydrodynamic Forces in Pressure Suppression Systems, Final Report," Dept. Mech. Eng., Massachusetts Institute of Technology, Nuclear Regulatory Commission report NUREG/CR-0003, 1978.
22. B. D. NICHOLS AND C. W. HIRT, *Nucl. Sci. Eng.* **73** (1980), 196.