201
2/5/80

LA-8163-MS
Informal Report

MASTER

Lib. 644

Simulation of Tank Draining Phenomena with the NASA SOLA-VOF Code

University of California

LOS ALAMOS SCIENTIFIC LABORATORY
Post Office Box 1663 Los Alamos, New Mexico 87545

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

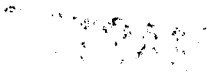
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

An Affirmative Action/Equal Opportunity Employer



This report was not edited by the Technical Information staff.

This work was supported by the Lewis Research Center of the National Aeronautics and Space Administration (NASA).

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

UNITED STATES
DEPARTMENT OF ENERGY
CONTRACT W-7405-ENG. 36

LA-8163-MS
Informal Report
UC-32
Issued: December 1979

MASTER

Simulation of Tank Draining Phenomena with the NASA SOLA-VOF Code

R. S. Hotchkiss

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

CONTENTS

SIMULATION OF TANK DRAINING PHENOMENA WITH THE NASA SOLA-VOF CODE

by

R. S. Hotchkiss

ABSTRACT

The NASA SOLA-VOF code is a modified version of the SOLA-VOF computer program, specifically designed to calculate the fluid dynamics involved in baffled and unbaffled tank draining problems. It solves the time dependent finite-difference equations that govern the two-dimensional motions of fluids with a free surface upon which surface tension forces can act. The VOF method of tracking the free surface provides an algorithm by which multivalued free surface calculations with surface tension are easily performed.

Calculations can be made in either planar or cylindrical geometries with a variety of boundary conditions. The surface tension boundary condition is modeled by an applied surface pressure and wall adhesion effects are specified by a wall contact angle.

Complete descriptions are given of the code structure, of procedures for running and setting up the code, of the variables used and of test problems that show the excellent agreement between the calculations and experiments of tank draining problems. A complete computer listing is included in an appendix.

I. INTRODUCTION

Statement of the Problem

The free surface dynamics of many common fluids in normal gravity environments are relatively insensitive to surface tension effects. However, in low gravity applications or in fluids possessing very large surface tension coefficients, the forces induced by surface tension become important influences upon the surface motion.

The SOLA-VOF technique, developed by Hirt and Nichols,[1] was initially designed to provide a relatively simple, yet powerful, means of computing flows of fluids with multivalued free surfaces or two materials separated by a multivalued interface in two dimensions. The SOLA-VOF code, however, has been expanded to include the effects of surface tension in either of these types of flows.[2]

The surface tension model was developed initially for application to the draining problems of liquid propellants from tanks in space vehicles. This problem involves the draining of various liquid propellants from a hemispherically bottomed cylindrical tank in very low gravity environments; environments in which surface tension may significantly influence the flow. In fact, in low gravity environments the surface tension forces are so influential on the draining characteristics of a tank that a residual amount of fluid can be trapped in the tank as a result of surface tension alone.[3,4] Studies of these flows are therefore important in order to comprehend the extent of the residual volumes of propellant in the fuel tanks and the amount of additional payload to a space vehicle that the residual volume implies. Further, studies of mechanisms to reduce the residual volumes assist the design of propellant tanks that are capable of reducing the additional payload or eliminating it entirely.[5]

For this purpose, the NASA SOLA-VOF code has been designed. It is a general purpose, one-material time-dependent computer code capable of solving the equations of motion for incompressible, viscous fluids with free surfaces upon which surface tension forces exist. These flows can exist in the presence of obstacles and variable geometries such as the hemispherical section of the tank.

Although the code can be used to calculate a wide variety of problems, NASA SOLA-VOF contains modifications that allow its use to study flows in a hemispherically bottomed cylindrical tank of radius $R = 1$ with an outlet of radius $r = 0.1$. (Note: these units are nondimensional and may be dimensionally scaled to any size tank having an outlet to tank radius ratio of 0.1. All variables used in this report are nondimensional unless otherwise specified. Section VIII describes the scaling procedures to dimensionalize the results.) The code can be used to study flows characterized by Weber numbers (We) in the range $0.001 < We < \infty$ and Bond numbers (Bo) in the range $0 < Bo < \infty$. The ranges given here are those that can be accomplished in a reasonable amount of computer time; otherwise, there is no real lower limit to We. The definition of We used here is $0.0001/\sigma$ based on the tank dimensions with σ being the surface tension per unit length. Likewise, Bo is defined as g/σ where g is the acceleration of the environment. These flows can also be computed in the presence of a disk type baffle located in the tank above the outlet as desired.

The solution from the code provides a history of free surface motion in addition to a final value of residual volume. The code further automatically provides a graphical interpretation of this history in addition to the detailed numerical results at any desired instant of time.

II. METHODOLOGY

The formulation of finite-difference equations, boundary conditions, and stability criteria of the basic method are thoroughly discussed in App. A and will not be further discussed here. The reader is encouraged to study this appendix to gain a thorough understanding of the method.

The basis of the SOLA-VOF technique is the solution of the Navier-Stokes equations, the incompressible mass equation, and a transient transport equation for the convection of F, the fractional volume of fluid on a variably spaced mesh of cells spanning the region of interest. These equations are advanced through time in discrete time steps to provide the transient evolution of the flow field.

The major difference between the VOF method reported in App. A and the NASA SOLA-VOF code is the addition of surface tension along the free surface and the effects of wall adhesion. In this method surface tension is modeled as a pressure applied at the free surface. This surface pressure P_s is a function of the surface tension force per unit length σ, and the local curvature at the point of application and is defined by

$$P_s = -\sigma K \tag{1}$$

in which K is the curvature of the surface and is given by

$$K = K_{xy} + K_{cyl} = \frac{1}{R_{xy}} + \frac{1}{R_{cyl}} \tag{2}$$

in which R_{xy} is the principal radius of curvature in the x-y plane and R_{cyl} is the principal radius of curvature in the azimuthal direction. $R_{cyl} = \infty$ in planar coordinates . Given a surface function G = f(x) or G = f(y) the planar curvature is given by †

$$K_{xy} = \frac{G''}{[1 + (G')^2]^{3/2}} \tag{3}$$

---

†The customary absolute value sign has been omitted from the numerator because the sign of G'' determines the proper sign of the curvature K_{xy}.

in which the primes denote differentiation with respect to the independent variable. If G denotes a single-valued surface height function dependent only on x (i.e., nearly horizontal), Eq. (3) would give K_{xy} . If G, on the other hand, was given as G = f(y), a single-valued height function of y alone (i.e., a nearly vertical surface), then (3) would also give K_{xy}. As already noted, the VOF method can accommodate multivalued surfaces, thus in the computing mesh the surface height function is really given by G = f(x,y) and Eq. (3) does not globally apply. Fortunately, however, a local evaluation of G (i.e., in a given surface cell) can be made on the basis of the local single valuedness of G, i.e., locally either G = f(x) or G = f(y). The method by which this is discerned is the essence of surface tension in the NASA SOLA-VOF code. Figure 1a shows a locally near horizontal surface (G = f(x)) passing through surface cell (i,j). The curvature in cell (i,j) can be computed from (3) by evaluating G at the three points indicated. G is evaluated by summing the height of fluid at the center and adjacent columns. This procedure requires no knowledge of whether the fluid is above or below the surface, although in the example shown it is assumed to be below the surface and the heights are evaluated relative to the bottom of the j-1 row of cells. The procedure is†

$$G_{i+1} = AVFR^{*} = \sum_{\ell=j-1}^{j+1} F_{i+1,\ell}\ \delta y_{\ell}$$

$$G_{i} = AVFCX^{*} = \sum_{\ell=j-1}^{j+1} F_{i,\ell}\ \delta y_{\ell}$$

$$G_{i-1} = AVFL^{*} = \sum_{\ell=j-1}^{j+1} F_{i-1,\ell}\ \delta y_{\ell} \qquad (4)$$

†The superscript * appearing with a capitalized name denotes the FORTRAN name associated with such a variable in the code.

then $G'' = GPP^* = \left[\frac{(AVFR - AVFCX)}{1/2(\delta x_i + \delta x_{i+1})} - \frac{(AVFCX - AVFL)}{1/2(\delta x_i + \delta x_{i-1})} \right] \cdot \frac{1}{1/2 \langle \delta x_i \rangle}$

where $\langle \delta x_i \rangle = 1/2\ \delta x_{i-1} + \delta x_i + 1/2\ \delta x_{i+1}$,

and $G' = GP^* = \frac{1}{2} \left[\frac{(AVFR - AVFCX)}{1/2(\delta x_i + \delta x_{i+1})} + \frac{(AVFCX - AVFL)}{1/2(\delta x_i + \delta x_{i-1})} \right]$ (5)

and (3) can be evaluated.

A similar evaluation is made for near vertical surfaces (G = f(y)) in the vicinity of cell (i,j) as shown in Fig. 1b. In this case, fluid heights are summed along rows in a manner completely analogous to that above. This procedure is

$$G_{j+1} = AVFT^* = \sum_{\ell=i-1}^{i+1} F_{\ell,j+1}\ \delta x_\ell$$

$$G_j = AVFCY^* = \sum_{\ell=i-1}^{i+1} F_{\ell,j}\ \delta x_\ell$$

$$G_{j-1} = AVFB^* = \sum_{\ell=i-1}^{i+1} F_{\ell,j-1}\ \delta x_\ell \qquad (6)$$

now

$$G'' = GPP^* = \left[\frac{(AVFT - AVFCY)}{1/2(\delta y_j + \delta y_{j+1})} - \frac{(AVFCY - AVFB)}{1/2(\delta y_j + \delta y_{j-1})} \right] \cdot \frac{1}{1/2 \langle \delta y_j \rangle}$$

where $\langle \delta y_j \rangle = 1/2\ \delta y_{j-1} + \delta y_j + 1/2\ \delta y_{j+1}$,

and

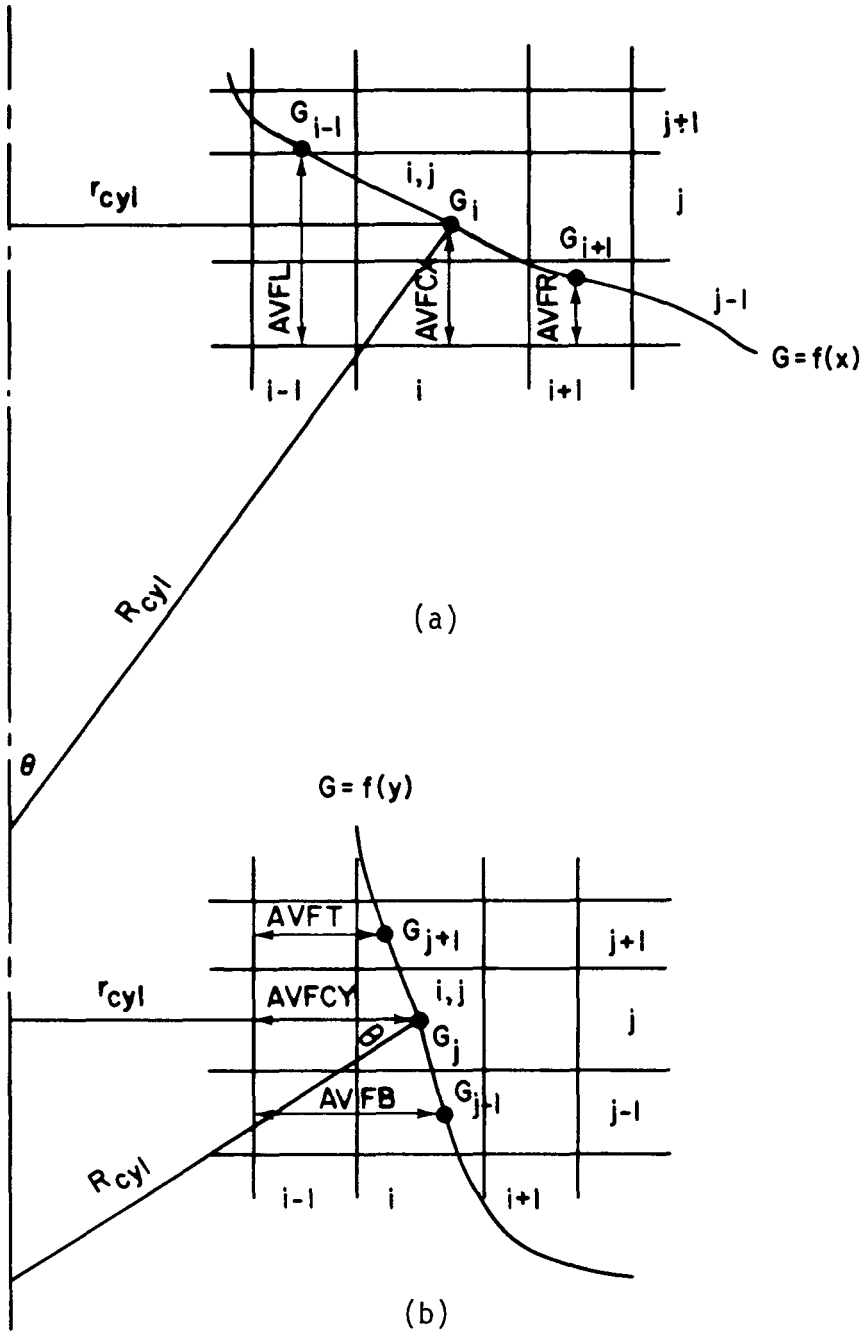


Fig. 1. (a) The near horizontal surface requires that columns be locally summed to provide the surface heights necessary to determine the planar curvature. The cylindrical radius of curvature is also shown. (b) The vertical surface requires that rows be locally summed to provide the surface heights necessary to determine the planar curvature. The cylindrical radius of curvature is also shown.

$$G' = GP^* = \frac{1}{2}\left[\frac{(AVFT - AVFCY)}{1/2(\delta y_j + \delta y_{j+1})} + \frac{(AVFCY - AVFB)}{1/2(\delta y_j + \delta y_{j-1})}\right] \tag{7}$$

for the evaluation of (3).

The problem with these two procedures is that when one is computing in cell (i,j), how does one know which of the two to use. In other words, how does one know if the fluid surface is more nearly horizontal or more nearly vertical. This question is readily answered by partially doing both procedures. From (4), the surface slope at cell (i,j) with respect to the horizontal can be made, namely

$$\frac{dG}{dx} = PFX^* = \frac{(AVFR - AVFL)}{1/2\delta x_{i-1} + \delta x_i + 1/2\delta x_{i+1}} \quad . \tag{8}$$

From (6), the surface slope at cell (i,j) with respect to the vertical is similarly,

$$\frac{dG}{dy} = PFY^* = \frac{(AVFT - AVFB)}{1/2\delta y_{j-1} + \delta y_j + 1/2\delta y_{j+1}} \quad . \tag{9}$$

A comparison of (8) with (9) reveals the answer to the question posed. If |PFY| > |PFX| the surface is more horizontal than vertical and (5) is used. Similarly if |PFX| > |PFY|, the surface is more vertical than horizontal and (7) is used. Equation (3) can now be fully evaluated, and in turn can be used to solve for the surface pressure P_s in cell (i,j). The value of G'' in (3), determined by (5) or (7) provides the proper sign for K_{xy} to be used in (1). Thus, fluid inside a locally convex surface with $G'' < 0$ will be influenced by a positive surface pressure; fluid inside a locally concave surface with $G'' > 0$ will experience a negative surface pressure.

The only remaining variable to be evaluated to fully define P_s in (1) is the K_{cyl} term in (2). The orientation of the surface in cell (i,j) is now known and it is a simple matter to determine upon which side of the horizontal or vertical surface the fluid is located. If the surface is nearly horizontal and PFY > 0,

fluid is above the surface; if PFY < 0, fluid is below the surface. Similarly for a near vertical surface, PFX > 0 implies that the fluid is to the right and PFX < 0 implies that the fluid is to the left of the surface. Further, for a near horizontal surface, PFX = tan θ where θ is the angle the surface makes with the horizontal. Likewise for near vertical surfaces PFY = tan θ, only now θ is the angle between the surface and the vertical. Both of these situations are depicted in Figs. 1a and 1b. To solve for the cylindrical component of curvature in the near horizontal case $\left(\text{i.e., } K_{cyl} = \frac{1}{R_{cyl}} = \frac{\sin\theta}{r_{cyl}}\right)$ is straightforward since r_{cyl} is given by XI_i which is the distance in the x direction from the axis to the center of cell (i,j). In this case the sign of the curvature depends upon PFX; if PFX < 0, as shown, K_{cyl} is positive (and implies a negative surface pressure), and vice versa. For the near vertical case, $K_{cyl} = \frac{1}{R_{cyl}} = \frac{\cos\theta}{r_{cyl}}$. Now r_{cyl} depends upon which side of the surface the fluid is located. If fluid is to the left, $r_{cyl} = x_{i-\frac{1}{2}} + F_{ij}\,\delta x_i$; if fluid is to the right $r_{cyl} = x_{i+\frac{1}{2}} - F_{ij}\,\delta x_i$. The sign of K_{cyl} is again determined by PFX the same as before.

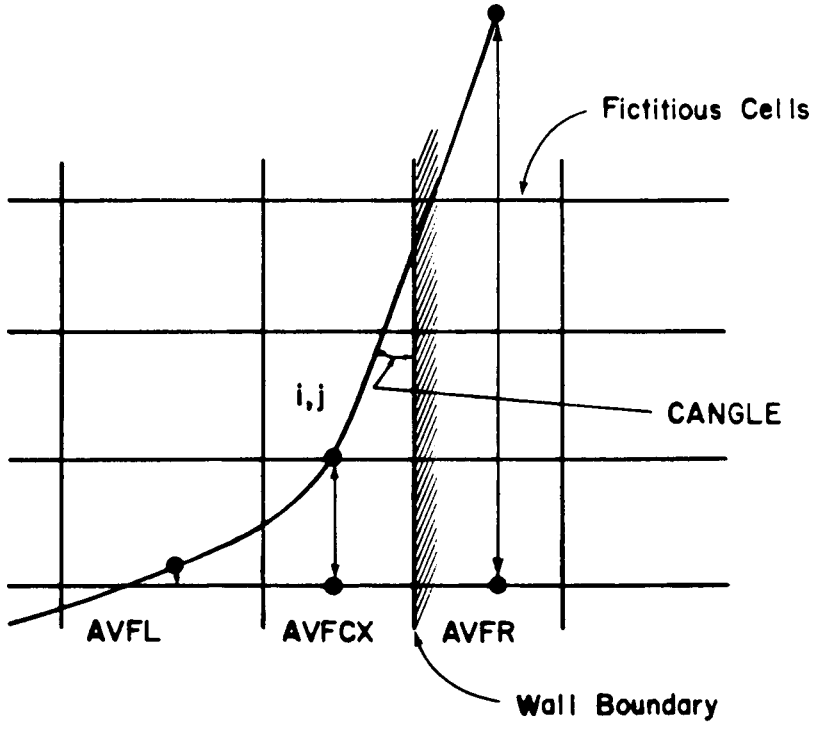
With these quantities determined, the curvature K in (2) can be evaluated and used in (1) to determine the surface pressure which is used in the code as an applied pressure in the surface cell in which it is computed.

Wall adhesion effects are modeled simply with the contact angle CANGLE. As the surface slope is being determined at a boundary from (4) or (6), the value of AVFT, AVFB, AVFR or AVFL is set consistent with CANGLE. For example, suppose the surface cell at the boundary is considered as shown in Figs. 2a and 2b with fluid assumed to be below the surface. A surface pressure is desired for this cell and will be calculated in the same manner as previously outlined except in this case, the surface must make an angle CANGLE, with the wall. Equations (4) and (6) are computed as usual except the values of AVFR and AVFT are set as follows:

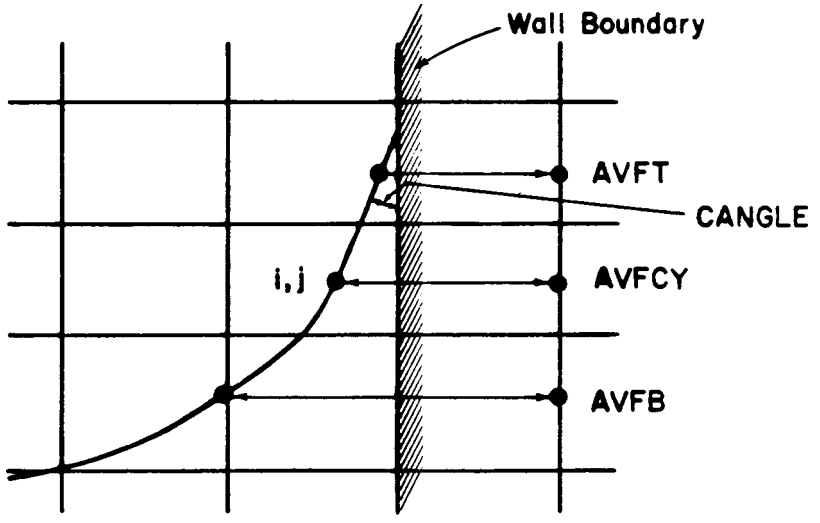
$$AVFR = AVFCX + \frac{1}{2}\,(\delta x_i + \delta x_{i+1})/TANCA$$

$$AVFT = AVFCY - \frac{1}{2}\,(\delta y_j + \delta y_{j+1})\,TANCA$$

where TANCA = tan (CANGLE). The remainder of the procedure remains the same. Similar adjustments are made to the appropriate combination of the variables



(a)



(b)

Fig. 2. (a) Wall adhesion for the locally near horizontal case is modeled by setting the value of AVFR such that CANGLE is the angle between the wall and the surface. (b) Wall adhesion for the locally near vertical case is modeled by setting the value of AVFT such that CANGLE is the angle between the wall and the surface.

AVFL, AVFR, AVFB, AVFT depending on the location of an obstacle or a wall (i.e., whether it be above, below, left or right of the surface cell).

The surface tension scheme together with the standard VOF method can be used to move an initially flat horizontal interface to an equilibrium position consistent with the contact angle and Bond number (i.e., a miniscus). Yet, to do so, in order to establish initial conditions for a calculation, would require considerably more calculational time than solving for the initial surface distribution directly. An ordinary differential equation for the static equilibrium displacement from an initially horizontal surface is found by minimizing the total potential energy. If such a surface displacement is given by $Y_s = f(r)$, $0 \leqslant r \leqslant 1$, then the differential equation is:

$$\frac{1}{r}\frac{d}{dr}\left[\frac{rY_s'}{\sqrt{1 + (Y_s')^2}}\right] - B_o\, Y_s - 2\cos(\text{CANGLE}) = 0$$

subject to the conditions $Y_s'(r = 0) = 0$ and $Y_s'(r = 1) = \tan(\text{CANGLE})$. Further since the volume of fluid must remain constant during the minimization

$$\text{Volume} = 2\pi \int_0^1 rY_s\, dr = 0 \quad .$$

A recursion solution to this equation can be obtained by substituting $Z = \frac{Y_s'}{\sqrt{1 + (Y_s')^2}}$ in the differential equation above and differencing the resulting equations. The difference equations become

$$Z_{sj} = Z_{sj-1}\left(\frac{r_{j-1}}{r_j}\right) + \delta r\left(\frac{r_{j-\frac{1}{2}}}{r_j}\right)\left[2\cos(\text{CANGLE}) - B_o\left(Y_{sj-1} + \frac{\delta r}{2}\frac{Z_{sj-1}}{\sqrt{1 - Z_{sj-1}^2}}\right)\right]$$

and since $Y_s' = \frac{Z_s}{\sqrt{1 - Z_s^2}}$

$$Y_{sj} = Y_{sj-1} + \frac{\delta r}{2}\left(\frac{Z_{sj-1}}{\sqrt{1 - Z_{sj-1}^2}} + \frac{Z_{sj}}{\sqrt{1 - Z_{sj}^2}}\right) \tag{10}$$

and the volume equation becomes YSUM = 0 with

$$YSUM = \sum_{j=1}^{N-1} r_j \, Y_{sj} \, \delta r + \frac{1}{2} Y_{sN} \tag{11}$$

where N = the number of partitions across the interval $0 \leqslant r \leqslant 1$, and

δr = the partition spacing.

The method by which (10) is solved is:

(a) choose a value of $Y_s(0) = -\frac{2 \cos (CANGLE)}{2 |B_0|}$ for $B_0 > 0$; if $B_0 = 0$, set $Y_s(0) = 0$

(b) calculate the Z_{sj}, j = 1, ..., N, $Z_s(0) = 0$

(c) if $Z_{sN} > 1$ replace $Y_s(j = 0)$ with $Y_s(j = 0) \times 1.05$ and repeat step b

(d) compute the Y_{sj}, j = 1, ..., N

(e) if $|Z_{sN} - \cos (CANGLE)| > \varepsilon$ go on to step f, otherwise STOP

(f) compute YSUM

(g) reset $Y_s(0) = Y_s(0) - \frac{YSUM}{2\pi}$

(h) return to step b and repeat the sequence until convergence is reached.

NOTE: If $B_0 = 0$, step e is skipped and after step f the convergence test is made $|YSUM| < \varepsilon$. If this is violated, proceed to step g, then return only to d and repeat this sequence.

Equations (10) and (11) were formulated on the basis of cylindrical coordinates. The equations and procedures for their solution are perfectly valid, however, for plane coordinates if the r and 2π factors are replaced by unity and every appearance of 2 cos (CANGLE) is replaced by cos (CANGLE). In either case the resulting distribution of Y_s can be uniformly augmented to any desired initial surface position and used to determine the set of $F_{i,j}$'s in every column of cells for the mesh selected. This is done by interpolating between the nearest

est values of the Y_{sj} to find a surface height at the center of each column of cells and setting the value of F(i,j) in each column accordingly.

III. NASA SOLA-VOF STRUCTURE

The NASA SOLA-VOF code is written in subroutine form such that each subroutine performs an individual task in the calculation. The subroutine names are symbolically selected to indicate the function that each performs. For the most part, each routine is positioned in the code in the calculational order prescribed in App. A for the solution of the governing equations.

Each subroutine is listed below in the order of its appearance. A brief description is included to describe the major functions of each subroutine.

SOLA-VOF (main program)

(a) Reads and prints the input and output data.

(b) Contains the calling sequences to the other subroutines and thus provides cyclic control over the calculation.

(c) Computes the time step, DELT, used each cycle and increments the time with this value, t → t + DELT.

(d) Increments the cycle number by one each cycle.

(e) Provides a shutdown procedure in the event that a solution cannot be obtained that satisfies mass conservation.

FILMST (FILM SeT-up)

(a) Provides the necessary buffers, links, and logical units for the use of local graphics software. This routine must be written specifically for each graphics system.

MESHST (MESH SeT-up)

(a) Generates the computing mesh from the input data established in NAMELIST/MSHSET/.

(b) Evaluates all of the necessary geometric variables that are used throughout the code.

(c) Computes the relaxation factors (BETA(i,j)) that are used in the pressure iteration.

(d) Sets up obstacles by defining obstacle cells as having BETA(i,j) = - 1.0. Obstacle definition, in general, must be coded by hand for each problem.

SETUP (general set-up)

(a) Initialize constants necessary to the calculation.

(b) Computes the scaling factors and centering shifts required for graphics output.

(c) Calls the ICON subroutine to provide an initial surface configuration for either cylindrical or planar geometry consistent with the contact angle and centerline symmetry boundary conditions. ICON is the routine that initializes the F(i,j) arrays for the entire mesh for the tank draining problem.

(d) Computes the initial hydrostatic pressure distribution to initialize the P(i,j) pressure array.

(e) Initializes marker particle number.

(f) Sets up the initial velocity with U(i,j) = UI and V(i,j) = VI everywhere in the mesh.

ICON (Initial surface CONfiguration)

(a) Computes the solution of the two point boundary value problem for the initial equilibrium position of the free surface. The parameters of the equation are the contact angle (CANGLE) and the Bond number (BOND).

(b) Computes the fractional volume of fluid in each cell F(i,j) based upon the free surface position.

(c) Plots and prints initial surface configuration.

BC (Boundary Conditions)

(a) Sets the values of appropriate variables at rigid free slip, no slip, continuative outflow, and periodic boundaries.

(b) Sets the values of appropriate variables around the boundary established by the free surface.

(c) Allows for special boundary condition inclusions, such as inflow boundaries; these must be included by hand as needed for each problem in general. However, for the tank draining problem, the set-up is fixed for inflow or outflow at the tank outlet.

TILDE

(a) Computes an explicit solution for each of the momentum equations. (i.e., new values of velocities are obtained from the time n values of pressure, convective and diffusive accelerations.) These tilde values will be advanced to time n+1 values in the pressure iteration.

PRESIT (PRESsure ITeration)

(a) Iterates the velocity and pressure field such that mass is conserved in each cell of the mesh (i.e., $|D(i,j)| < EPSI$), except surface cells.

(b) Computes a surface cell pressure adjustment based on the applied surface pressure, yet mass conservation in the surface is not iterated, it is set by application of the free surface boundary conditions.

PARMOV (PARticle MOVement)

(a) Computes the movement of marker particles in the velocity field just found.

(b) Provides the necessary bookkeeping to allow marker particles that exit the mesh to be replaced by newly input particles. (NOTE: Particle initializations must be done by hand in the SETUP subroutine.)

VFCONV (Volume Fraction CONVection)

(a) Computes the solution to the equation

$$\frac{\partial F}{\partial t} + \nabla \cdot \underset{\sim}{u}F = 0 \quad .$$

(b) Computes and stores for printout any errors in volume (i.e., loss or gain) during the calculation of step (a).

PETACL (PETA interpolation factor CaLculation)

(a) Used only for surface cells.

(b) Determines the slope of the surface in surface cells.

(c) Determines the cell flag, NF(i,j) to indicate the interpolation neighbor of the surface cell. (The interpolation neighbor is the cell adjacent to the surface cell containing fluid and with which surface cell pressures are interpolated to provide the proper value of P_s at the surface.)

NF = 1, implies a neighbor to the left.

NF = 2, neighbor to the right.

NF = 3, neighbor on the bottom.

NF = 4, neighbor on the top.

Thus, the above information describes the orientation of the surface (whether vertical or horizontal) and on which side of this surface fluid exists.

(d) Computes the surface pressure PS(i,j) caused by surface tension in surface cells.

(e) Computes the factor PETA(i,j) that is a measure of the nondimensional distance from the cell center to the surface along the cell midline toward the interpolation neighbor.

DRAW (Generates graphics output of problem data)

(a) Draws velocity vector and free surface distributions.

(b) Draws the mesh during CYCLE = 0.

(c) Provides particle plots as requested.

FRAME

(a) Draws a frame around graphics output (the frame size is scaled to the mesh size).

DRWOBS (DRaW OBStacles)

(a) Draws lines around all obstacles. In addition, the hemisphere on the bottom is drawn as a reference.

PLTPT (PLoT a PoinT)

(a) Provides the graphics system call to plot a single point (x1,y1).

(b) Computes and plots the symmetric point to be plotted if the flag ISYMPL is on. (Symmetry is always assumed to exist only about the y axis.)

DRWVEC (DraW a VECtor)

(a) Provides the graphics package system call to draw a line between points (x1,y1) and (x2,y2).

(b) Computes and plots the symmetric form of a given line if ISYMPL is turned on (i.e., = 1).

A simplified flow diagram showing the calling sequence of the routines is presented in Fig. 3.

IV. PRIMARY FORTRAN VARIABLES

The primary variables used in the program, other than those in the NAMELIST input blocks, are listed below. The algebraic form of the variable used in the finite difference equations is given with a brief description.

VARIABLE		DESCRIPTION
BETA(I,J)	$\dfrac{\omega}{\delta t\left(\dfrac{1}{\delta x_i^2}+\dfrac{1}{\delta y_j^2}\right)}$	Relaxation factor.
CYCLE		Cycle number.

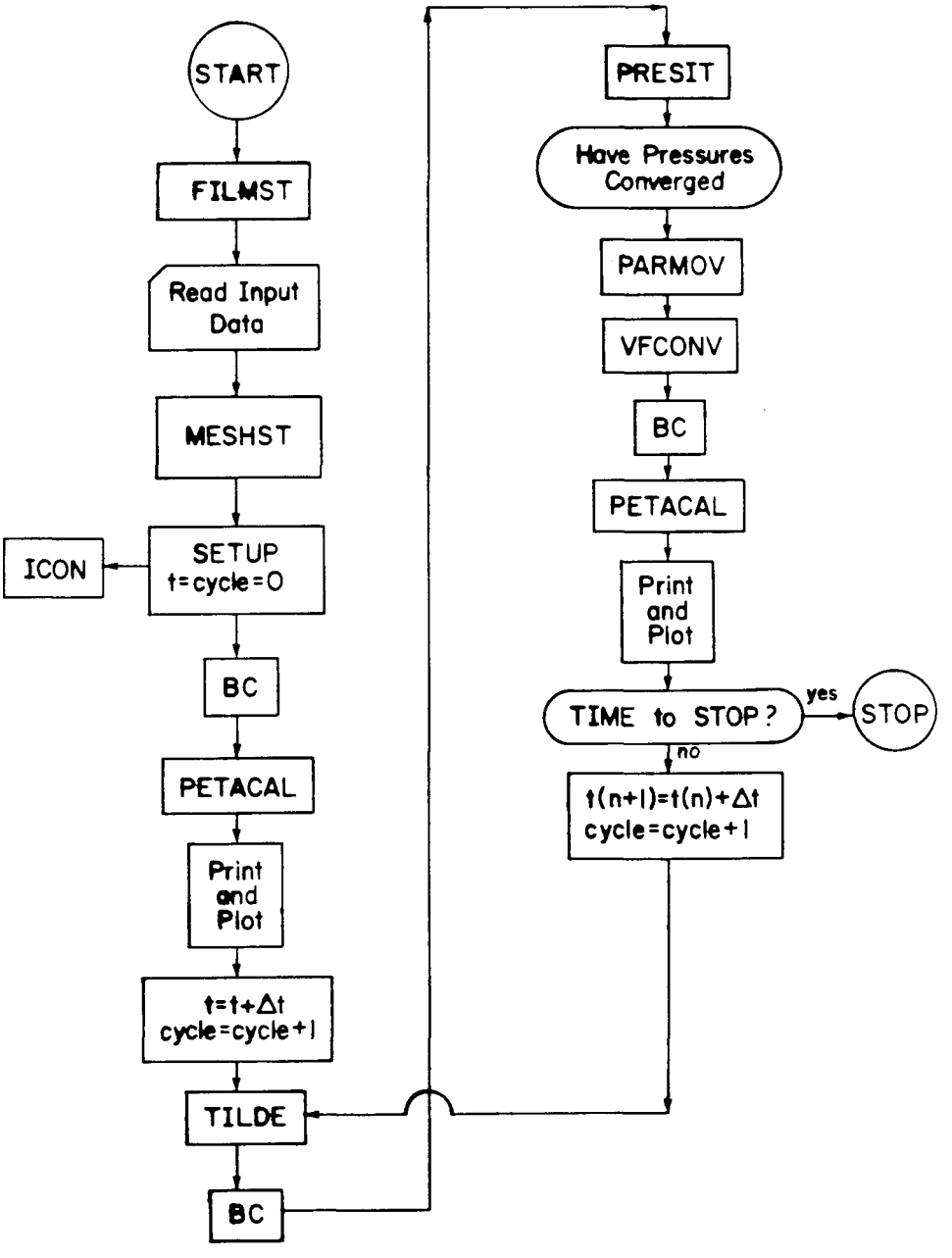


Fig. 3. Flow chart of the NASA SOLA-VOF code.

CURV(K)	$\frac{d^2G/dX^2}{[1 + (dG/dX)^2]^{3/2}}$	Theoretical curvature of kth segment of initial surface with respect to x direction (ONLY FOUND IN ICON).
CURV	$\frac{1}{R_{xy}} + \frac{1}{R_{cyl}}$	The numerical sum of the principal curvatures.
CURVCY	$\frac{1}{R_{cyl}}$	Azimuthal curvature for cylindrical geometry.
CURVY(K)		Theoretical curvature of kth segment of initial surface with respect to y direction.
CURVXY	$\frac{d^2G/dY^2}{[1 + (dG/dY)^2]^{3/2}}$	Numerical curvature with respect to either x or y direction.
D(I,J)	$\nabla \cdot \underset{\sim}{u}$	The divergence of the velocity field.
DELT	δt	Time step.
DELX(I)	δx_i	The mesh spacing of the ith cell along the x axis.
DELY(J)	δy_j	The mesh spacing of the jth cell along the y axis.
F(I,J)		The volume of fluid per unit volume of cell (i,j) at time level n+1.
FLG		A flag that indicates convergence of pressure iteration if = 0, nonconvergence if = 1.
FN(I,J)		The volume of fluid per unit volume of cell (I,J) at time level n.
FUX	$u\ \partial u/\partial x$	The flux of u momentum in the x direction.
FUY	$v\ \partial u/\partial y$	The flux of u momentum in the y direction.
FVX	$u\ \partial v/\partial x$	The flux of v momentum in the x direction.
FVY	$v\ \partial v/\partial y$	The flux of v momentum in the y direction.
GP	G'	The derivative of the surface height with respect to either x or y depending on surface slope.
GPP	G"	The second derivative of surface height with respect to either x or y depending on surface slope.

HCL		Height of the surface at the centerline of the tank.
HWALL		Height of the surface at the wall of the tank.
IBAR		The number of real cells in the x direction.
IMAX	IBAR+2	The number of real plus fictitious cells (2) in the x direction.
IM1	IMAX-1	The value of I at the last real cell in the x direction.
IP(k)		The index of the cell in the x direction containing particle k.
ITER		The iteration number.
JBAR		The number of real cells in the y direction.
JMAX	JBAR+2	The number of real plus fictitious cells (2) in the y direction.
JM1	JMAX-1	The value of J at the last real cell in the y direction.
JP(k)		The index of the cell in the y direction containing particle k.
NAME		The problem title that is read in as input in a 20A4 format.
NF(I,J)		The flag of surface cell (I,J) indicating the location of its interpolation neighbor.
P(I,J)		The pressure of cell (I,J) at time level n+1.
PETA(I,J)		The ratio of the distance between cell centers to the distance between the surface and center of the interpolation cell.
PN(I,J)		The pressure of cell (I,J) at time level n.
PS(I,J)	-SIGMA · CURV	The surface pressure computed from the surface tension coefficient and curvature.
RDX(I)	$1/\delta x_i$	
RDY(J)	$1/\delta y_j$	
RX(I)	$1/X_i$	
RXI(I)	$1/XI_i$	
RYJ(J)	$1/YJ_j$	

SF		Scale factor for plotting.
SIGMA	σ = .0001/WEBER	The surface tension force per unit length for the tank draining problem.
T		Time
TANCA	tan (CANGLE)	Tangent of the contact angle.
U(I,J)		The x direction velocity on the right side of cell (I,J) at time n+1.
UN(I,J)		The x direction velocity on the right side of cell (I,J) at time n.
V(I,J)		The y direction velocity at the top of cell (I,J) at time n+1.
VCHGT		The accumulated loss or gain of F from inaccuracies in numerical solution of F advection.
VINIT	$2\pi/3$	Initial nondimensional volume of hemispherical bottom of tank.
VISX		The viscous accelerations in the x direction.
VISY		The viscous accelerations in the y direction.
VN(I,J)		The y direction velocity at the top of cell (I,J) at time n.
VOLBAR	VOLUME/VINIT	The ratio of the fluid volume in the tank to that of the hemispherical bottom.
VOLUME		The volume of fluid in the tank.
X(I)	X_i	The x distance to the right edge of cell (I,J).
XI(I)	XI_i	The x distance to the center of cell (I,J).
XP(k)		The x coordinate of particle k.
XSHFT		The shift along the plotting absicca to center the plot frame on film.
Y(J)	Y_j	The y distance to the top of cell (I,J).
YJ(J)	YJ_j	The y distance to the center of cell (I,J).
YP(k)		The y coordinate of particle k.
YS(K)		The solution height of the initial surface at segment k (used in ICON).

YSHFT		The shift along the plotting ordinate to center the plot frame on film.
ZS(K)	$\sin\theta$	The sin of the angle of deviation from horizontal of the kth segment of the surface (used in ICON).

V. INPUT VARIABLES

All of the input data to the NASA SOLA-VOF code are in NAMELIST blocks with the exception of the problem title NAME which is a 20A4 format. The NAMELIST variables occur in two blocks. The first block is /XPUT/ and contains all of the physical parameters necessary to specify and run the problem. The second block /MSHSET/ contains the geometrical information necessary for the creation of the variable mesh and the number of cells used in it.

The following is a list of all the input variables and a description of each.

NAMELIST /XPUT/

DELT	The time step used either to start the problem (must be chosen in accord with stability criteria) or to run the problem with a constant time step.
NU	The kinematic viscosity of the fluid.
CYL	= 0 for plane geometry. = 1 for cylindrical geometry.
EPSI	The convergence criterion for the pressure iteration (i.e., $\lvert\nabla\cdot\underset{\sim}{u}\rvert$ < EPSI). This is typically set to 10^{-3} for most problems scaled such that velocities are of the order of unity.
DZRO	$\lvert\nabla\cdot\underset{\sim}{u}/\text{DZRO}\rvert$ < EPSI is the true convergence test. DZRO = 1.0 satisfies the criterion above for the selection of EPSI. However, DZRO can be used to accommodate any desired magnitudes of velocity while keeping EPSI on the order of 10^{-3}.
GX	Acceleration of gravity or environment in x direction.

GY	Acceleration in y direction similar to GX.
UI	The initial U velocity to be set everywhere in the mesh.
VI	The initial V velocity to be set everywhere in the mesh.
VELMX	The maximum velocity to which all velocity vectors in the mesh will be scaled upon plotting.
TWFIN	Time when to finish the calculation.
PRTDT	Print time step (i.e., time interval between prints on paper).
PLTDT	Plot time step (i.e., time interval between plots on film).
OMG	The overrelaxation coefficient for the pressure iteration. Typically OMG = 1.8 but can be picked $1.0 \leq OMG < 2.0$.
ALPHA	The parameter that specifies the relative amount of centered or donor cell differencing of the advective flux terms. ALPHA = 0 for centered differencing. ALPHA = 1 for donor differencing. Fractional values between 0 and 1 can also be used.
WL,WR,WT,WB	Stands for wall left, right, top, and bottom, respectively. These flags set the desired wall boundary condition. Each of the parameters can assume the following values = 1 for rigid free-slip wall = 2 for rigid no-slip wall = 3 for continuative outflow boundary = 4 for periodic boundary. NOTE: Inflow boundaries must be prescribed by hand in the special boundary condition section of subroutine B.C.
PARTN	The number of marker particles to be placed in the flow field.

CWTD,TRST		Cycle when to tape dump, time to restart. Can be neglected in the NASA SOLA-VOF code as there is no tape dump capability.
MOVY		= 0 if no movie is desired = 1 if a movie is desired.
DTMVP		Problem time interval between movie frames.
AUTOT		Automatic time step flag = 0 specifies that the constant DELT previously given is used for the calculation. = 1 specifies that the time step DELT will automatically be computed based on the stability condition $\text{DELT} = \text{MIN}\left(\frac{\delta x_i}{\lvert U_{MAX}\rvert}, \frac{\delta y_j}{\lvert V_{MAX}\rvert}\right)$.
FLHT		Fluid height (i.e., the initial vertical position of a horizontal surface). Subroutine ICON finds a miniscus solution about this height. (For tank draining problems this can also be termed the initial fill height. Since the tank radius is R = 1 in this code, the value of FLHT is just set to the number of desired fill heights.)
ISYMPL		= 0 no symmetry plots = 1 film frames are drawn as a symmetric plot about the vertical axis.
WEBER	We	Weber number WEBER = .0001/SIGMA for the tank draining problem.
BOND	Bo	BOND number. GY = SIGMA x BOND is set from this specification and overrides the value of GY previously set.
ISRF10		= 0 no surface tension effects are included in calculation = 1 turns on surface tension forces.
CANGLE		The contact angle between the fluid and tank wall. (Specified in degrees.)

VOUT	The outflow velocity (is predetermined to be = - 1.0 for all tank draining problems). Set VOUT = + 1.0 to perform inflow or tank filling calculations.
IBAFF	= 0 means no obstacles are placed in the tank for a baffle (i.e., unbaffled draining) = 1 places the baffle obstacles in the flow field. For best results, the baffle must be used in a mesh that has mesh lines at x = .625, y = 0.4, and y = 0.56.

NAMELIST /MSHSET/

NKX	The number of submeshes used to compose the complete mesh that spans the x direction.
XL	The x coordinate of the left edge of a submesh. Must be specified for NKX submeshes.
XC	The x coordinate of the convergence point of a submesh. Must be specified for NKX submeshes.
XR	The x coordinate of the right edge of a submesh. Must be specified for NKX submeshes.
NXL	The number of cells to be placed between coordinates XL and XC in a submesh. Must be specified for NKX submeshes.
NXR	The number of cells to be placed between coordinates XC and XR in a submesh. Must be specified for NKX submeshes.
DXMN	The minimum value of δx_i that occurs in a submesh on each side of the convergence point XC. Must be specified for NKX submeshes.

The following input numbers generate the mesh in the y direction and are analogous to the x values previously specified.

NKY	The number of submeshes in the y direction that compose the complete mesh.
YL	The y coordinate of the left edge of a submesh as one views the submesh in the direction of the negative X axis.
YC	The y coordinate of the convergence point of a submesh as one views it in the direction of the negative x axis.
YR	The y coordinate of the right edge of the submesh, as one views it in the direction of the negative X axis.
NYL	The number of cells in a submesh between locations YL and YC.

NYR The number of cells in a submesh between locations YC and YR.

DYMN The minimum value of δy_j that occurs in a submesh on each side of the convergence point.

YCENTR The y coordinate of the center of the hemisphere at the tank bottom. This should be set at a value of 1.0 plus the height of the neck of the outlet. For the test problems already done, this value was chosen to be = 1.1.

The variable mesh is constructed by linking a group of submeshes together to achieve any desired distribution of cell spacing. This is done in the same manner in both directions. The number of cells is specified in each submesh on each side (i.e., to the left and to the right) of the convergence point. Both cells directly adjacent to the convergence point have a cell spacing equal to the minimum value specified in the input as DXMN or DYMN. The cell spacing is then expanded quadratically from the convergence point cell to the left and right edges of the submesh in accordance with the number of desired cells on either side. If a uniform cell spacing on the left (right) has a cell size that is less than the minimum size input as DXMN or DYMN, a uniform spacing is then used on the left (right). The number of cells to the left and to the right of the convergence point need not be equal but there must be at least one on both sides.

When two or more submeshes are linked together, it is imperative that the location of the left edge of the right submesh be the same as the location of the right edge of the left submesh.

An example of the proper format to be used to specify a mesh spanning the x dimension $LW \leqslant X \leqslant RW$ with n submeshes is

$$NKX = n,\ XL = LW,\ XL_2,\ XL_3,\ \ldots XL_n,\ XC = XC_1,\ XC_2,\ \ldots,\ XC_n,$$
$$XR = XL_2,\ XL_3,\ \ldots,\ RW,\ NXL = NL_1,\ NL_2,\ \ldots,\ NL_n,$$
$$NXR = NR_1,\ NR_2,\ \ldots,\ NR_n,\ DXMN = DXMN_1,\ \ldots,\ DXMN_n$$

in which NL_i represents the number of cells to the left of XC_i and NR_i is the number of cells to the right of XC_i in each submesh i, i = 1, ..., n.

VI. SETTING UP A PROBLEM

Since the NASA SOLA-VOF code has been modified for the tank draining problem, there are no special inclusions needed to run these problems. In general, however, special sections may be added by hand for inflow boundaries and any other special boundary conditions not provided (in the B.C. subroutine) and to ini-

tialize obstacles (in the MESHST subroutine). The hemispherical tank section is included in the code with a procedure that compares the distance from the hemispherical center (0,YCENTR) to the center of each cell i,j (XI_i, YJ_j) with R = 1. If the distance is greater than 1, the cell flagged with BETA(i,j) = -1.0 for an obstacle cell; if the distance is less than 1, the cell is assumed to contain fluid. Other than this, all that is required is to initialize the NAMELIST/XPUT/ and NAMELIST/MSHSET/ variables. Once these variables are determined it is imperative that the PARAMETER statement at the very beginning of the code be modified to provide the proper storage requirements for the problem selected. The PARAMETER statement has the form

PARAMETER(IBAR2 = 13, JBAR2 = 38, NPRTS = 1, MSHX =4, MSHY = 4).

The variables named in this statement are used to set the dimensions of the COMMON blocks at compile time and cannot be used as variables elsewhere in the code. The values specified in the parameter statement are easily determined from the input data

IBAR2 = NXL + NXR + 2
JBAR2 = NYL + NYR + 2
NPRTS = PARTN
MSHX = NKX
MSHY = NKY

The dimensions can be larger than these equations specify but never smaller.

Due to the 32 bit word length of the UNIVAC 1104 computer, the code NASA SOLA-VOF has been modified to allow DOUBLE PRECISION computing. This feature provides 64 bit computing, similar to that of a CDC-7600, the computer on which the code was developed. A requirement of this feature regarding input data is that floating point input numbers must be specified in double precision.

Thus, the setup procedure for running the NASA SOLA-VOF code is very straightforward and simple. For tank draining problems, one can easily use the mesh setup data provided in the next section of this report for the baffled and unbaffled cases. There is, of course, no restriction on the mesh used for the unbaffled problem, only geometric requirements. In this case, it is crucial to place a vertical grid line at X = 0.1 which is the radius of the outlet and to have a smooth variation of DELX across the mesh due to the accuracy requirements outlined in App. A.

The mesh provided for the baffled draining problem is another matter and should be used for all baffled problems. If finer resolution is desired a scaled analog of this mesh is required. The reason is because of the position of the baffle relative to the outlet. The general prescription of defining the annular opening from the mesh data provided allows no cells to resolve the space between the baffle and hemispherical tank wall. A special condition is provided in subroutine MESHST to flag cell (8,5) as a fluid cell and thus provide a single cell resolution of the annular space. A more desirable situation is to have at least two cells across such an opening, yet the cost of such a desire for the tank draining problem is great both in terms of storage and computer time. Any changes to the mesh for baffled problems should be accompanied by either a removal or alteration of the statement IF(IBAFF.GT.0.AND.I.EQ.8.AND.J.EQ.5) GO TO 140 in the obstacle setting section of the MESHST.

VII. TEST PROBLEMS

Five test cases were performed to demonstrate the utility of the NASA SOLA-VOF code for tank draining problems. Test cases 1 and 2 are unbaffled problems while cases 3, 4, and 5 are for baffled tanks. A summary of the parameters involved in each problem is given in Table I. It should be noted that a few of the test cases were recomputed with a contact angle of 1°. No apparent differences (from calculations with 5° contact angle) were observed in the solutions so obtained. Thus, there appears to be a calculational insensitivity to small contact angles except for flows strongly dominated by surface tension.

The remainder of this section will be devoted to the discussion of the test cases and a comparison, where possible, of calculated with experimental results.

A. Case 1

The input data for this calculation are given in Table II. These data are arranged as a NAMELIST data set with the first part of the data being the /XPUT/ block and the latter part, the /MSHSET/ block. The symmetrically drawn mesh generated by the /MSHSET/ data is shown in Fig. 4a in which calculations are performed only on the right half.

The initial free surface position that results from the ICON subroutine is shown in Fig. 4b. A comparison of the initial free surface as determined by the F = 0.5 contour line can be made with Fig. 4c. (Note: All free surface plots that are produced by the code utilize the F = 0.5 contour line to approximate the

TABLE I

TEST PROBLEM PARAMETERS

Case Number	1	2	3	4	5
Weber	1.06	0.01	1.52	4.05	9.12
Bond	-5	0	0	0	0
Initial Fill Height	3	2	2	2	2
Contact Angle	5°	5°	5°	5°	5°
Baffle	No	No	Yes	Yes	Yes

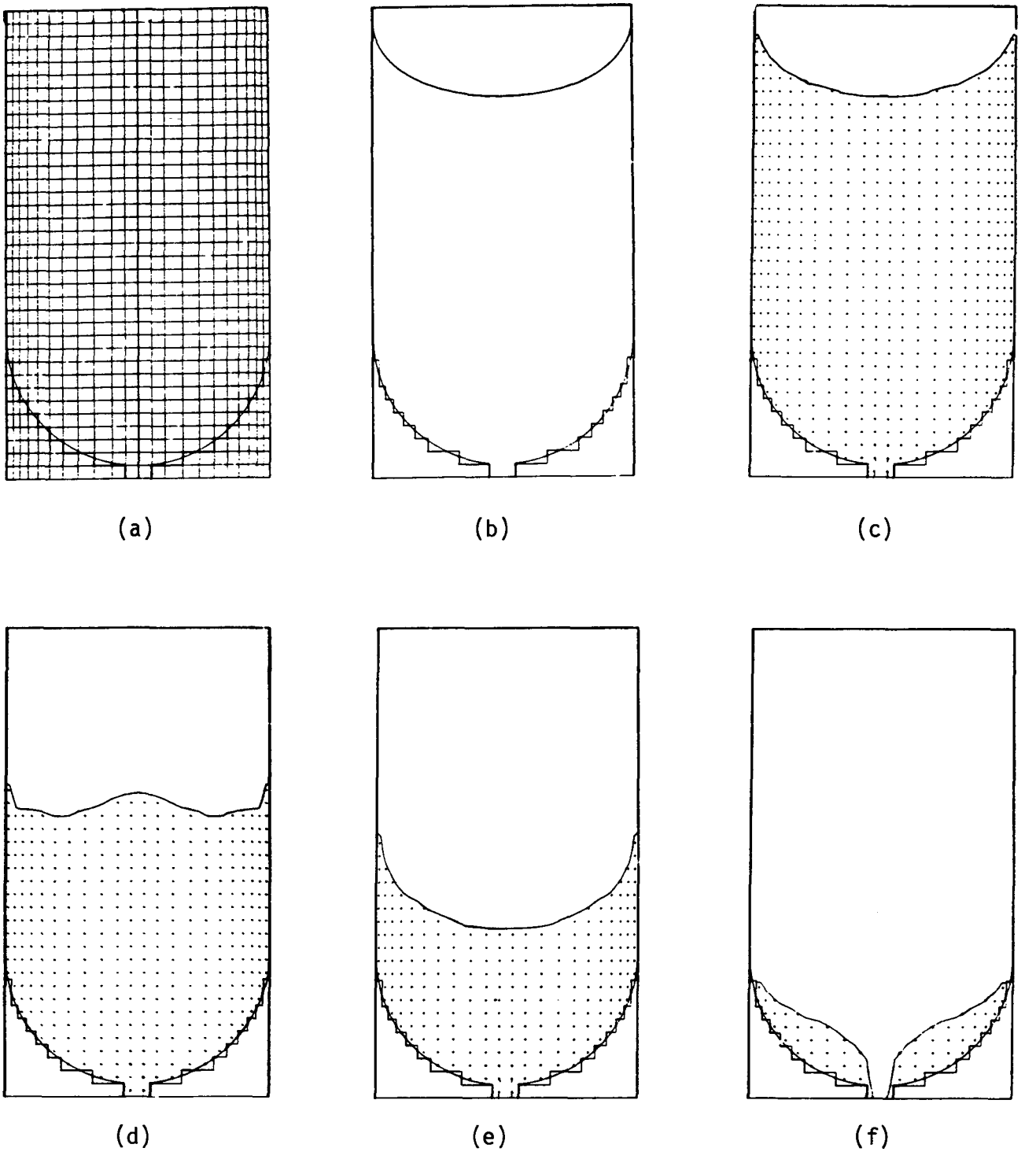


Fig. 4. Sequence of plots for Case 1. (a) Symmetrically plotted mesh used, (b) initial surface from ICON, (c) free surface plot at t = 0, (d) surface configuration at t = 154, (f) surface configuration at vapor ingestion showing residual volume at t = 241.

TABLE II

NAMELIST INPUT DATA FOR EACH TEST CASE

```
      NASA TANK DRAINING TEST PROBLEM  -- COURSE MESH --CASE 1
DELT=1.0D-04 NU=0.0 CYL=1.0 EPSI=.001 DZRO=1.0 GX=0.0 GY=0.0 UI=0.0
VI=0.0 VELMX=1.0 TWFIN=500.0 PRTDT=99999.0 PLTDT=1.0 OMG=1.8 ALPHA=1.0
WL=1 WR=1 WT=1 WB=1 PARTN=0.  MOVY=0 DTMVP=1.5
AUTOT=1. FLHT=3.0 ISYMPL=1
ISRF10=1 WEBER=1.06 BOND=-5.0 VOUT=-1.0 CANGLE=5.0
XPUT
NKX=2 XL=0. .2 XC=.1 .95 XR=.2 1.0 NXL=1 8 NXR=1 1 DXMN=.1 .05
NKY=1 YL=0. YC=1.8 YR=3.6 NYL=18 NYR=18 DYMN=0.1 YCENTR=1.1
MSHSET
```

```
      NASA TANK DRAINING TEST PROBLEM  -- COARSE MESH-- CASE 2
DELT=1.0D-04 NU=0.0 CYL=1.0 EPSI=.001 DZRO=1. GX=0. GY=0. UI=0.
VI=0. VELMX=1. TWFIN=500.0 PRTDT=99999. PLTDT=1.0 OMG=1.8 ALPHA=1.
WL=1 WR=1 WT=1 WB=1 PARTN=0.  MOVY=0 DTMVP=1.5
AUTOT=1. FLHT=2.0 ISYMPL=1
ISRF10=1 WEBER=.01 BOND=0.0 VOUT=-1.0 CANGLE=5.0
XPUT
NKX=1 XL=0. XC=0.5 XR=1.0 NXL=5 NXR=5 DXMN=0.1
NKY=1 YL=0. YC=1.8 YR=3.6 NYL=18 NYR=18 DYMN=0.1 YCENTR=1.1
MSHSET
```

```
      NASA TANK DRAINING TEST PROBLEM  -- COURSE MESH --CASE 3
DELT=1.0D-04 NU=0.0 CYL=1.0 EPSI=.001 DZRO=1. GX=0. GY=0. UI=0.
VI=0. VELMX=1. TWFIN=500.0 PRTDT=99999. PLTDT=1.0 OMG=1.8 ALPHA=1.
WL=1 WR=1 WT=1 WB=1 PARTN=0.  MOVY=0 DTMVP=1.5
AUTOT=1. FLHT=2.0 ISYMPL=1 IBAFF=1
ISRF10=1 WEBER=1.52 BOND=0.0 VOUT=-1.0 CANGLE=5.0
XPUT
NKX=3 XL=0. .2 .625 XC=.1 .4125 .8125 XR=.2 .625 1.0
NXL=1 2 2 NXR=1 2 2 DXMN=.1 .10625 .09375
NKY=3 YL=0. .4 .56 YC=.2 .48 2.08 YR=.4 .56 3.6
NYL=2 1 15 NYR=2 1 15 DYMN=.1 .08 .101333
YCENTR=1.1
MSHSET
```

```
      NASA TANK DRAINING TEST PROBLEM  -- COURSE MESH --CASE 4
DELT=1.0D-04 NU=0.0 CYL=1.0 EPSI=.001 DZRO=1. GX=0. GY=0. UI=0.
VI=0. VELMX=1. TWFIN=500.0 PRTDT=99999. PLTDT=1.0 OMG=1.8 ALPHA=1.
WL=1 WR=1 WT=1 WB=1 PARTN=0.  MOVY=0 DTMVP=1.5
AUTOT=1. FLHT=2.0 ISYMPL=1 IBAFF=1
ISRF10=1 WEBER=4.05 BOND=0.0 VOUT=-1.0 CANGLE=5.0
XPUT
NKX=3 XL=0. .2 .625 XC=.1 .4125 .8125 XR=.2 .625 1.0
NXL=1 2 2 NXR=1 2 2 DXMN=.1 .10625 .09375
NKY=3 YL=0. .4 .56 YC=.2 .48 2.08 YR=.4 .56 3.6
NYL=2 1 15 NYR=2 1 15 DYMN=.1 .08 .101333
YCENTR=1.1
MSHSET
```

```
      NASA TANK DRAINING TEST PROBLEM  -- COURSE MESH --CASE 5
DELT=1.0D-04 NU=0.0 CYL=1.0 EPSI=.001 DZRO=1. GX=0. GY=0. UI=0.
VI=0. VELMX=1. TWFIN=500.0 PRTDT=99999. PLTDT=1.419 OMG=1.8 ALPHA=1.
WL=1 WR=1 WT=1 WB=1 PARTN=0.  MOVY=1 DTMVP=1.419
AUTOT=1. FLHT=2.0 ISYMPL=1 IBAFF=1
ISRF10=1 WEBER=9.12 BOND=0.0 VOUT=-1.0 CANGLE=5.0
XPUT
NKX=3 XL=0. .2 .625 XC=.1 .4125 .8125 XR=.2 .625 1.0
NXL=1 2 2 NXR=1 2 2 DXMN=.1 .10625 .09375
NKY=3 YL=0. .4 .56 YC=.2 .48 2.08 YR=.4 .56 3.6
NYL=2 1 15 NYR=2 1 15 DYMN=.1 .08 .101333
YCENTR=1.1
MSHSET
```

free surface; a procedure that is completely satisfactory for such calculations as these.) Figures 4d-4f show the free surface configuration at selected times with the last frame at the time vapor ingestion occurs, which shows the residual volume remaining in the tank. Table III is a list of the residual volumes computed for each case. A careful inspection of Fig. 4 will reveal the fact that the angle made between the wall and surface in the free surface plots (beginning with Fig. 4c) is not the contact angle previously reported. This consequence is only an artifact of the computer generated drawing since no special procedure was built into the graphics routines to accurately convey the wall-surface intersection. Thus, one must make a mental extrapolation of the surface near the wall to visualize the proper contact angle.

Figure 5 shows a history of the centerline and wall positions based on the nondimensional time t. The curve for the centerline shows a few surface waves superimposed upon a near linear decline while the curve for the wall displays a stepwise appearance. The centerline curve is an accurate representation of the surface history at the cylindrical axis. The wall curve, however, represents the location of the top of the last wall surface cell as a function of time. This method of defining the wall contact point was chosen for simplicity and convenience, and is admittedly a coarse approximation. The wall surface location is a rather arbitrary point to define due to the fact that the surface is modeled by fractional values of F and the application of a contact angle boundary condition at the wall. To be consistent with various optical methods used experimentally to determine wall-surface interface, a more complicated method would have to be devised.

A comparison of the Case 1 calculations with the experiment reported in [6] is made in Fig. 12. There is generally excellent agreement between the two, although the calculation shows more sloshing than the experiment. Part of this sloshing phenomena is due to the coarse mesh used to perform the calculation. The rather large cells near the centerline prevent the precise evaluation of the surface pressure (i.e., curvature) necessary to retard the growth of the surface convexity.[†]

[†]Since the completion of the NASA SOLA-VOF code, the ongoing development of the SOLA-VOF code has produced changes in the fluxing terms of the F equation that significantly improve the comparison presented. A finer mesh can also improve the accuracy at the centerline. However, it is not the only source of error. Concerned investigators are advised to follow the current developments of the SOLA-VOF code.

TABLE III

RESIDUAL VOLUME

[VOLBAR = $V_{RES}/(2\pi/3)$]

Case 1	0.42
Case 2	0.46
Case 3	0.51
Case 4	0.69
Case 5	0.77

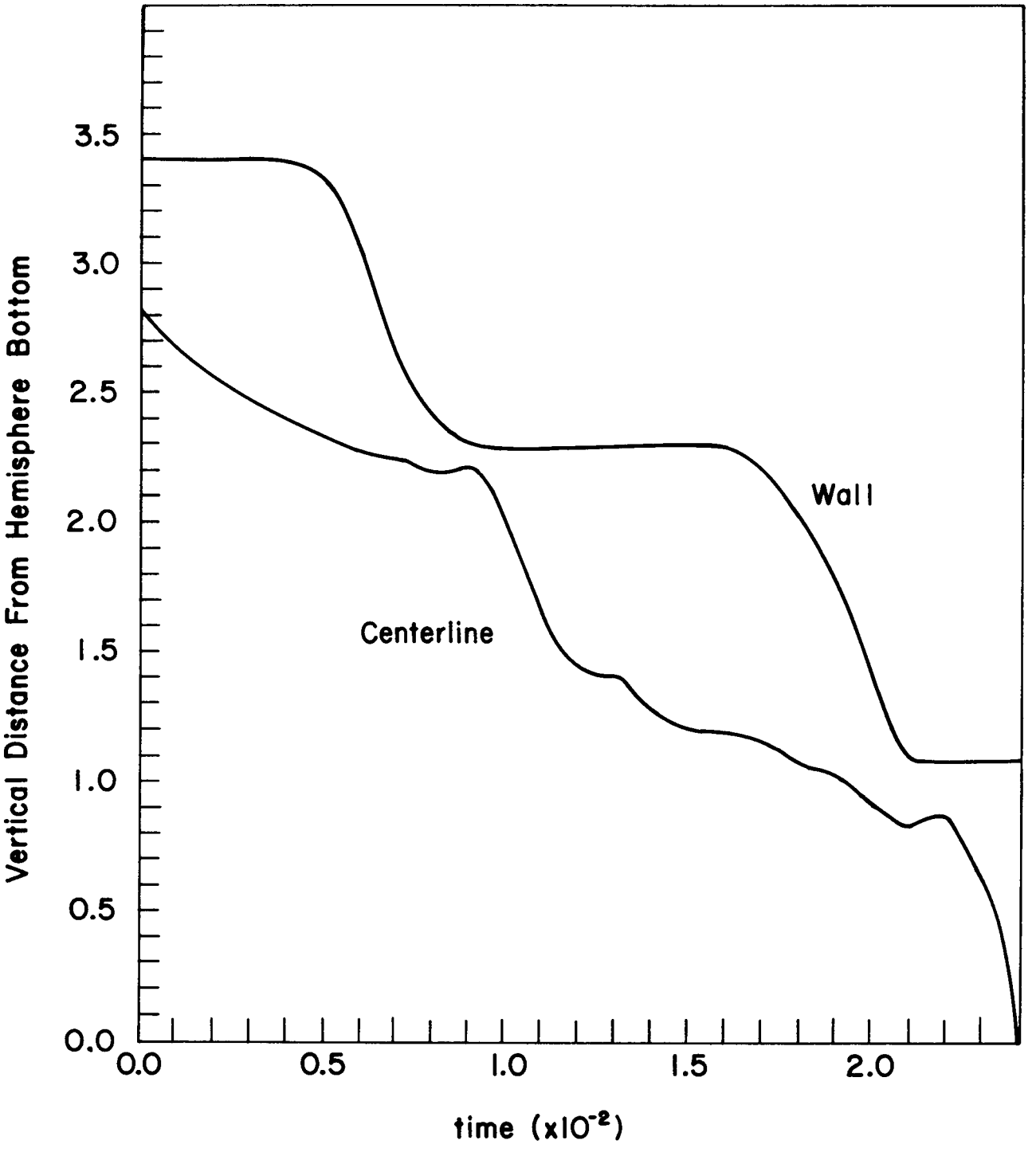


Fig. 5. Centerline and wall contact point histories for Case 1.

As previously mentioned, the contact point at the wall shows the consequences of the simple contact point definition, however, the curve shows the correct average slope and can therefore be useful for some engineering purposes.

B. Case 2

Case 2 calculations were made with the input data given in Table II. A sequence of the resulting free surface plots is shown in Fig. 6. Figures 6a and 6b show the solution of the equilibrium position differential equations and the initial surface represented by the F = 0.5 contour, respectively. The remaining figures show the surface at selective times including the time at which vapor ingestion occurs (Fig. 6f) and the residual volume in the tank (see Table III). Figure 7 shows the history of the centerline and wall contact point positions as previously discussed. Here, one sees the sloshing that takes place in this zero gravity, surface tension dominated flow. No experimental results are available for comparison with this calculation.

C. Cases 3, 4 and 5

The three baffled draining cases exhibit such similarity that they are presented together. Figure 8 shows the typical flow pattern for the three cases. Although Fig. 8 contains plots of the Case 4 calculation, the other two cases exhibit only slight differences in free surface position at the times given.

Figure 8a shows the mesh used to calculate all of these cases with the baffle represented by the cross-hatched area. The data used for the mesh are specified in Table II. (Note that the only differences in Table II for Cases 3-5 is the Weber number and the data used to set up a movie run for Case 5.) Figures 8b-8f show a sequence of surface positions at various times. These cases exhibit no sloshing and the only major difference in them is shown in Fig. 8f in which the time at which vapor ingestion occurs is given for each of the cases. This figure shows the fluid column just prior to vapor ingestion. Figures 9-11 exhibit the centerline and wall contact point behavior of Cases 3-5. The wall contact point does not move in any of the cases, while the centerline histories indicate a steady decline of the surface with a very rapid vapor ingestion. The typical residual volume pattern as shown in Fig. 8f indicates that fluid is trapped by wall adhesion on top of the baffle and along the wall. The free surface contours that appear in this figure across the annular space between the disk baffle and the hemispherical bottom are a result of the numerical errors that occur from the single cell across the space. It is usually more desirable to place two or more

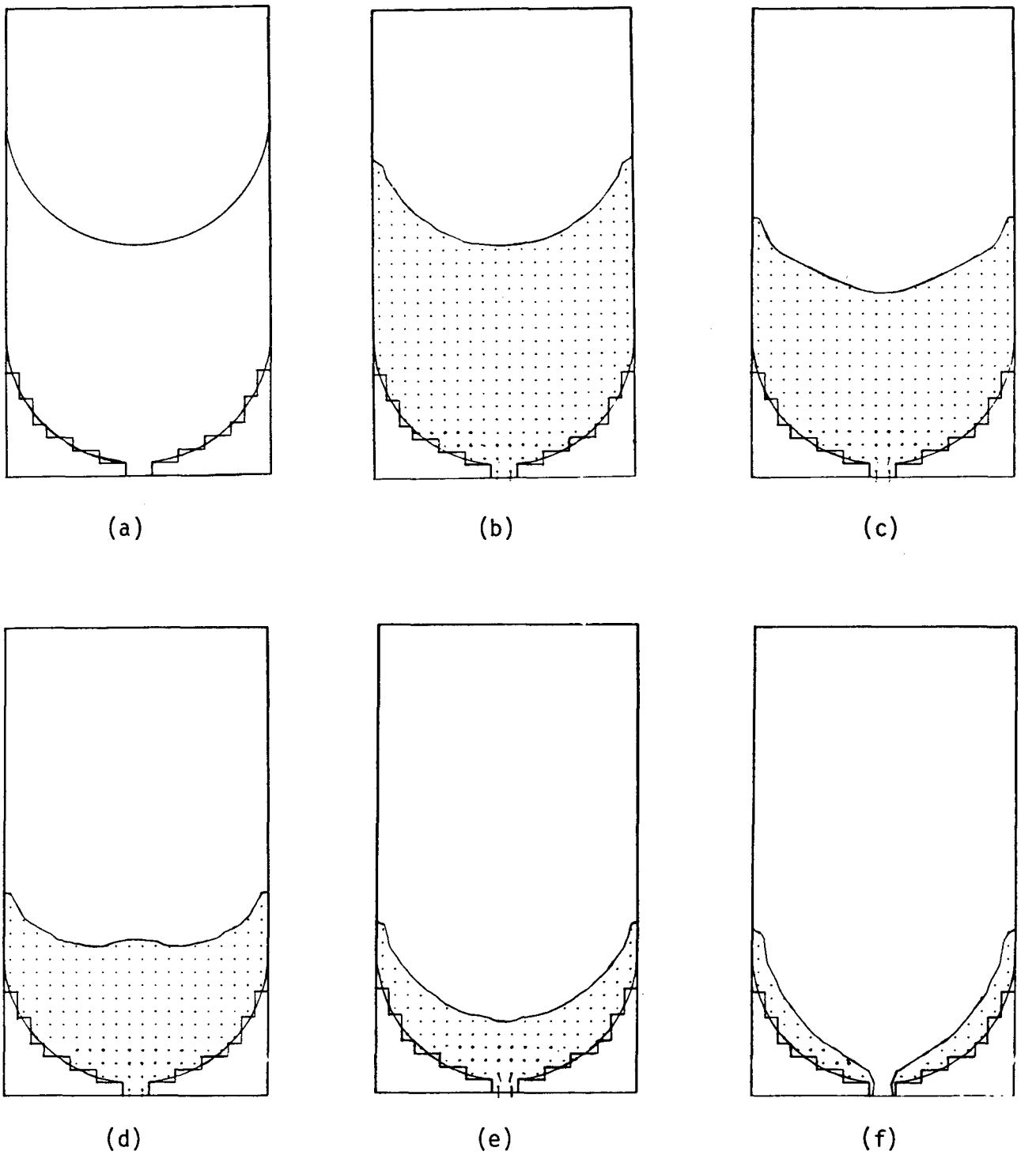


Fig. 6. Sequence of plots for Case 2. (a) Equilibrium surface solution from ICON, (b) free surface plot at t = 0, (c) free surface at t = 40, (d) free surface at t = 80, (e) free surface at t = 100, (f) residual volume at vapor ingestion t = 139.

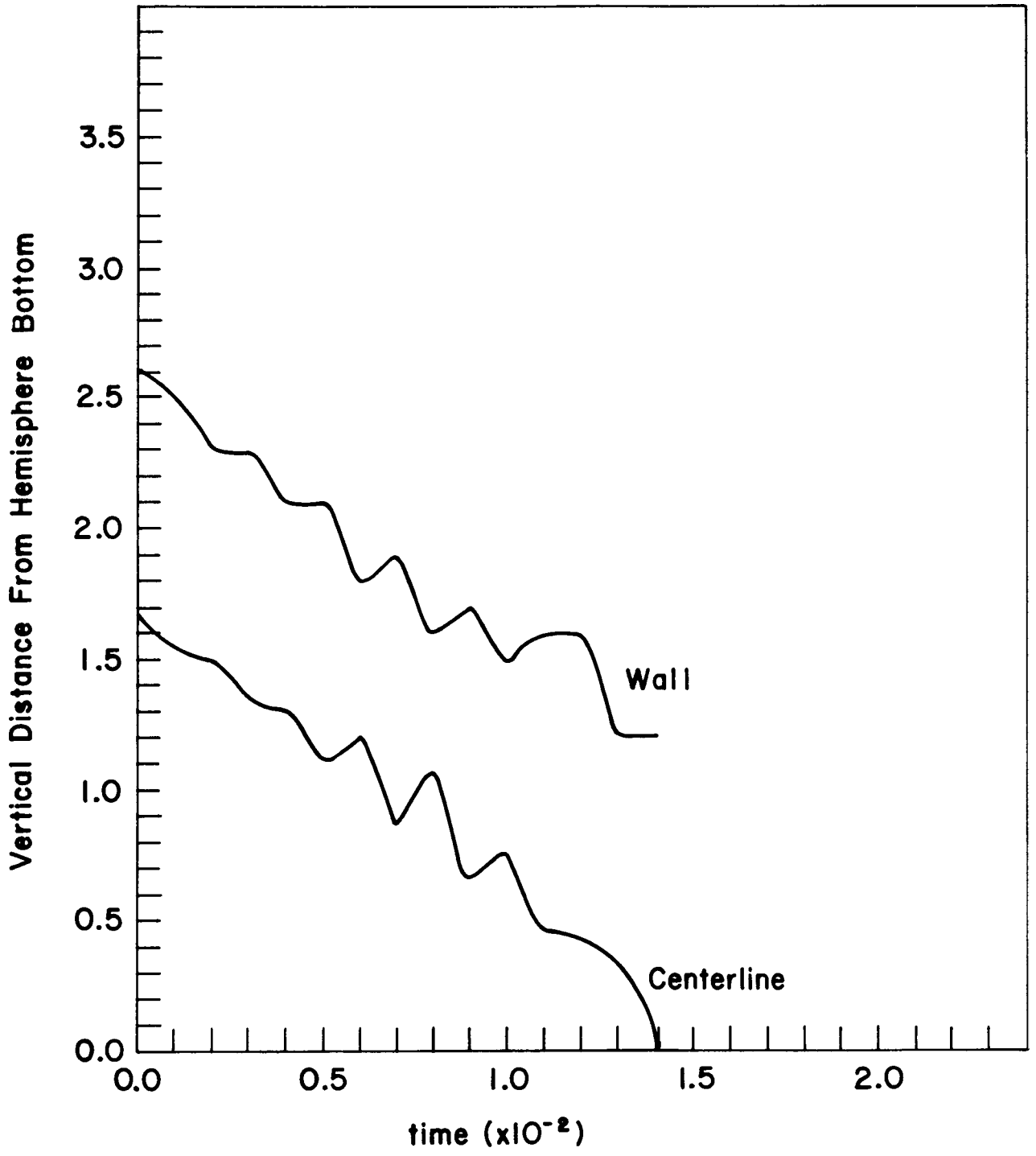


Fig. 7. Centerline and wall contact point histories for Case 2.

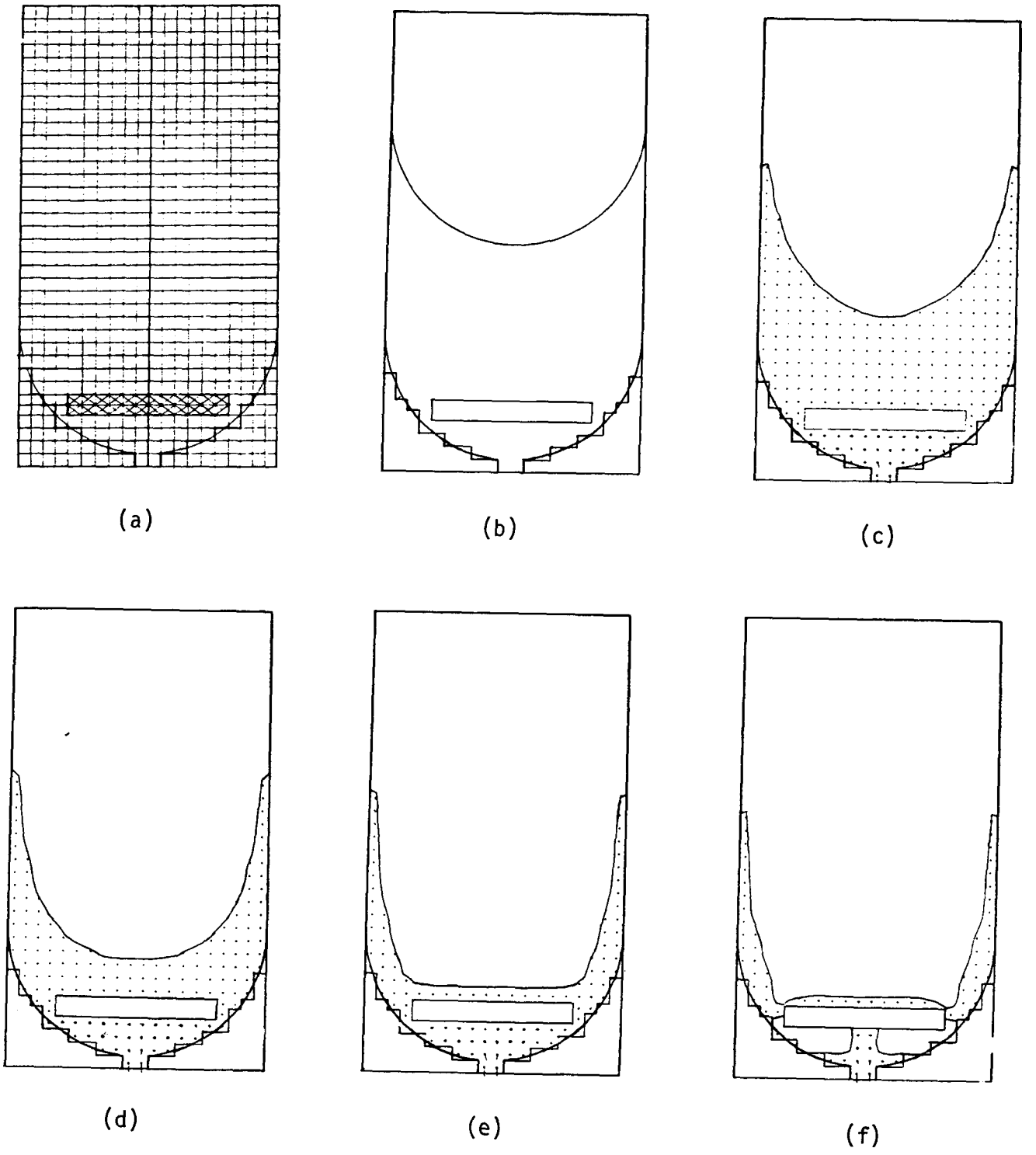


Fig. 8. Sequence of plots for Case 4 but typical for Cases 3 and 5. (a) Mesh used for baffled cases, (b) initial surface from ICON, (c) surface configuration at t = 30, (d) free surface at t = 70, (e) free surface at t = 100, (f) surface configuration just prior to complete vapor ingestion at t = 129 (Case 3), t = 117 (Case 4), and t = 112 (Case 5).

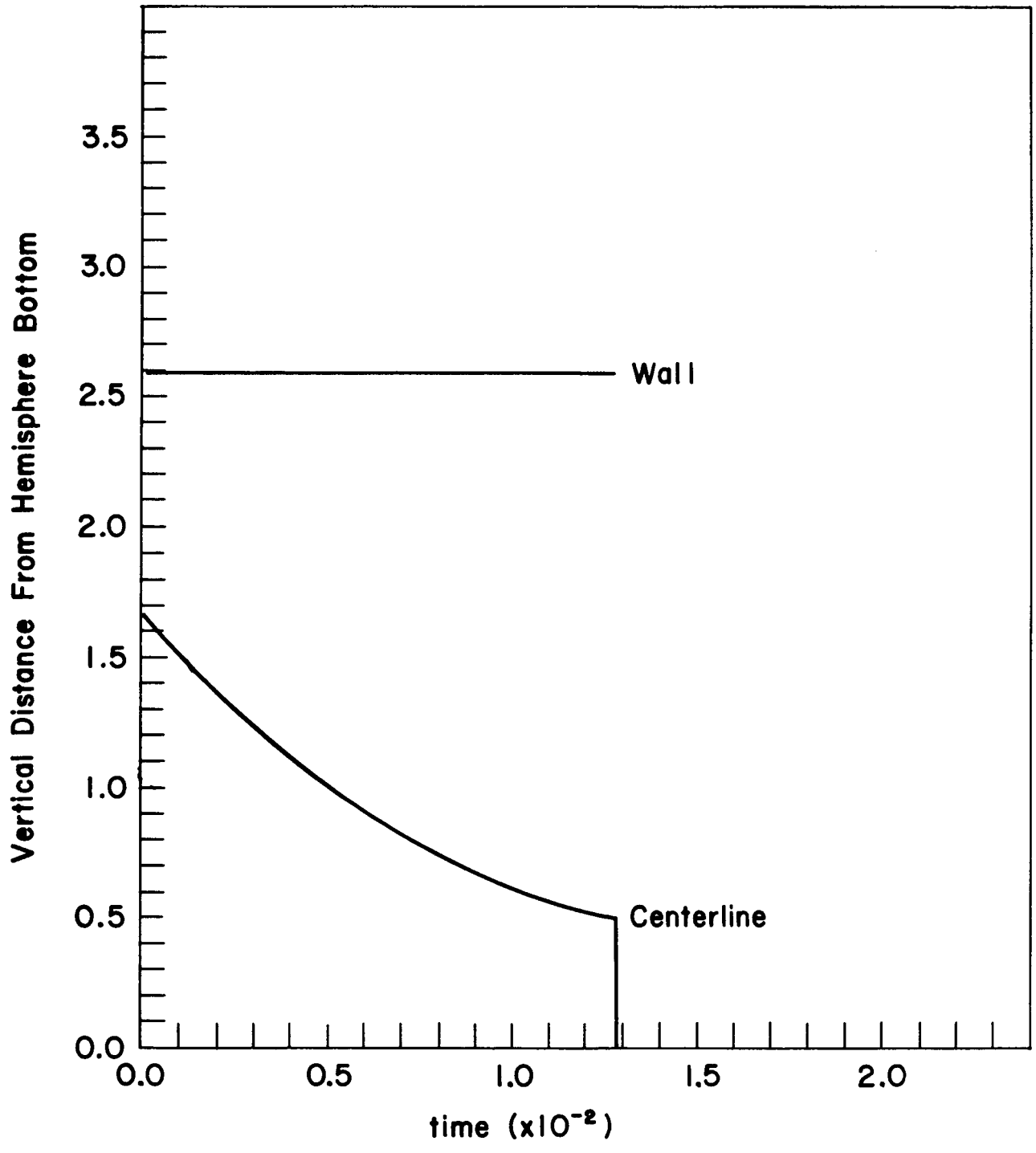


Fig. 9. Centerline and wall contact point histories for Case 3.

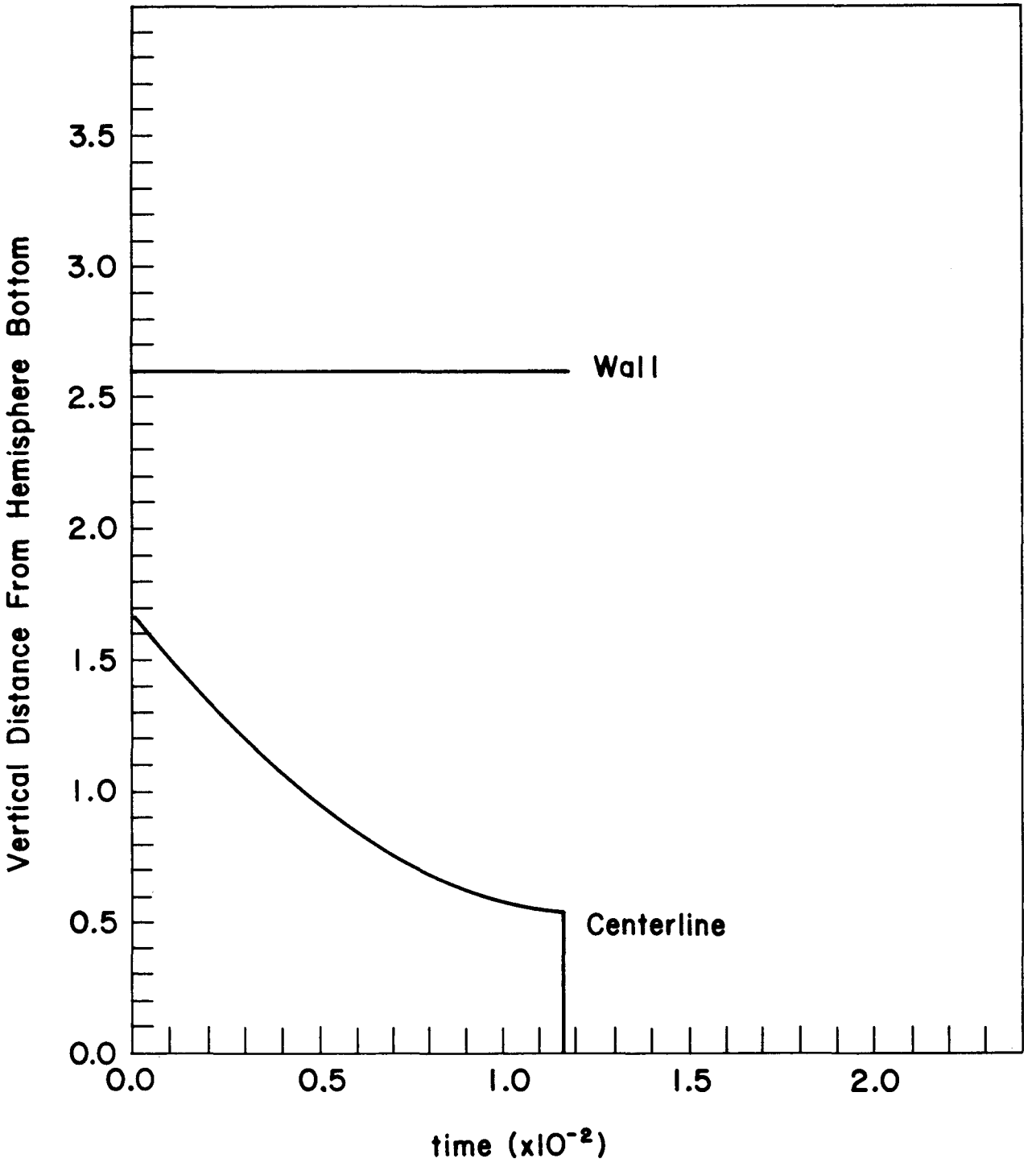


Fig. 10. Centerline and wall contact point histories for Case 4.

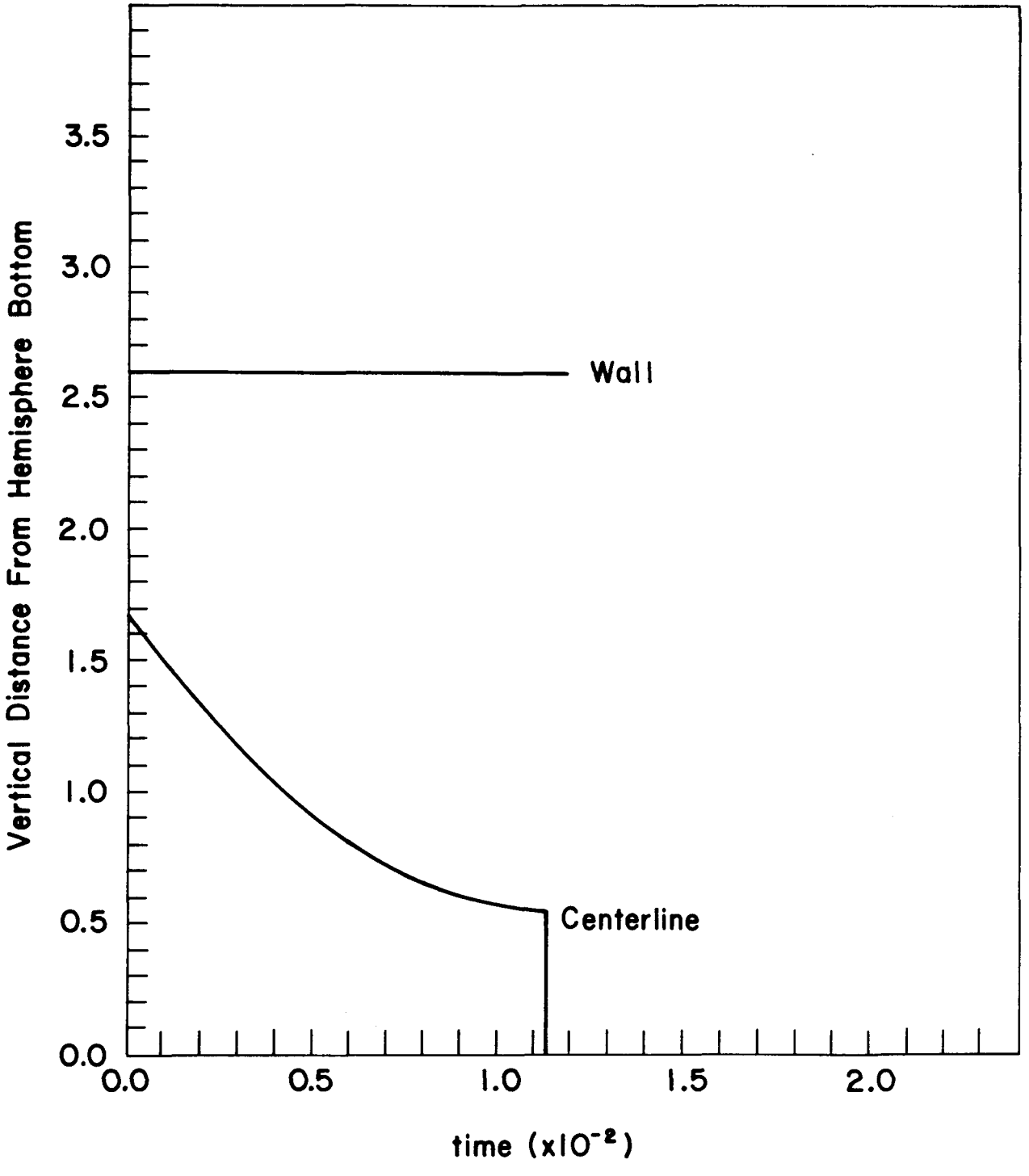


Fig. 11. Centerline and wall contact point histories for Case 5.

cells between obsacles to resolve the flow through such a region. However, in these cases, the flow dynamics are dominated by the tank outlet and a coarse resolution can be used with little error.

The residual volumes for these three cases are listed in Table III and plotted in Fig. 13. The experimental results are from [7]. The apparent lack of agreement between the calculated and experimental values is readily explained by the fact that a time delay in the experimental apparatus was included in the determination of the experimental values. When the true drain time is determined by subtracting the time delay of the experimental apparatus, multiplied by the volumetric flow rate and the product is subtracted from the initial volume, the experimental data points of Fig. 13 significantly shift to the right. Thus, a much better agreement is actually obtained than that presented.

VIII. SIMILITUDE OF TANK DRAINING PROBLEMS

Because all of the tank draining problems are performed nondimensionally, it is necessary to scale the results to obtain dimensional quantities.

The Weber number is customarily defined as $We = Q^2/\pi^2\sigma R^3$, in which

$Q = AV = \pi r^2 V$, the volumetric flow rate

R = the tank radius

V = the outflow velocity, and

r = the outlet radius.

Calculationally, (i.e., in the code) these quantities are all nondimensional and set to the predetermined values $R = 1.0$, $r = 0.1$, and $V = -1.0$. Thus, in this case, $We = .0001/\sigma$. Similarly, the Bond number $Bo = gR^2/\sigma$ becomes $Bo = g/\sigma$.

We define a nondimensional quantity X as the dimensional quantity X' divided by the scale of that quantity X_o, then $X' = X X_o$. Hence all that is needed to fully dimensionalize the calculational results are the length and time scales, L_o and T_o, respectively.

The length scale is determined by the radius of the tank being modeled (i.e., $L_o = R_o$); a time scale is then obtained from either the known outflow velocity $T_o = \frac{L_o V}{V'}$ or the known volumetric flow rate

$$T_o = \frac{L_o^3 Q}{Q'} = \frac{L_o^3 \pi (0.1)^2 1}{Q'} = \frac{.01 \pi L_o^3}{Q'} .$$

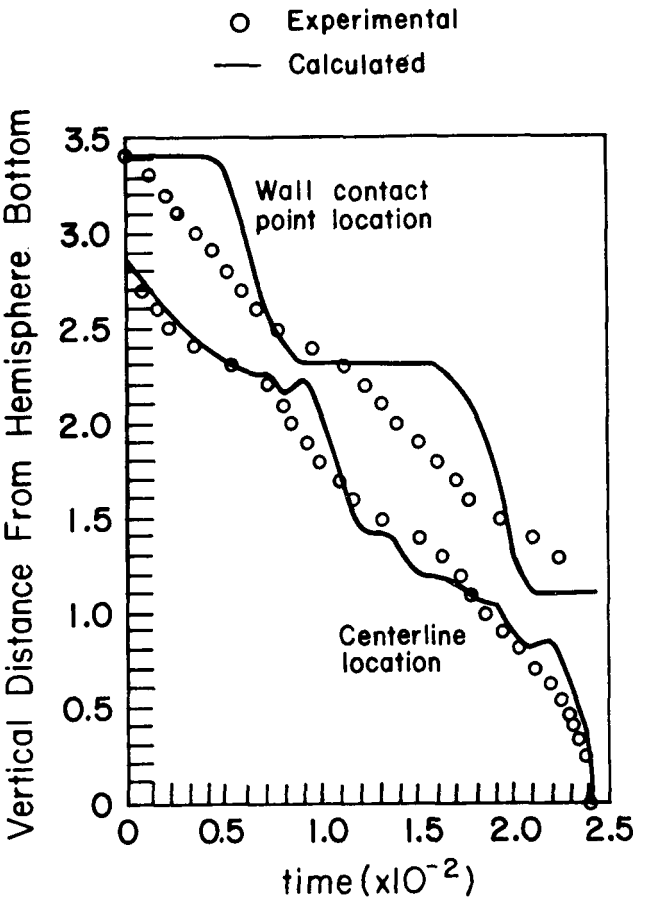


Fig. 12. Comparison of Case 1 results with the data of Symons [6].

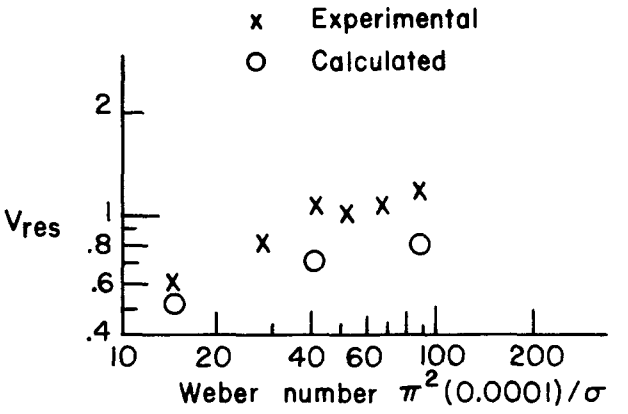


Fig. 13. Comparison of baffled draining results (Cases 3-5) with the data of Symons [7].

Now all quantities can be scaled to their appropriate values by

$$g' = g L_o T_o^{-2}$$
$$\sigma' = \sigma L_o^3 T_o^{-2}$$
$$Q' = .01 \pi L_o^3 T_o^{-1}$$
$$X' = X L_o$$

and so on.

Variables input to the code or variables output by the code should always be interpreted with this predetermined nondimensionality in mind.

ACKNOWLEDGMENTS

The author wishes to thank C. W. Hirt, T. D. Butler, and B. D. Nichols for many helpful discussions about the methodology used and results obtained with the NASA SOLA-VOF code.

This report is submitted in final fulfillment of the tasks outlined in the Los Alamos Scientific Laboratory proposal LASL P-938, March 1978, to the NASA Lewis Research Center.

REFERENCES

1. C. W. Hirt and B. D. Nichols, "Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries," J. Comp. Phys., submitted for publication.

2. 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 in preparation.

3. S. G. Berenyi and K. L. Abdalla, "Vapor Ingestion Phenomenon in Hemispherically Bottomed Tanks in Normal Gravity and in Weightlessness," NASA report NASA TN D-5704 (April 1970).

4. G. D. Bizzell and G. E. Crane, "Numerical Simulation of Low Gravity Draining," NASA report NASA CR-135004 LMSC-D521581 (May 1976).

5. E. P. Symons, "Contoured Tank Outlets for Draining of Cylindrical Tanks in Low-Gravity Environment," NASA Technical Paper 1492 (July 1979).

6. E. P. Symons, "Draining Characteristics of Hemispherically Bottomed Cylinders in a Low-Gravity Environment," NASA Technical Paper 1297 (August 1978).

7. E. P. Symons, "Outlet Baffles - Effort on Liquid Residuals from Zero-Gravity Draining of Hemispherically Ended Cylinders," NASA Technical Memorandum NASA TM X-2631 (September 1972).

APPENDIX A

VOLUME OF FLUID (VOF) METHOD FOR THE DYNAMICS OF FREE BOUNDARIES

VOLUME OF FLUID (VOF) METHOD FOR THE DYNAMICS OF FREE BOUNDARIES

C. W. Hirt and B. D. Nichols
Theoretical Division, Group T-3
University of California
Los Alamos Scientific Laboratory
Los Alamos, NM 87545

ABSTRACT

Several methods have been previously used to approximate free boundaries in finite-difference numerical simulations. In this paper 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 detailed 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 Sec. 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 Sec. III. In Sec. 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 Sec. V with various examples that show the strength of the VOF technique for treating problems involving highly complicated free surface flows.

Finally, in Sec. 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 forces may be computed, in a manner completely analogous to a Lagrangian calculation. The two methods differ, however, in the manner in which the fluid elements are moved to new 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 treat-

ed. 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,

$$\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} = v \qquad (1)$$

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 algorithms 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 in-

crease significantly with this method, because of the large increase in the number of point coordinates that must be stored. Surfaces are defined to lie 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 to contain 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, knowing both the normal direction and the value of F in a boundary cell, 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,

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

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 or pure Eulerian mesh, the flux of F moving with the fluid through a cell must be computed, but as noted in Sec. 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 Sec. IV (Subsec. 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 conditions 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, details are 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 Sec. 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, et al., have recently combined the SOLA-SURF code with a different interface tracking technique based on a VOF-like concept [12].

The following subsections provide details 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.

A. Outline

SOLA-VM uses an Eulerian mesh of rectangular cells having variable sizes, δx_i for the i^{th} column and δy_j for the j^{th} 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,

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

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

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, Eq. (3), must be supplemented with the incompressibility condition,

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

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

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

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, $(i\pm1,j)$ or $(i,j\pm1)$, 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:

(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. Details of these steps and boundary conditions are given in the following subsections.

B. Momentum Equation 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 i^{th} cell in the x-direction and j^{th} cell in the y-direction. Half integer subscripts refer to cell boundary locations. For example, $Q^n_{i,j+\frac{1}{2}}$, 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) is

$$u^{n+1}_{i+\frac{1}{2},j} = u^n_{i+\frac{1}{2},j} + \delta t \left[- \left(p^{n+1}_{i+1,j} - p^{n+1}_{i,j} \right) / \delta x_{i+\frac{1}{2}} + g_x - FUX - FUY + VISX \right]$$

$$v^{n+1}_{i,j+\frac{1}{2}} = v^n_{i,j+\frac{1}{2}} + \delta t \left[- \left(p^{n+1}_{i,j+1} - p^{n+1}_{i,j} \right) / \delta y_{j+\frac{1}{2}} + g_y - FVX - FVY + VISY \right] \quad (6)$$

Here $\delta x_{i+\frac{1}{2}} = 1/2(\delta x_i + \delta x_{i+1})$ and $\delta y_{j+\frac{1}{2}} = 1/2(\delta y_j + \delta y_{j+1})$. 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. Because the pressures at time level n+1 are not known at the beginning of the cycle, Eq. (6) cannot be used directly to evaluate (u^{n+1}, v^{n+1}), but must be combined with the continuity equation as described below. In the first step of a solution, therefore, the p^{n+1} in these equations are replaced by p^n to get a first guess for the new 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 approximations 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., $\nabla\cdot \underline{uu}$ instead of $\underline{u}\cdot\nabla\underline{u}$). Thus, FUX would be, for example, $\frac{\partial u^2}{\partial x}$ rather than $u\frac{\partial u}{\partial x}$. The divergence form was preferred in MAC because it provided a simple way to insure conservation of momentum in the difference approximations. This may be seen by considering the control volume used for $u_{i+\frac{1}{2},j}$ 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 one 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 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,

$$FUX = \left[u_{i+1,j} < u_{i+1,j} > - u_{i,j} < u_{i,j} >\right]/\delta x_{i+\frac{1}{2}} \quad , \tag{7}$$

where, e.g.,

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

$$\langle u_{i,j} \rangle = \begin{cases} u_{i-\frac{1}{2},j} & , \text{ if } u_{i,j} \geq 0 \\ u_{i+\frac{1}{2},j} & , \text{ if } u_{i,j} < 0 \end{cases} .$$

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

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

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 $\underline{u} \cdot \nabla \underline{u}$ 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

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

where

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

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

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

and where sgn(u) means the sign of $u_{i+\frac{1}{2},j}$. 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. The finite-difference form used for Eq. (5) is

$$\left(p^{n+1}_{i,j} - p^{n}_{i,j}\right)/(c^2\delta t) + D^{n+1}_{i,j} = 0 \qquad (10)$$

where

$$D^{n+1}_{i,j} = \left(u^{n+1}_{i+\frac{1}{2},j} - u^{n+1}_{i-\frac{1}{2},j}\right)/\delta x_i + \left(v^{n+1}_{i,j+\frac{1}{2}} - v^{n+1}_{i,j-\frac{1}{2}}\right)/\delta y_j$$
$$+ \xi \left(u^{n+1}_{i+\frac{1}{2},j} + u^{n+1}_{i-\frac{1}{2},j}\right)/(2x_i) \quad .$$

Since the velocities appearing in D are evaluated at the new time level, which depend on the n+1 level pressures according to Eq. (6), this equation is an implicit relation for the new pressures. A solution may be obtained by the following iterative process. The computational mesh is swept row by

row starting with the bottom row and working upward. In each cell containing fluid, but not a free surface, the pressure change needed to drive the left side of Eq. (10), call it S, toward zero is

$$\delta p = - S/(\partial S/\partial p) \tag{11}$$

where S is evaluated with the most updated values of p that are available, and the derivative is with respect to $p_{i,j}$. The new estimate for the cell pressure is then

$$P_{ij} + \delta p \tag{12}$$

and new estimates for the velocities located on the sides of the cell are

$$\begin{aligned} &u_{i+\frac{1}{2},j} + \delta t\ \delta p/\delta x_{i+\frac{1}{2}} \\ &u_{i-\frac{1}{2},j} - \delta t\ \delta p/\delta x_{i-\frac{1}{2}} \\ &v_{i,j+\frac{1}{2}} + \delta t\ \delta p/\delta y_{j+\frac{1}{2}} \\ &v_{i,j-\frac{1}{2}} - \delta t\ \delta p/\delta y_{j-\frac{1}{2}} \ , \end{aligned} \tag{13}$$

where the velocities appearing here are again the most updated values available.

A similar procedure is used in cells containing a free surface, except that the S used in Eq. (11) is not the left side of Eq. (10), but a relation that leads to the proper free surface boundary condition when driven to zero by the iteration [4]. The boundary condition is satisfied by setting the surface cell pressure ($p_{i,j}$) equal to the value obtained by a lin-

ear interpolation between the pressure wanted at the surface (p_s) and a pressure inside the fluid (p_N). 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. Then the S function giving this result is

$$S = (1 - \eta)\, p_N + \eta\, p_s - p_{i,j} \tag{14}$$

where $\eta = d_c/d$ is the ratio of the distance between the cell centers and the distance between the free surface and the center of the interpolation cell, see Fig. 4.

A complete iteration, therefore, consists of adjusting pressures and velocities in all cells occupied by fluid according to Eqs. (11-13), where S is given by Eq. (10) for an interior cell and by Eq. (14) for a surface cell. Convergence of the iteration is achieved when all cells have S values whose magnitudes are below some small number, ε. Typically, ε is of order 10^{-3}, although it can vary with the problem being solved and the units chosen for the problem.

In some cases, convergence of the iteration can be accelerated by multiplying δp from Eq. (11) by an over-relaxation factor ω. A value of ω that is often optimum is 1.8, but in no case should it exceed 2.0; otherwise an unstable iteration results.

In practice, the free surface condition, Eq. (14), leads to an over-relaxation type of instability when the interpolation factor ω is greater than one. Stability can be insured by under-relaxing the pressure variations in cells used as interpolation neighbors for surface cells. In

particular, the relaxation factor ω used in a cell acting as an interpolation neighbor for a surface cell must be replaced with

$$\frac{\omega}{1 - \omega(1-\eta)\delta t\ R} \tag{15}$$

where

$$R = \left(\frac{\partial S}{\partial P}\ \Delta x\ d_{\xi}\right)^{-1}.$$

Here η and d_{ξ} refer to the surface cell, while the S derivative is the value for the neighbor cell. Also Δx is δx of the surface cell if the neighbor lies in the x-direction, otherwise, Δx is equal to δy of the surface cell. The idea behind (15) is that the pressure change in the neighbor cell is coupled to the pressure in the surface cell, which in turn is dependent on the neighbor cell pressure through the linear interpolation, Eq. (14). To insure stability, this feedback type of coupling of the surface cell on its neighbor cell can be algebraically computed and used to define the stable relaxation limit, Eq. (15).

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

$$\frac{\partial F}{\partial t} + \frac{1}{r}\frac{\partial rFu}{\partial x} + \frac{\partial Fv}{\partial y} = 0 \quad , \tag{16}$$

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 (16), which is in divergence

form, is here more convenient for numerical approximation and is the form used in the following discussion. When Eq. (16) 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 and 14-15).

The basic 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. The total flux of fluid volume and void volume crossing this 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

$$\delta F = \text{MIN} \{ F_{AD} \, |V_x| + CF \, , \, F_D \, \delta x_D \} \tag{17}$$

and where

$$CF = \text{MAX} \{ (1.0 - F_{AD}) \, |V_x| - (1.0 - F_D) \, \delta x_D, \, 0.0 \}$$

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. (17) 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. (17). 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,

$$F = F_D \, |V_x| \quad ,$$

in which the F value in the donor cell is used to define the fractional area of the cell face fluxing fluid, see Fig. 5b. As discussed in Sec. IV.F, numerical stability requires that $|V_x|$ 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. (17). In case (d) of Fig. 5, more fluid than the amount $F_A|V_x|$, 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. (17).

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. (16) 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 like zero and one because roundoff errors 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 then 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

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

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

$$\left(\frac{dY}{dx}\right)_i = 2(Y_{i+1} - Y_{i-1})/(\delta x_{i+1} + 2\delta x_i + \delta x_{i-1}) \quad . \tag{18}$$

A similar calculation can be made for dX/dy,

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

and

$$\left(\frac{dX}{dy}\right)_j = 2(X_{j+1} - X_{j-1})/(\delta y_{j+1} + 2\delta y_j + \delta y_{j-1}) \quad . \tag{19}$$

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 Sec. 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. (18)-(19), and from this an effective surface tension pressure, p_s, to be applied at the surface according to Eq. (14).

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. Consider, for example, the left boundary. If this is a rigid free-slip wall, the normal velocity there must be zero and the tangential velocity should have no normal gradient, i.e.,

$$\left.\begin{aligned} u_{1,j} &= 0 \\ v_{1,j} &= v_{2,j} \\ p_{1,j} &= p_{2,j} \\ F_{1,j} &= F_{2,j} \end{aligned}\right\} \text{ for all } j \ .$$

If the left boundary is a no-slip rigid wall, then the tangential velocity component at the wall should also be zero, i.e.,

$$\left.\begin{aligned} u_{1,j} &= 0 \\ v_{1,j} &= -v_{2,j} \\ p_{1,j} &= p_{2,j} \\ F_{1,j} &= F_{2,j} \end{aligned}\right\} \text{ for all } j \ .$$

These conditions are imposed on the velocities computed from the momentum equations and after <u>each</u> pass through the mesh during the pressure iteration.

Continuative or outflow boundaries always pose a problem for low-speed calculations, because whatever prescription is chosen can potentially affect the entire flow field. What is needed is a prescription that permits fluid to flow out of the mesh with a minimum of upstream influence. In SOLA-VOF, the continuative boundary conditions used at the left wall are

$$\left.\begin{aligned} u_{1,j} &= u_{2,j} \\ v_{1,j} &= v_{2,j} \\ p_{1,j} &= p_{2,j} \\ F_{1,j} &= F_{2,j} \end{aligned}\right\} \text{ for all } j \ .$$

These conditions, however, are only imposed after applying the momentum equations and not after each pass through the pressure iteration.

For periodic boundary conditions in the x-direction, the left and right boundaries must be set to reflect the periodicity. This is easiest when the period length is chosen equal to the distance from the left wall to the left boundary of the last interior cell in the mesh at the right side. That is, two columns of cells, i=IMAX and i=IMAX-1, are reserved on the right side of the mesh for the setting of periodic boundary conditions. The conditions are then, on the left

$$\left.\begin{aligned} u_{1,j} &= u_{IM2,j} \\ v_{1,j} &= v_{IM2,j} \\ v_{2,j} &= v_{IM1,j} \\ p_{2,j} &= p_{IM1,j} \\ F_{2,j} &= F_{IM1,j} \end{aligned}\right\} \quad \text{for all } j$$

and on the right

$$\left.\begin{aligned} u_{IM1,j} &= u_{2,j} \\ v_{IMAX,j} &= v_{3,j} \end{aligned}\right\} \quad \text{for all } j \ ,$$

where IM1 = IMAX-1 and IM2 = IMAX-2. In this case, these conditions are imposed on velocities computed from the explicit momentum equations and after each pressure iteration.

A constant pressure boundary condition at the left wall is set by keeping the pressure in column i=2 constant and otherwise treating the boundary as continuative.

Boundary conditions similar to those for the left wall are used at the right, top, and bottom boundaries of the mesh. Of course, the normal and tangential velocities at the top and bottom boundaries are v and u, respectively.

For convenience, the SOLA-VOF code has been written so that any of the above boundary conditions can be automatically imposed by setting input numbers. To increase the usefulness of the basic code, specified inflow and outflow boundaries and internal obstacles within the fluid region are easily incorporated. In the case of obstacles that are restricted to shapes constructed by blocking out cells of the computing mesh, semi-automatic rigid wall boundary conditions are included in the SOLA-VOF code. For this purpose, an array used to store relaxation factors for the pressure iteration, $BETA_{i,j}$, is also used to flag obstacle cells. In particular, because legitimate relaxation factors must be positive numbers, a negative value (say -1.0) serves as a flag. The flag values must be programmed into the setup section of the code for each application. Thereafter, the code automatically eliminates computations for all momentum and continuity equations in flagged obstacle cells. Boundary conditions for normal velocities, pressures, and the volume of fluid function are automatically set in the main boundary condition section of the code. Because all velocity components within obstacles are set to zero, no-slip tangential velocity conditions are only first-order accurate. That is, tangential ve-

locities are zero at locations shifted from the actual boundary by one-half of a cell width.

Specified inflow and outflow conditions at mesh boundaries or at locations within the mesh must be programmed into the boundary condition section of the code. A special location is reserved in the SOLA-VOF code for this purpose at the end of the regular boundary condition section.

2. Free Surface Boundaries. The free surface boundary condition for normal stress is automatically satisfied by the implicit pressure calculation using Eq. (14). This condition must be supplemented with the specification of velocities immediately outside the surface, where these values are needed in the finite-difference approximations for points inside the surface. The specifications used in SOLA-VOF are identical to those used in many earlier Marker-and-Cell codes. Velocities must be set on every cell boundary between a surface cell and an empty cell. If the surface cell has only one neighboring empty cell, the boundary velocity is set to insure the vanishing of $D_{i,j}$, the velocity divergence defined in Eq. (10). When there are two or more empty cell neighbors, the individual contributributions to the divergence, $\frac{1}{r}\frac{\partial ru}{\partial r}$ and $\frac{\partial v}{\partial y}$, are separately set to zero. In some cases, it is also necessary to assume zero values for $\frac{\partial u}{\partial y}$ or $\frac{\partial v}{\partial x}$. These latter conditions are additionally used to set exterior tangent velocities to a free surface on boundaries between empty cells adjacent to a surface cell.

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 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 insure 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. 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 5 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.

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 5 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.

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 downwards 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 is 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.

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 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 is a result 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. Nevertheless, 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, "An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds," J. Comp. Phys. 14, 227 (1974).

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, "Calculating Three-Dimensional Free Surface Flows in the Vicinity of Submerged and Exposed Structures," J. Comp. Phys. 12, 234 (1973).

4. B. D. Nichols and C. W. Hirt, "Improved Free Surface Boundary Conditions for Numerical Incompressible Flow Calculations," J. Comp. Phys. 8, 434 (1971).

5. B. D. Nichols and C. W. Hirt, "Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies," Proc. First Intern. Conf. Num. Ship Hydrodynamics, Gaithersburg, MD, Oct. 1975.

6. F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow," Phys. Fluids 8, 2182 (1965); 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, "Relativistic Fluid Dynamics Calculations with the Particle-in-Cell Technique," J. Comp. Phys. 20, 119 (1976).

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, "A Computer Study of Finite Amplitude Water Waves," J. Comp. Phys. 6, 68 (1970).

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, "Adding Limited Compressibility to Incompressible Hydrocodes," J. Comp. Phys., to be published.

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, in preparation.

17. C. W. Hirt, "Heuristic Stability Theory for Finite-Difference Equations," J. Comp. Phys. 2, 339 (1968).

18. J. C. Martin and W. J. Moyce, Phil. Trans. Roy. Soc. London A244, 312 (1952).

19. J. J. Stoker, Water Waves, Interscience, New York, NY (1957).

20. F. H. Harlow and J. E. Welch, "Numerical Study of Large-Amplitude Free-Surface Motions," Phys. Fluids 9, 842 (1966).

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, "Numerical Simulation of BWR Vent-Clearing Hydrodynamics," Nuc. Sci. and Eng., accepted for publication.

Figure Captions

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

2. Location of variables in a typical mesh cell.

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

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

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.

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 = 1/2 contour line, which is why the top right corner at t = 0.0 is not 90°.

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

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

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

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

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.

12. Schematic of MIT single vent test apparatus.

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

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

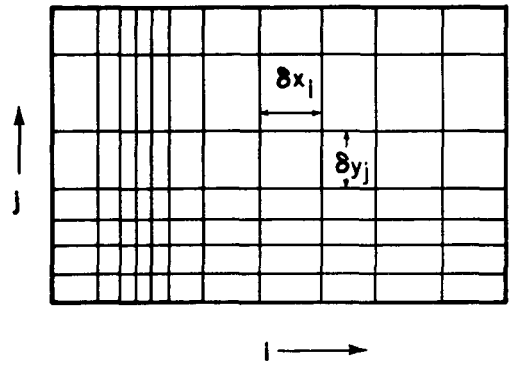


Fig. 1.

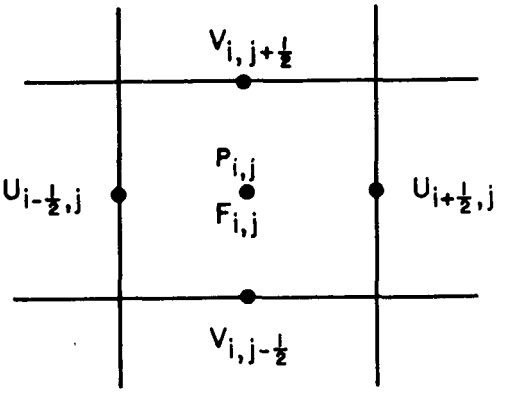


Fig. 2.

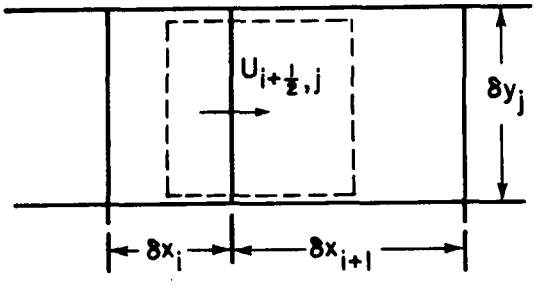


Fig. 3.

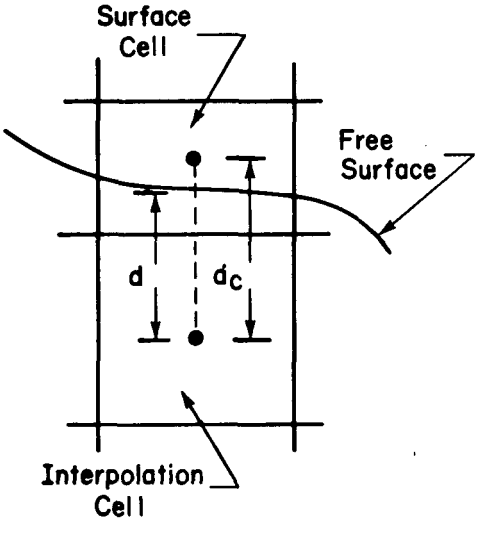


Fig. 4.

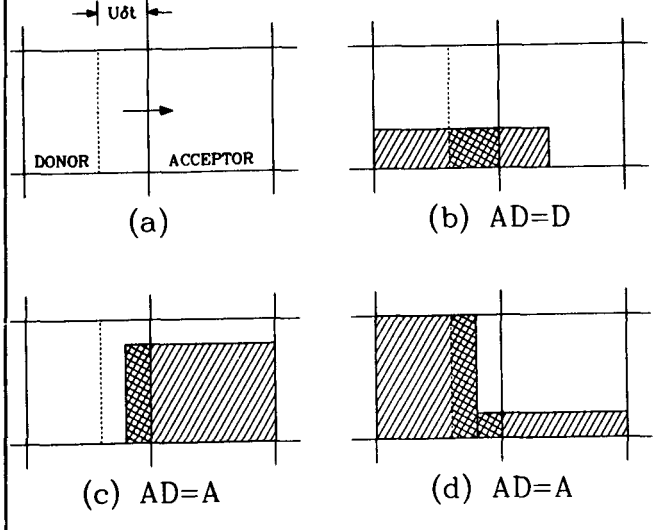


Fig. 5.

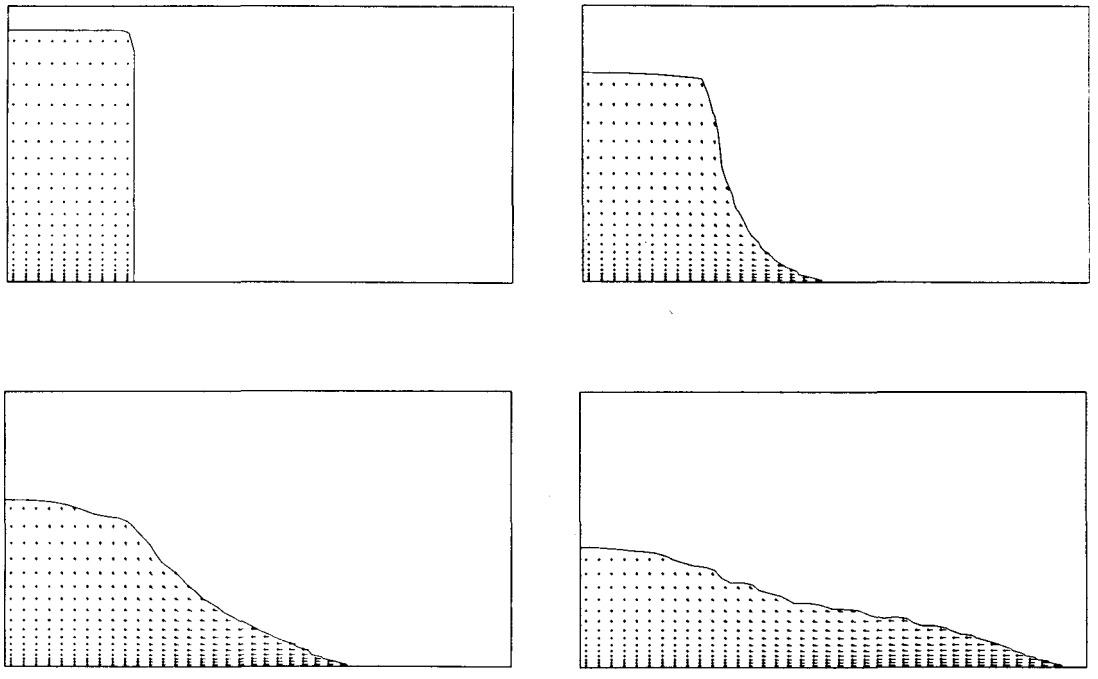


Fig. 6.

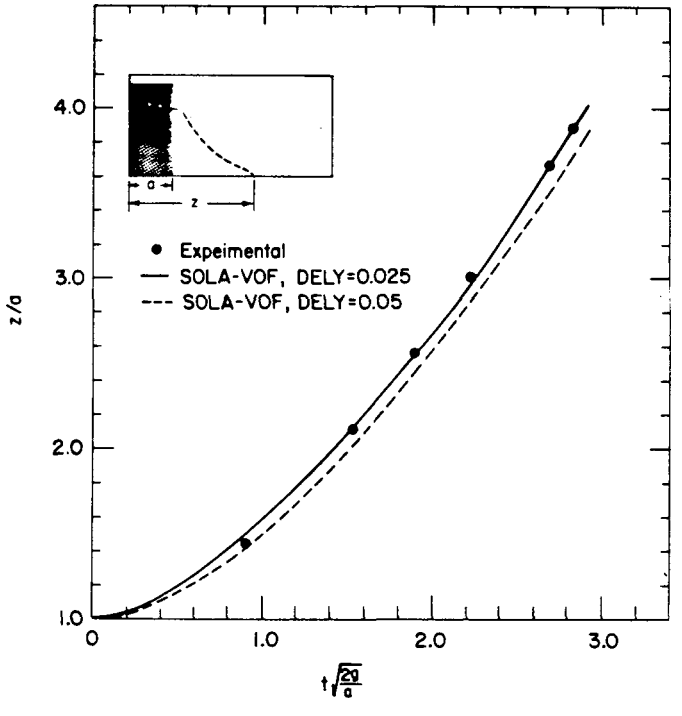

4.0
3.0
2.0
1.0
0
1.0
2.0
3.0
z/a
a
z
Expeimental
SOLA-VOF, DELY=0.025
SOLA-VOF, DELY=0.05

Fig. 7.

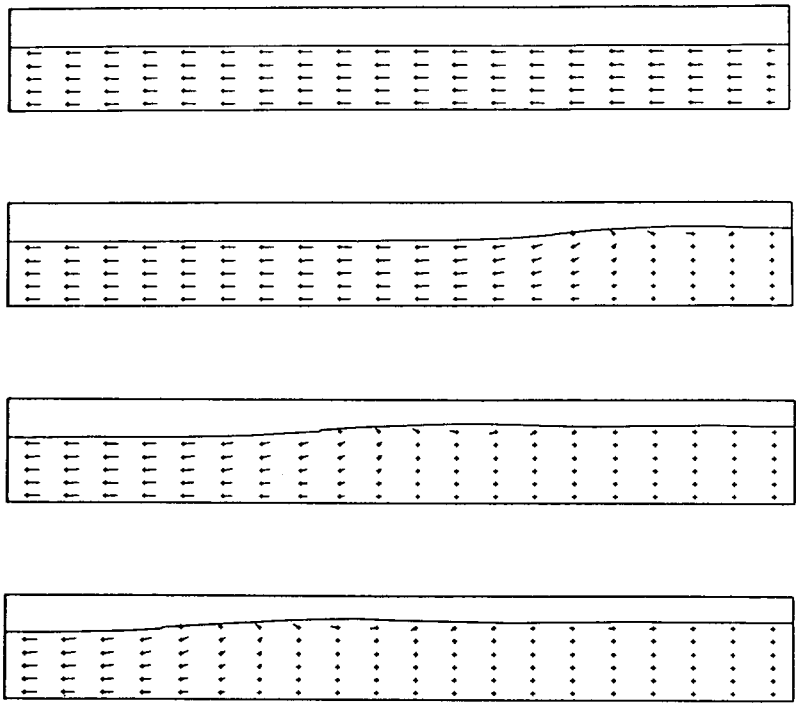


Fig. 8.

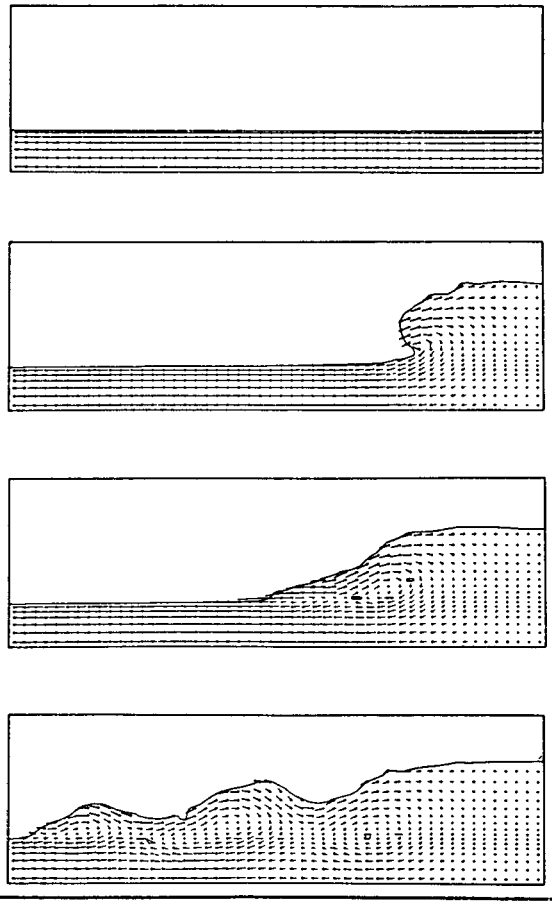


Fig. 9.

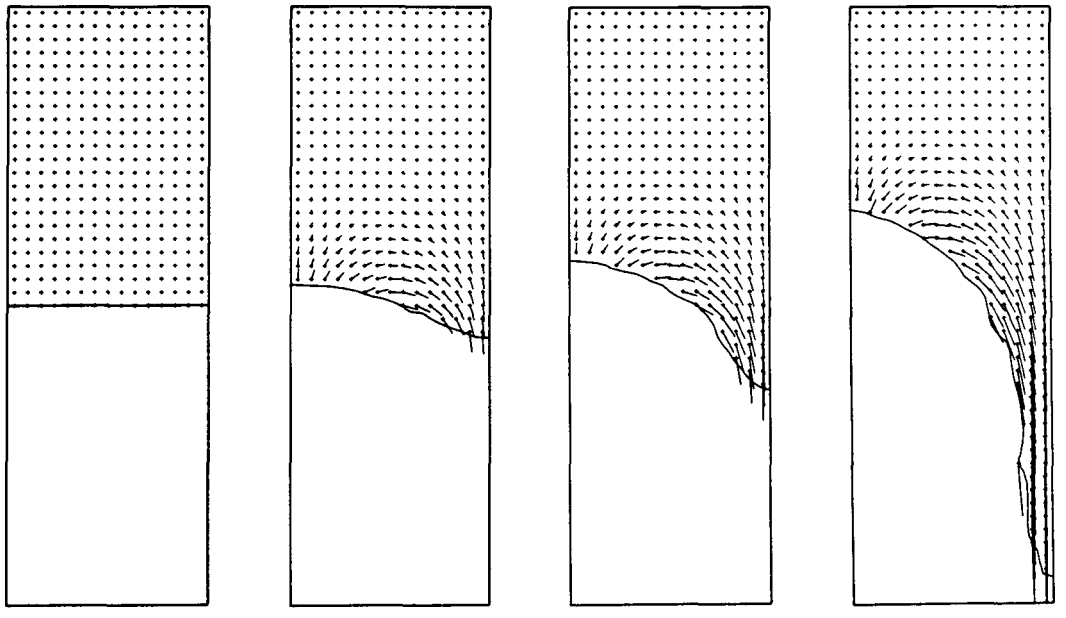


Fig. 10.

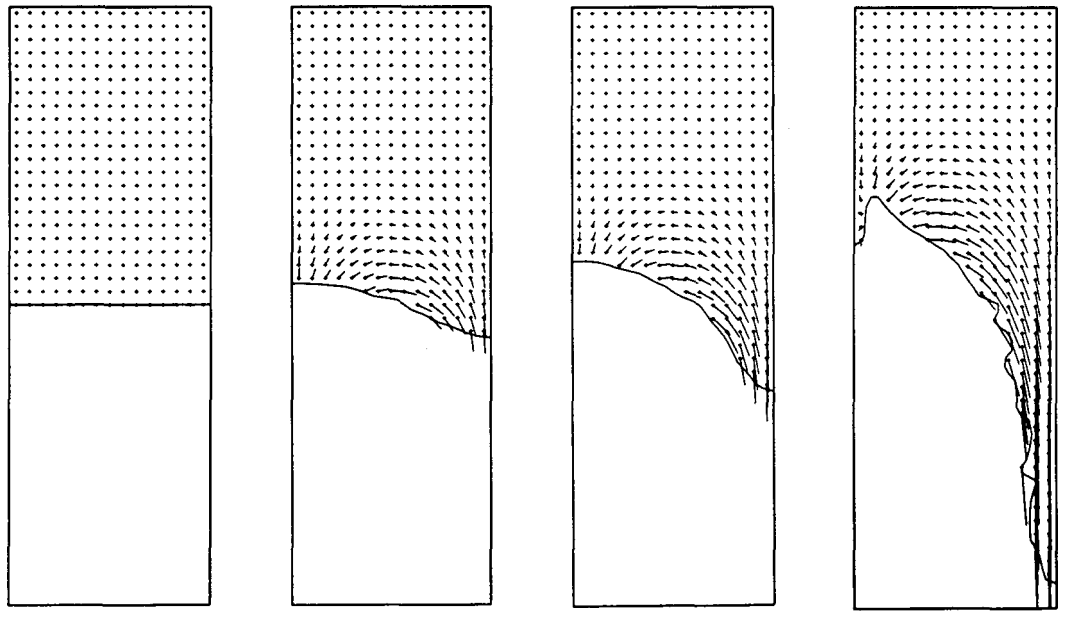


Fig. 11.

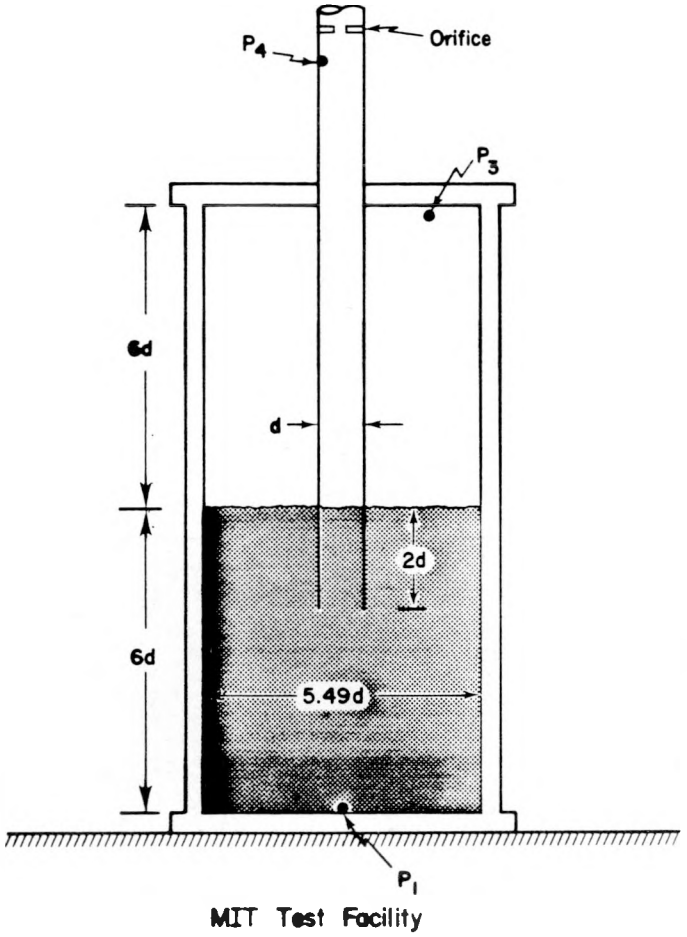


Fig. 12.

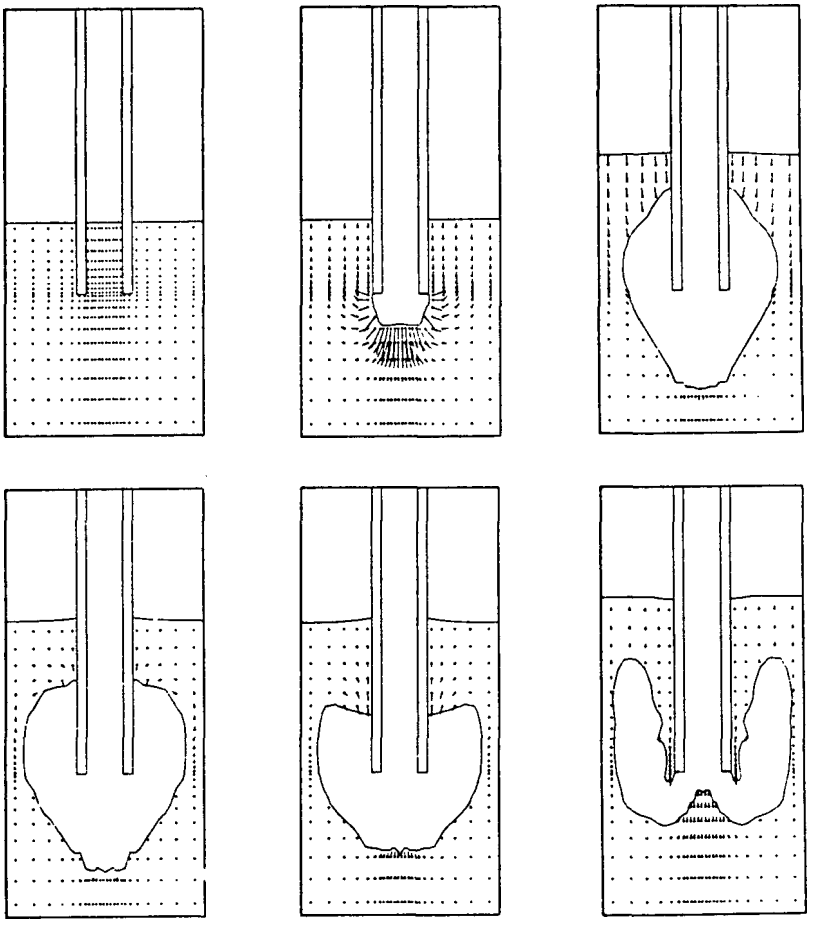


Fig. 13.

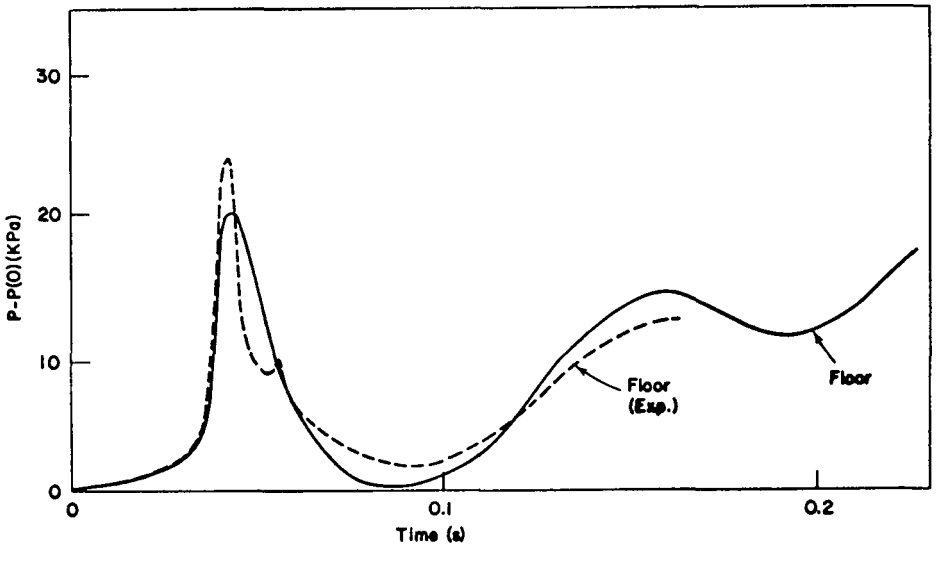


Fig. 14.

LASL Identification
SOLA VOF LP No.288

APPENDIX B

FORTRAN LISTING OF THE NASA SOLA-VOF COMPUTER CODE

```
C**$$**$$**BEGINNING OF ELEMENT COMDECK/            **$$**$$**C
COMDECK  PROC
      PARAMETER  IBAR2=13,JBAR2=38,NPRTS=1,MSHX=4,MSHY=4
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C     ----- BEGIN COMDECK TYPE       -----
      REAL Z, XO1, YO1, XO2, YO2
      DOUBLE PRECISION LONG, NU, NORMX, NORMY
      INTEGER CYCLE, WL, WR, WT, WB
C     ----- END COMDECK TYPE         -----
C     ----- BEGIN COMDECK COMMON1    -----
C
      COMMON /TD/ ACOM(1), UN(IBAR2,JBAR2), VN(IBAR2,JBAR2), P(IBAR2
     1 ,JBAR2), F(IBAR2,JBAR2), NF(IBAR2,JBAR2), PETA(IBAR2,JBAR2), XP
     2 (NPRTS), YP(NPRTS), IP(NPRTS), JP(NPRTS), TWPRT, TWPLT, NP, NTD,
     3 TWTD, PS(IBAR2,JBAR2)
C
      COMMON /GEN/ U(IBAR2,JBAR2), V(IBAR2,JBAR2), D(IBAR2,JBAR2), BETA
     1 (IBAR2,JBAR2), FN(IBAR2,JBAR2), X(IBAR2), XI(IBAR2), RXI(IBAR2),
     2 DELX(IBAR2), RDX(IBAR2), RX(IBAR2), Y(JBAR2), DELY(JBAR2), RDY
     3 (JBAR2), YJ(JBAR2), RYJ(JBAR2),  NAME(20),
     4  XL(MSHX), XC(MSHX), XR(MSHX), DXMN(MSHX), NXL(MSHX),
     5 NXR(MSHX), YL(MSHY), YC(MSHY), YR(MSHY), DYMN(MSHY), NYL(MSHY),
     6 NYR(MSHY)
      COMMON /GEN/ IBAR, JBAR, DELT, NU, CYL, EPSI, DZRO, GX, GY, UI, VI
     1 , VELMX, TWFIN, PRTDT, PLTDT, OMG, ALPHA, WL, WR, WB, WT, PARTN,
     2 CWTD, TRST, T, CYCLE, IMAX, JMAX, IM1, JM1, IM2, JM2, EMF, EMF2,
     3 EM4, ITER, EM6, NKX, NKY, IREP, VCHGT, GYI, AUTOT, FLG, PDC, JTMN
     4 , PdW, EMF1, FLHT, ISYMPL, WEBER, BOND, SIGMA, PI, ISRF10, CANGLE
     5 , TANCA, RPD, YCENTR, VOUT, IBAFF, EP8, EP9, EP10, EM10, DP1
      COMMON /FLM/ Z(200)
      COMMON /MOVIE/ MOVY, DTMVP, VELMX1, UVEC, VVEC, XCC, YCC, JOBTP,
     1 IOBLF, IOBRG, JOBBT, XSHFT, YSHFT, XTR, YTR, X8R, Y8R, XTL, YTL,
     2 XMIN, XMAX, D1, D2, D3, SF, YMIN, YMAX
      DIMENSION Q(IBAR2,JBAR2)
      EQUIVALENCE (UN,XPLT), (VN,YPLT), (Q,FN)
C     ----- END COMDECK COMMON1    -----
 END
C**$$**$$**BEGINNING OF ELEMENT SOLAVOF/            **$$**$$**C
      INCLUDE COMDECK,LIST
C
C     SOLA-VOF, VARIABLE MESH ****************************************
C
      NAMELIST /XPUT/ DELT,NU,CYL,EPSI,DZRO,GX,GY,UI,VI,VELMX,TWFIN
     1 ,PRTDT,PLTDT,OMG,ALPHA,WL,WR,WT,WB,PARTN,CWTD,TRST,MOVY,DTMVP
     2 ,AUTOT,FLHT,ISYMPL,WEBER,BOND,ISRF10,CANGLE,VOUT,IBAFF
      NAMELIST /MSHSET/ NKX,XL,XC,XR,NXL,NXR,DXMN,NKY,YL,YC,YR,NYL,NYR
     1 ,DYMN,YCENTR
C
C      DATA BLOCK FOR CODE - ALL ELEMENTS ARE IN COMMON
C
      DATA EMF /1.0D-06/, EM6 /1.0D-06/, EMF2 /2.5D-07/, EM4 /1.0D-04/
      DATA EP8 /1.0D+08/, EP9 /1.0D+09/, EP10 /1.0D+10/, EM10 /1.0D-10/
      DATA DP1 /1.0D0/
      DATA PI /3.14159265D0/, RPD /0.017453292D0/
      DATA CANGLE /5.0D0/
      DATA VOUT /-1.0D0/, IBAFF /0/
C
      CALL FILMST
```

```
C
      READ (5,210) NAME
      WRITE (6,200)
      WRITE (6,210) NAME
C
C     READ AND PRINT INITIAL INPUT DATA
C
      READ (5,XPUT)
C
C     * SPECIAL INPUT DATA, VARIABLE MESH, READ IN,CALCULATE AND PRINT
C
      READ (5,MSHSET)
C
      CALL MESHST
C
      WRITE (6,250) IBAR,JBAR,DELT,NU,CYL,EPSI,DZRO,GX,GY,UI,VI,VELMX
     1 ,TWFIN,PRTDT,PLTDT,OMG,ALPHA,WL,WR,WT,WB,PARTN,CWTD,TRST,MOVY
     2 ,DTMVP,AUTDT,FLHT,ISYMPL,WEBER,BOND,ISRF10,CANGLE,VOUT
      WRITE (6,290) NKX
      WRITE (17,200)
      WRITE (17,250) IBAR,JBAR,DELT,NU,CYL,EPSI,DZRO,GX,GY,UI,VI,VELMX
     1 ,TWFIN,PRTDT,PLTDT,OMG,ALPHA,WL,WR,WT,WB,PARTN,CWTD,TRST,MOVY
     2 ,DTMVP,AUTDT,FLHT,ISYMPL,WEBER,BOND,ISRF10,CANGLE,VOUT
      WRITE (6,290) NKX
      WRITE (17,290) NKX
      DO 10 I=1,NKX
      WRITE (6,300) I,XL(I),XC(I),XR(I),NXL(I),NXR(I),DXMN(I)
      WRITE (17,300) I,XL(I),XC(I),XR(I),NXL(I),NXR(I),DXMN(I)
   10 CONTINUE
      WRITE (6,310) NKY
      WRITE (17,310) NKY
      DO 20 I=1,NKY
      WRITE (6,300) I,YL(I),YC(I),YR(I),NYL(I),NYR(I),DYMN(I)
      WRITE (17,300) I,YL(I),YC(I),YR(I),NYL(I),NYR(I),DYMN(I)
   20 CONTINUE
C
      VINIT=2.0*PI/3.0
C
      CALL SETUP
C
      IF (CYCLE.GT.0) GO TO 30
C
      CALL BC
C
      GO TO 40
C
C     * START CYCLE
C
   30 CONTINUE
      IF (CYCLE.GT.20) OMG=1.0
      ITER=0
      FLG=1.0
      PMX=EM6
C
      CALL TILDE
C
      CALL BC
C
      CALL PRESIT
C
```

```
      IF (T.GT.EP9) GO TO 50
C
      CALL PARMOV
C
      CALL VFCONV
C
      CALL BC
C
   40 CONTINUE
C
      CALL PETACL
C
C     PRINT AND PLOT
C
      VOLUME=0.0
      DO 50 I=2,IM1
      DO 50 J=2,JM1
      IF (BETA(I,J).LT.0.0.OR.F(I,J).LT.EM6) GO TO 50
      IF (NF(2,J).NE.0.AND.F(2,J+1).LT.EM6) HCL=F(2,J)*DELY(J)+Y(J-1)
      IF (NF(IM1,J).EQ.2) HWALL=Y(J)
      IF (NF(IM1,J).EQ.3) HWALL=F(IM1,J)*DELY(J)+Y(J-1)
      VOLUME=VOLUME+F(I,J)*DELY(J)*DELX(I)*(2.0*PI*XI(I)*CYL+(1.0-CYL))
      IF (NF(I,J).NE.1.AND.NF(I,J).NE.2) GO TO 50
      QQ=2.0*DFLOAT(NF(I,J))-3.0
      VOLUME=VOLUME+CYL*QQ*PI*DELX(I)**2*DELY(J)*F(I,J)*(1.0-F(I,J))
   50 CONTINUE
      VOLBAR=VOLUME/VINIT
      WRITE (6,280) T,CYCLE,VOLUME,VOLBAR,HCL,HWALL
      WRITE (6,240) ITER,T,DELT,CYCLE,VCHGT
      IF (MOVY.EQ.1) GO TO 70
      WRITE (17,280) T,CYCLE,VOLUME,VOLBAR,HCL,HWALL
   60 CONTINUE
      IF (T.GT.0.) GO TO 70
      WRITE (17,240) ITER,T,DELT,CYCLE,VCHGT
   70 CONTINUE
      IF (CYCLE.LE.0) GO TO 80
      IF (T+EM6.LT.TWPLT) GO TO 110
      TWPLT=TWPLT+PLTDT
   80 CONTINUE
      IF (MOVY.EQ.1) GO TO 100
      CALL PAGEG (7,0,0,1)
      WRITE (17,270) NAME
      WRITE (17,240) ITER,T,DELT,CYCLE,VCHGT
      WRITE (17,220)
      DO 90 I=1,IMAX
      DO 90 J=1,JMAX
      WRITE (17,230) I,J,J(I,J),V(I,J),P(I,J),D(I,J),PS(I,J),F(I,J),NF(I
     1 ,J),PETA(I,J)
   90 CONTINUE
  100 CONTINUE
      CALL DRAW
  110 CONTINUE
      IF (CYCLE.LE.0) GO TO 120
      IF (T+EM6.LT.TWPRT) GO TO 140
      TWPRT=TWPRT+PRTDT
  120 CONTINUE
      WRITE (6,270) NAME
      WRITE (6,240) ITER,T,DELT,CYCLE,VCHGT
      WRITE (6,260)
      WRITE (6,220)
```

```
      DO 130 I=1,IMAX
      DO 130 J=1,JMAX
      WRITE (6,230) I,J,U(I,J),V(I,J),P(I,J),D(I,J),PS(I,J),F(I,J),NF(I
     1 ,J),PETA(I,J)
  130 CONTINUE
  140 CONTINUE
C
C     SET THE ADVANCE TIME ARRAYS INTO THE TIME -N ARRAYS
C
      DO 150 I=1,IMAX
      DO 150 J=1,JMAX
      UN(I,J)=U(I,J)
      VN(I,J)=V(I,J)
      U(I,J)=0.0
      V(I,J)=0.0
      D(I,J)=0.0
      FN(I,J)=F(I,J)
  150 CONTINUE
C
C     ADJUST DELT
C
      IF (AUTOT.LT.0.5) GO TO 180
      DUMX=EM10
      DVMX=EM10
      DELTN=DELT
      DO 160 I=2,IM1
      DO 160 J=2,JM1
      UDM=DABS(UN(I,J))/(XI(I+1)-XI(I))
      VDM=DABS(VN(I,J))/(YJ(J+1)-YJ(J))
      DUMX=DMAX1(DUMX,UDM)
      DVMX=DMAX1(DVMX,VDM)
  160 CONTINUE
      DTMP=1.01D0
      IF (ITER.GT.25) DTMP=0.99D0
      DELTD=DELT*DTMP
      CON=0.25
      IF (CYCLE.GT.10) CON=0.45D0
      DELT=DMIN1(DELTD,CON/DUMX,CON/DVMX)
      IF (MOVY.GT.0) DELT=DMIN1(DELT,DTMVP)
      DTRA=DELTN/DELT
      DO 170 I=1,IMAX
      DO 170 J=1,JMAX
      IF (BETA(I,J).LT.0.0) GO TO 170
      BETA(I,J)=BETA(I,J)*DTRA
  170 CONTINUE
  180 CONTINUE
C
C     ADVANCE TIME T=T+DELT
C
      T=T+DELT
      IF (DELT.LT.EM6) T=EP8
      IF (T.GT.TWFIN) GO TO 190
      CYCLE=CYCLE+1
      IF (FN(2,3).GT.0.100.OR.VOUT.GT.0.0) GO TO 30
      TWPLT=T
      TWFIN=T
      GO TO 30
  190 CONTINUE
      CALL EXITG (7)
      CALL EXIT
```

```
241 C
242    200 FORMAT (1H1)
243    210 FORMAT (20A4)
244    220 FORMAT (4X,1HI,5X,1HJ,9X,1HU,14X,1HV,15X,1HP,15X,1HD,12X,2HPS,13X,
245       1 1HF,11X,2HNF,9X,4HPETA)
246    230 FORMAT (2X,I3,3X,I3,6(3X,1PD12.5),3X,I3,3X,D12.5)
247    240 FORMAT (6X,6HITER= ,I5,5X,6HTIME= ,1PD12.5,5X,6HDELT= ,1PD12.5,5X,
248       1 7HCYCLE= ,I4,5X,7HVCHGT= ,1PD12.5)
249    250 FORMAT (1H ,5X,6HIBAR= ,I4/6X,6HJBAR= ,I4/6X,6HDELT= ,1PD12.5/8X,4
250       1 HNU= ,D12.5/7X,5HCYL= ,D12.5/6X,6HEPSI= ,D12.5/6X,6HDZRO= ,D12.5/
251       2 8X,4HGX= ,D12.5/8X,4HGY  ,D12.5/8X,4HUI= ,D12.5/8X,4HVI= ,D12.5/5
252       3 X,7HVELWX= ,D12.5/5X,7H WFIN= ,D12.5/5X,7HPRTDT= ,D12.5/5X,7HPLTD
253       4T= ,D12.5/7X,5HOMG= ,D12.5/5X,7HALPHA= ,D12.5/8X,4HWL= ,I4/8X,4HWR
254       5= ,I4/8X,4HWT= ,I4/8X,4HWB= ,I4/5X,7HPARTN= ,D12.5/5X,6HCWTD= ,D12
255       6 .5/5X,6HTRST= ,D12.5/6X,6HMOVY= ,D12.5/5X,7HDTMVP= ,D12.5/5X,7HAU
256       7TOT= ,D12.5/6X,6HFLHT= ,D12.5/3X,9HISYMPLT= ,I4/5X,7HWEBER= ,D12.5
257       8 /6X,6HBOND= ,D12.5/3X,9HISURF10= ,I4/4X,8HCANGLE= ,D12.5/6X,6HVOU
258       9T= ,D12.5)
259    260 FORMAT (1HC)
260    270 FORMAT (1H ,18X,20A4,1X,A10,2(1X,A8))
261    280 FORMAT (2X,6HTIME= ,1PD12.5,3X,7HCYCLE= ,I6,3X,8HVOLUME= ,1PD12.5,
262       1 3X,8HVOLBAR= ,1PD12.5,3X,5HHCL= ,1PD12.5,3X,7HHWALL= ,1PD12.5)
263    290 FORMAT (2X,5HNKX=,I4)
264    300 FORMAT (2X,5HMESH= ,I4,3X,3HL= ,1PD12.5,3X,3HC= ,D12.5,3X,3HR=
265       1 ,D12.5,3X,4HNL= ,I4,3X,4HNR= ,I4,3X,5HDMN= ,D12.5)
266    310 FORMAT (2X,5HNKY= ,I4)
267        END
```

```
C**$***$***BEGINNING OF ELEMENT FILMST/            **$***$***C
      SUBROUTINE FILMST
      COMMON /FLM/ Z(200)
C     ***** SETUP OF OUTPUT FILM TYPE *****
C
      CALL MODESG (Z,48)
      CALL SETSMG (Z,19,15.0)
      CALL SETSMG (Z,20,10.0)
      CALL OBJCTG (Z,2.5,0.0,12.5,10.0)
      CALL SUBJEG (Z,0.0,0.0,1.0,1.0)
      CALL SCOUTH (Z)
C     ***** END OF FILM OUTPUT *****
C
      RETURN
      END
```

```
C**$$**$$**BEGINNING OF ELEMENT MESHST/            **$$**$$**C
      SUBROUTINE MESHST
      INCLUDE COMDECK,LIST
      I=1
      J=1
      X(1)=XL(1)
      Y(1)=YL(1)
      DO 30 K=1,NKX
      DXML=(XC(K)-XL(K))/NXL(K)
      DXMR=(XR(K)-XC(K))/NXR(K)
      DXMN1=DXMN(K)
      NT=NXL(K)
      TN=NT
      TN=DMAX1(TN,1.0+EM6)
      DXMN(K)=DMIN1(DXMN1,DXML)
      BMC=((2.0*TN+1.0)*(XC(K)-XL(K))-TN*(TN+2.0)*DXMN(K))/(TN*(TN-1.0))
      CMC=(TN*DXMN(K)-XC(K)+XL(K))/(TN*(TN-1.0))
      DO 10 L=1,NT
      I=I+1
   10 X(I)=X(I-1)+BMC+(2.0*DFLOAT(L)+1.0)*CMC
      NT=NXR(K)
      TN=NT
      TN=DMAX1(TN,1.0+EM6)
      DXMN(K)=DMIN1(DXMN1,DXMR)
      BMC=((2.0*TN+1.0)*(XR(K)-XC(K))-TN*(TN+2.0)*DXMN(K))/(TN*(TN-1.0))
      CMC=(TN*DXMN(K)-XR(K)+XC(K))/(TN*(TN-1.0))
      DO 20 L=1,NT
      I=I+1
   20 X(I)=X(I-1)+BMC+(2.0*TN+3.0-2.0*DFLOAT(L))*CMC
   30 CONTINUE
      DO 60 K=1,NKY
      DYML=(YC(K)-YL(K))/NYL(K)
      DYMR=(YR(K)-YC(K))/NYR(K)
      DYMN1=DYMN(K)
      NT=NYL(K)
      TN=NT
      TN=DMAX1(TN,1.0+EM6)
      DYMN(K)=DMIN1(DYMN1,DYML)
      BMC=((2.0*TN+1.0)*(YC(K)-YL(K))-TN*(TN+2.0)*DYMN(K))/(TN*(TN-1.0))
      CMC=(TN*DYMN(K)-YC(K)+YL(K))/(TN*(TN-1.0))
      DO 40 L=1,NT
      J=J+1
   40 Y(J)=Y(J-1)+BMC+(2.0*DFLOAT(L)+1.0)*CMC
      NT=NYR(K)
      TN=NT
      TN=DMAX1(TN,1.0+EM6)
      DYMN(K)=DMIN1(DYMN1,DYMR)
      BMC=((2.0*TN+1.0)*(YR(K)-YC(K))-TN*(TN+2.0)*DYMN(K))/(TN*(TN-1.0))
      CMC=(TN*DYMN(K)-YR(K)+YC(K))/(TN*(TN-1.0))
      DO 50 L=1,NT
      J=J+1
   50 Y(J)=Y(J-1)+BMC+(2.0*TN+3.0-2.0*DFLOAT(L))*CMC
   60 CONTINUE
      NUMX=I
      NUMY=J
      NUMXM1=NUMX-1
      NUMYM1=NUMY-1
      NUMXP1=NUMX+1
      NUMYP1=NUMY+1
      IBAR=NUMX-1
```

```
      JBAR=NUMY-1
      IMAX=IBAR+2
      JMAX=JBAR+2
      IM1=IMAX-1
      JM1=JMAX-1
      IM2=IMAX-2
      JM2=JMAX-2
C
C     * CALCULATE VALUES NEEDED FOR VARIABLE MESH
C
      DO 80 I=1,NUMX
      IF (X(I).EQ.0.0) GO TO 70
      RX(I)=1.0/X(I)
      GO TO 80
   70 RX(I)=0.0
   80 CONTINUE
      DO 90 I=2,NUMX
      XI(I)=0.5*(X(I-1)+X(I))
      DELX(I)=X(I)-X(I-1)
      RXI(I)=1.0/XI(I)
   90 RDX(I)=1.0/DELX(I)
      DELX(1)=DELX(2)
      XI(1)=XI(2)-DELX(2)
      RXI(1)=1.0/XI(1)
      RDX(1)=1.0/DELX(1)
      DELX(NUMXP1)=DELX(NUMX)
      XI(NUMXP1)=XI(NUMX)+DELX(NUMX)
      X(NUMXP1)=XI(NUMXP1)+0.5*DELX(NUMXP1)
      RXI(NUMXP1)=1.0/XI(NUMXP1)
      RDX(NUMXP1)=1.0/DELX(NUMXP1)
      DO 100 I=2,NUMY
      YJ(I)=0.5*(Y(I-1)+Y(I))
      RYJ(I)=1.0/YJ(I)
      DELY(I)=Y(I)-Y(I-1)
      RDY(I)=1.0/DELY(I)
  100 CONTINUE
      DELY(1)=DELY(2)
      RDY(1)=1.0/DELY(1)
      YJ(1)=YJ(2)-DELY(2)
      RYJ(1)=1.0/YJ(1)
      DELY(NUMYP1)=DELY(NUMY)
      YJ(NUMYP1)=YJ(NUMY)+DELY(NUMY)
      RYJ(NUMYP1)=1.0/YJ(NUMYP1)
      RDY(NUMYP1)=1.0/DELY(NUMYP1)
C
C     *CALCULATE BETA(I,J) MESH
C
      DO 110 I=2,NUMX
      DO 110 J=2,NUMY
      XX=DELT*RDX(I)*(2.0/(DELX(I)+DELX(I-1))+2.0/(DELX(I+1)+DELX(I)))
     1 +DELT*RDY(J)*(2.0/(DELY(J)+DELY(J-1))+2.0/(DELY(J+1)+DELY(J)))
      BETA(I,J)=OMG/XX
  110 CONTINUE
      WRITE (6,160)
      WRITE (17,160)
      DO 120 I=1,NUMXP1
      WRITE (6,170) I,X(I),I,RX(I),I,DELX(I),I,RDX(I),I,XI(I),I,RXI(I)
      WRITE (17,170) I,X(I),I,RX(I),I,DELX(I),I,RDX(I),I,XI(I),I,RXI(I)
  120 CONTINUE
      WRITE (6,160)
```

```
      WRITE (17,160)
      DO 130 I=1,NUMYP1
      WRITE (6,180) I,Y(I),I,DELY(I),I,RDY(I),I,YJ(I),I,RYJ(I)
      WRITE (17,180) I,Y(I),I,DELY(I),I,RDY(I),I,YJ(I),I,RYJ(I)
  130 CONTINUE
C
C      * * SET BETA(I,J)= -1.0 IN OBSTACLE CELLS
C          MUST BE DONE BY HAND IN GENERAL
C
      DO 140 I=2,IM1
      YCIRC=YCENTR-DSQRT(1.0-XI(I)**2)
      DO 140 J=2,JM1
      IF (IBAFF.GT.0.AND.I.EQ.8.AND.J.EQ.5) GO TO 140
      IF (XI(I).GT.0.100.AND.YJ(J).LT.YCIRC) BETA(I,J)=-1.0
      IF (IBAFF.EQ.0) GO TO 140
      IF (XI(I).LT.0.62500.AND.(YJ(J).GT.0.400.AND.YJ(J).LT.0.5600)
     1 ) BETA(I,J)=-1.0
  140 CONTINUE
      WRITE (6,160)
      WRITE (17,160)
      DO 150 J=1,NUMY
      DO 150 I=1,NUMX
      WRITE (6,190) I,J,BETA(I,J)
      WRITE (17,190) I,J,BETA(I,J)
  150 CONTINUE
      RETURN
C
  160 FORMAT (1H1)
  170 FORMAT (1X,2HX(,I2,2H)=,1PD12.5,2X,3HRX(,I2,2H)=,1PD12.5,2X,5HDELX
     1(,I2,2H)=,1PD12.5,1X,4HRDX(,I2,2H)=,1PD12.5,2X,3HXI(,I2,2H)=,1PD12
     2 .5,2X,4HRXI(,I2,2H)=,1PD12.5)
  180 FORMAT (1X,2HY(,I2,2H)=,1PD12.5,3X,5HDELY(,I2,2H)=,1PD12.5,3X,4HRD
     1Y(,I2,2H)=,1PD12.5,3X,3HYJ(,I2,2H)=,1PD12.5,3X,4HRYJ(,I2,2H)=,1PD1
     2 2.5)
  190 FORMAT (2X,5HBETA(,I2,1H,,I2,2H)=,1PD14.7)
      END
```

```
C**$$**$$**BEGINNING OF ELEMENT SETUP/            **$$**$$**C
      SUBROUTINE SETUP
      INCLUDE COMDECK,LIST
C
C     * COMPUTE CONSTANT TERMS AND INITIALIZE NECESSARY VARIABLES
C
      T=0.0
      ITER=0
      CYCLE=0
      TWPRT=0.0
      TWPLT=0.0
      NTD=0
      SIGMA=EM4*PI/WEBER
      GY=SIGMA*BOND
      CANGLE=CANGLE*RPD
      TANCA=DTAN(CANGLE)
C
C     * SET CONSTANT TERMS FOR PLOTTING
C
      XMIN=X(1)
      XMAX=X(IM1)
      IF (ISYMPL.GT.0) XMIN=-XMAX
      YMIN=Y(1)
      YMAX=Y(JM1)
      D1=XMAX-XMIN
      D2=YMAX-YMIN
      D3=DMAX1(D1,D2)
      SF=1.0/D3
      XSHFT=0.5*(1.0-D1*SF)
      YSHFT=0.5*(1.0-D2*SF)
C
C     DETERMINE SLOPED BOUNDARY LOCATION
C
C     SET INITIAL TOP SURFACE CONFIGURATION
C
C     COMPUTE INITIAL VOID - FRACTION FUNCTION F IN CELLS
      IF (ISRFIC.GT.0.AND.FLHT.GT.EM6) CALL ICON
C
C       SET F(I,J)=1.0 IN OBSTACLE CELLS
C
      DO 10 I=2,IM1
      DO 10 J=2,JM1
      IF (BETA(I,J).LT.0.0) F(I,J)=1.0
   10 CONTINUE
C
C     CALCULATE HYDROSTATIC PRESSURE
C
      DO 30 I=2,IM1
      DO 30 J=2,JM1
      IF (F(I,J).GT.1.0-EM6.OR.F(I,J).LT.EM6) GO TO 30
      JS=J
      DO 20 J1=2,JS
      P(I,J1)=-GY*(YJ(JS)+(F(I,JS)-0.5)*DELY(JS)-YJ(J1))
   20 CONTINUE
   30 CONTINUE
C
C     PARTICLE SET UP
C
      NP=PARTN
C
```

```
C      * SPECIAL INPUT DATA
C       **********
       VCHGT=0.0
       EMF1=1.0-EMF
       DO 40 J=2,JM1
       DO 40 I=2,IM1
       PS(I,J)=0.0
   40 CONTINUE
       DXMIN=EP10
       DO 50 I=2,IM1
   50 DXMIN=DMIN1(DELX(I),DXMIN)
       DYMIN=EP10
       DO 60 I=2,JM1
   60 DYMIN=DMIN1(DELY(I),DYMIN)
       VELMX1=DMIN1(DXMIN,DYMIN)/VELMX
       IF (CYCLE.GT.0) GO TO 80
C
C      * SET INITIAL VELOCITY FIELD INTO U AND V ARRAYS
C
       DO 70 I=2,IM1
       DO 70 J=2,JM1
       V(I,J)=VI
       U(I,J)=UI
       IF (F(I,J).LT.EMF) U(I,J)=0.0
       IF (F(I,J).LT.EMF) V(I,J)=0.0
   70 CONTINUE
   80 RETURN
       END
```

```
C**$$**$$**BEGINNING OF ELEMENT ICON/            **$$**$$**C
      SUBROUTINE ICON
      INCLUDE COMDECK,LIST
      DIMENSION YS(500), ZS(500), CURV(500), CURVY(500)
      FLHTO=FLHT+0.1D0
      N=500
      NM1=N-1
      EPSILN=2.0*E-4
      DEL=1.0/DFLOAT(N)
      COSCA=DCOS(CANGLE)
      IF (CYL.GT.0.0) COSCA=2.0*COSCA
      GAMMA=SIGMA*(1.0+COSCA)
      ITMX=500
      IT=1
      RNOT=0.0*CYL+(1.0-CYL)
      YSNOT=0.0
      IF (BOND.NE.0.0) YSNOT=-0.5*DABS(COSCA/BOND)
   10 CONTINUE
   20 DO 60 J=1,N
      IF (J.GT.1) GO TO 30
      ZSJM1=0.0
      YSJM1=YSNOT
      GO TO 40
   30 ZSJM1=ZS(J-1)
      YSJM1=YS(J-1)
   40 RSQZ=1.0/DSQRT(1.0-ZSJM1**2)
      RJM1=DFLOAT(J-1)*DEL*CYL+(1.0-CYL)
      RJ=DFLOAT(J)*DEL*CYL+(1.0-CYL)
      RJMH=(DFLOAT(J)-0.5)*DEL*CYL+(1.0-CYL)
      ZS(J)=(ZSJM1*RJM1+DEL*RJMH*(COSCA-BOND*(YSJM1+0.5*DEL*ZSJM1*RSQZ))
     1 )/RJ
      IF (ZS(J).LT.1.0) GO TO 50
      YSNOT=YSNOT*1.05D0
      GO TO 10
   50 RSQZN=1.0/DSQRT(1.0-ZS(J)**2)
      YS(J)=YSJM1+0.5*DEL*(ZSJM1*RSQZ+ZS(J)*RSQZN)
   60 CONTINUE
      IF (BOND.EQ.0.0) GO TO 70
      ERR=DABS(ZS(N)-COSCA*(0.5*CYL+(1.0-CYL)))
      IF (ERR.LT.EPSILN) GO TO 120
   70 YSUM=0.5*(YSNOT*RNOT+YS(N))
      DO 80 J=1,NM1
      RJ=DFLOAT(J)*DEL*CYL+(1.0-CYL)
   80 YSUM=YSUM+YS(J)*RJ
      YSUM=YSUM*DEL
      IF (YSUM.LT.EPSILN.AND.BOND.EQ.0.0) GO TO 120
      DEN=(2.0*PI*CYL+(1.0-CYL))
      YSNOT=YSNOT-YSUM/DEN
      IT=IT+1
      IF (BOND.NE.0.0) GO TO 100
      DO 90 J=1,N
      YS(J)=YS(J)-YSUM/DEN
   90 CONTINUE
      IF (IT.GT.ITMX) GO TO 110
      GO TO 70
  100 IF (IT.LE.ITMX) GO TO 20
  110 IF (IT.GT.ITMX) WRITE (6,260) IT,YSUM,ZS(N),COSCA
      CALL EXIT
  120 CONTINUE
      YSNOT=YSNOT+FLHTO
```

```
      DO 130 K=1,N
      YS(K)=YS(K)+FLHTO
  130 CONTINUE
      CALL PAGEG (Z,0,0,1)
      CALL FRAME (XMIN,XMAX,YMAX,YMIN)
      CALL DRWOBS
      DO 160 K=1,N
      IF (K.GT.1) GO TO 140
      YSKM=YSNOT
      GO TO 150
  140 YSKM=YS(K-1)
  150 YSK=YS(K)
      XKM=DFLOAT(K-1)*DEL
      XK=DFLOAT(K)*DEL
      CALL DRWVEC (XKM,YSKM,XK,YSK,1)
  160 CONTINUE
      CALL PAGEG (Z,0,0,1)
      DO 170 K=1,NM1
      DHDXR=(YS(K+1)-YS(K))/DEL
      DHDXL=(YS(K)-YS(K-1))/DEL
      GP=0.5*(DHDXR+DHDXL)
      GPP=(DHDXR-DHDXL)/DEL
      CURV(K)=-GPP/((1.0+GP**2)**1.5)
      DHDYT=-DEL/(YS(K+1)-YS(K))
      DHDYB=-DEL/(YS(K)-YS(K-1))
      GYP=0.5*(DHDYT+DHDYB)
      GYPP=2.0*(DHDYT-DHDYB)/(YS(K+1)-YS(K-1))
      CURVY(K)=-GYPP/((1.0+GYP**2)**1.5)
      WRITE (6,300) K,YS(K),CURV(K),CURVY(K)
  170 CONTINUE
      DO 180 I=2,IM1
      K=XI(I)/DEL+1.0
      XCURV=CURV(K-1)+(XI(I)-DFLOAT(K-1)*DEL)*(CURV(K)-CURV(K-1))/DEL
      WRITE (6,290) I,XI(I),XCURV
  180 CONTINUE
      DO 210 I=2,IM1
      K=XI(I)/DEL+1.0
      IF (K.GT.1) GO TO 190
      XSKM1=0.0
      YSKM1=YSNOT
      GO TO 200
  190 XSKM1=DFLOAT(K-1)*DEL
      YSKM1=YS(K-1)
  200 FLHT=YSKM1+(XI(I)-XSKM1)*(YS(K)-YSKM1)/DEL
      DO 210 J=2,JM1
      F(I,J)=1.0
      IF (FLHT.GT.Y(J-1).AND.FLHT.LT.Y(J)) F(I,J)=RDY(J)*(FLHT-Y(J-1))
      IF (Y(J-1).GE.FLHT) F(I,J)=0.0
  210 CONTINUE
      DO 250 J=2,JM1
      IF (YJ(J).LT.YSNOT.OR.YJ(J).GT.YS(N)) GO TO 250
      DO 220 KK=2,N
      K=KK
      IF (YS(KK).GT.YJ(J).AND.YS(KK-1).LT.YJ(J)) GO TO 230
  220 CONTINUE
      GO TO 250
  230 CONTINUE
      TANG=(YS(K)-YS(K-1))/DEL
      ANG=DATAN(TANG)
      IF (ANG.LE.0.25*PI) GO TO 250
```

```
      FLHTX=X(IM1)-DEL*(DFLOAT(K-1)+(YJ(J)-YS(K-1))/(YS(K)-YS(K-1)))
      DO 240 L=2,IM1
      XIML=X(IM1)-X(L)
      XIML1=X(IM1)-X(L-1)
      F(L,J)=1.0
      IF (FLHTX.GT.XIML.AND.FLHTX.LT.XIML1) F(L,J)=RDX(L)*(FLHTX-XIML)
      IF (XIML.GE.FLHTX) F(L,J)=0.0
  240 CONTINUE
  250 CONTINUE
      DO 260 J=1,JMAX
      F(1,J)=F(2,J)
  260 F(IMAX,J)=F(IM1,J)
      DO 270 I=1,IMAX
      F(I,1)=F(I,2)
  270 F(I,JMAX)=F(I,JM1)
      RETURN
C
  280 FORMAT (2X,13HERROR IN ICON,2X,I5,2X,3(2X,1PD12.5))
  290 FORMAT (2X,I3,2X,1PD12.5,2X,1PD12.5)
  300 FORMAT (2X,I3,3(2X,1PD12.5))
      END
```

```
C**$$**$$**BEGINNING OF ELEMENT BC/            **$$**$$**C
      SUBROUTINE BC
      INCLUDE COMDECK,LIST
C
C     SET BOUNDARY CONDITIONS
C
      DO 100 J=1,JMAX
      F(1,J)=F(2,J)
      F(IMAX,J)=F(IM1,J)
      P(1,J)=P(2,J)
      P(IMAX,J)=P(IM1,J)
      GO TO (10,20,30,40), WL
   10 U(1,J)=0.0
      V(1,J)=V(2,J)
      GO TO 50
   20 U(1,J)=0.0
      V(1,J)=-V(2,J)*DELX(1)/DELX(2)
      GO TO 50
   30 IF (ITER.GT.0) GO TO 50
      U(1,J)=U(2,J)
      V(1,J)=V(2,J)
      GO TO 50
   40 U(1,J)=U(IM2,J)
      V(1,J)=V(IM2,J)
      GO TO 50
   50 GO TO (60,70,80,90), WR
   60 U(IM1,J)=0.0
      V(IMAX,J)=V(IM1,J)
      GO TO 100
   70 U(IM1,J)=0.0
      V(IMAX,J)=-V(IM1,J)*DELX(IMAX)/DELX(IM1)
      GO TO 100
   80 IF (ITER.GT.0) GO TO 100
      U(IM1,J)=U(IM2,J)
      V(IMAX,J)=V(IM1,J)
      GO TO 100
   90 U(IM1,J)=U(2,J)
      V(IMAX,J)=V(3,J)
  100 CONTINUE
      DO 200 I=1,IMAX
      F(I,1)=F(I,2)
      F(I,JMAX)=F(I,JM1)
      P(I,1)=P(I,2)
      P(I,JMAX)=P(I,JM1)
      GO TO (110,120,130,140), WT
  110 V(I,JM1)=0.0
      U(I,JMAX)=U(I,JM1)
      GO TO 150
  120 V(I,JM1)=0.0
      U(I,JMAX)=-U(I,JM1)*DELY(JMAX)/DELY(JM1)
      GO TO 150
  130 IF (ITER.GT.0) GO TO 150
      V(I,JM1)=V(I,JM2)
      U(I,JMAX)=U(I,JM1)
      GO TO 150
  140 V(I,JM1)=V(I,2)
      U(I,JMAX)=U(I,3)
      GO TO 150
  150 GO TO (160,170,180,190), WB
  160 V(I,1)=0.0
```

```
      U(I,1)=U(I,2)
      GO TO 200
  170 V(I,1)=0.0
      U(I,1)=-U(I,2)*DELY(1)/DELY(2)
      GO TO 200
  180 IF (ITER.GT.0) GO TO 200
      V(I,1)=V(I,2)
      U(I,1)=U(I,2)
      GO TO 200
  190 V(I,1)=V(I,JM2)
      U(I,1)=U(I,JM2)
  200 CONTINUE
C
C     FREE SURFACE AND SLOPED BOUNDARY CONDITIONS
C
      DO 450 I=2,IM1
      XRP=RDX(I)+0.5*RXI(I)
      RXRP=1.0/XRP
      XRM=RDX(I)-0.5*RXI(I)
      IF (XRM.GT.0.0) GO TO 210
      RXRM=0.0
      GO TO 220
  210 CONTINUE
      RXRM=1.0/XRM
  220 CONTINUE
      DO 450 J=2,JM1
      IF (BETA(I,J).GT.0.0) GO TO 230
      BMR=0.0
      BMT=0.0
      BML=0.0
      BMB=0.0
      F(I,J)=0.0
      P(I,J)=0.0
      IF (BETA(I+1,J).GT.0.0) BMR=1.0
      IF (BETA(I,J+1).GT.0.0) BMT=1.0
      IF (BETA(I-1,J).GT.0.0) BML=1.0
      IF (BETA(I,J-1).GT.0.0) BMB=1.0
      BMTOT=BMR+BMT+BML+BMB
      IF (BMTOT.LE.0.0) GO TO 450
      F(I,J)=(BMR*F(I+1,J)+BMT*F(I,J+1)+BML*F(I-1,J)+BMB*F(I,J-1))/BMTOT
      P(I,J)=(BMR*P(I+1,J)+BMT*P(I,J+1)+BML*P(I-1,J)+BMB*P(I,J-1))/BMTOT
      GO TO 450
  230 CONTINUE
      IF (F(I,J).LT.EMF.OR.F(I,J).GT.1.0-EMF) GO TO 450
      NFSB=0
      IF (F(I+1,J).LT.EMF) NFSB=NFSB+1
      IF (F(I,J+1).LT.EMF) NFSB=NFSB+2
      IF (F(I-1,J).LT.EMF) NFSB=NFSB+4
      IF (F(I,J-1).LT.EMF) NFSB=NFSB+8
      IF (NFSB.EQ.0) GO TO 400
      IF (NFSB.GT.8) GO TO 240
      GO TO (250,260,270,280,290,300,310,320), NFSB
  240 NFSB1=NFSB-8
      GO TO (330,340,350,360,370,380,390), NFSB1
  250 U(I,J)=(U(I-1,J)-DELX(I)*RDY(J)*(V(I,J)-V(I,J-1)))*(1-CYL)+CYL*(U
     1 (I-1,J)*XRM*RXRP-RDY(J)*RXRP*(V(I,J)-V(I,J-1)))
      GO TO 410
  260 V(I,J)=(V(I,J-1)-DELY(J)*RDX(I)*(U(I,J)-U(I-1,J)))*(1-CYL)+CYL*(V
     1 (I,J-1)-DELY(J)*(XRP*U(I,J)-XRM*U(I-1,J)))
      GO TO 410
```

```
  270 U(I,J)=U(I-1,J)*(1-CYL)+CYL*U(I-1,J)*XRM*RXRP
      GO TO 260
  280 U(I-1,J)=(U(I,J)+DELX(I)*RDY(J)*(V(I,J)-V(I,J-1)))*(1-CYL)+CYL*(U
     1 (I,J)*XRP*RXRM+RDY(J)*RXRM*(V(I,J)-V(I,J-1)))
      GO TO 410
  290 U(I-1,J)=U(I-1,J-1)
      GO TO 250
  300 U(I-1,J)=U(I,J)*(1-CYL)+CYL*U(I,J)*XRP*RXRM
      GO TO 260
  310 U(I-1,J)=U(I-1,J-1)
      U(I,J)=U(I,J-1)
      GO TO 260
  320 V(I,J-1)=(V(I,J)+DELY(J)*RDX(I)*(U(I,J)-U(I-1,J)))*(1-CYL)+CYL*(V
     1 (I,J)+DELY(J)*(XRP*U(I,J)-XRM*U(I-1,J)))
      GO TO 410
  330 U(I,J)=U(I-1,J)*(1-CYL)+CYL*U(I-1,J)*XRM*RXRP
      GO TO 320
  340 V(I,J)=V(I-1,J)
      GO TO 320
  350 V(I,J)=V(I-1,J)
      V(I,J-1)=V(I-1,J-1)
      GO TO 250
  360 U(I-1,J)=U(I,J)*(1-CYL)+CYL*U(I,J)*XRP*RXRM
      GO TO 320
  370 U(I,J)=U(I,J+1)
      U(I-1,J)=U(I-1,J+1)
      GO TO 320
  380 V(I,J)=V(I+1,J)
      V(I,J-1)=V(I+1,J-1)
      GO TO 280
  390 U(I,J)=U(I-1,J)*(1-CYL)+CYL*U(I-1,J)*XRM*RXRP
      V(I,J-1)=V(I,J)
      V(I,J+1)=V(I,J)
      GO TO 410
  400 CONTINUE
      GO TO 450
C
C     SET VELOCITIES IN EMPTY CELLS ADJACENT TO PARTIAL FLUID CELLS
C
  410 CONTINUE
      IF (FLG.GT.0.5.AND.ITER.GT.0) GO TO 450
      IF (F(I+1,J).GT.EMF) GO TO 420
      IF (F(I+1,J+1).LT.EMF) V(I+1,J)=V(I,J)
      IF (F(I+1,J-1).LT.EMF) V(I+1,J-1)=V(I,J-1)
  420 IF (F(I,J+1).GT.EMF) GO TO 430
      IF (F(I+1,J+1).LT.EMF) U(I,J+1)=U(I,J)
      IF (F(I-1,J+1).LT.EMF) U(I-1,J+1)=U(I-1,J)
  430 IF (F(I-1,J).GT.EMF) GO TO 440
      IF (F(I-1,J+1).LT.EMF) V(I-1,J)=V(I,J)
      IF (F(I-1,J-1).LT.EMF) V(I-1,J-1)=V(I,J-1)
  440 IF (F(I,J-1).GT.EMF) GO TO 450
      IF (F(I+1,J-1).LT.EMF) U(I,J-1)=U(I,J)
      IF (F(I-1,J-1).LT.EMF) U(I-1,J-1)=U(I-1,J)
  450 CONTINUE
C
C         SPECIAL BOUNDARY CONDITIONS
C
      DO 460 I=1,IMAX
      V(I,1)=0.0
      IF (XI(I).LT.0.100) V(I,1)=VOUT
```

```
      IF (VOUT.LT.EM6) GO TO 460
      F(I,1)=C.0
      IF (XI(I).LT.0.1D0) F(I,1)=1.0
      IF (XI(I).LT.0.1D0) F(I,2)=1.0
  460 CONTINUE
      RETURN
      END
```

```
C**$$**$$**BEGINNING OF ELEMENT TILDE/            **$$**$$**C
      SUBROUTINE TILDE
      INCLUDE COMDECK,LIST
C
C     * COMPUTE TEMPORARY U AND V
C
      DO 20 J=2,JM1
      DO 20 I=2,IM1
      U(I,J)=0.0
      RDELX=1.0/(DELX(I)+DELX(I+1))
      RDELY=1.0/(DELY(J)+DELY(J+1))
      IF (P(I,J)+P(I+1,J).LT.EMF) GO TO 10
      IF (BETA(I,J).LT.0.0.OR.BETA(I+1,J).LT.0.0) GO TO 10
      SGU=DSIGN(DP1,UN(I,J))
      DUDR=(UN(I+1,J)-UN(I,J))*RDX(I+1)
      DUDL=(UN(I,J)-UN(I-1,J))*RDX(I)
      RDXA=DELX(I)+DELX(I+1)+ALPHA*SGU*(DELX(I+1)-DELX(I))
      RDXA=1.0/RDXA
      FUX=RDXA*UN(I,J)*(DELX(I)*DUDR+DELX(I+1)*DUDL+ALPHA*SGU*(DELX(I+1)
     1 *DUDL-DELX(I)*DUDR))
      VBT=(DELX(I)*VN(I+1,J)+DELX(I+1)*VN(I,J))*RDELX
      VBB=(DELX(I)*VN(I+1,J-1)+DELX(I+1)*VN(I,J-1))*RDELX
      VAV=0.5*(VBT+VBB)
      DYT=0.5*(DELY(J)+DELY(J+1))
      DYB=0.5*(DELY(J-1)+DELY(J))
      DUDT=(UN(I,J+1)-UN(I,J))/DYT
      DUDB=(UN(I,J)-UN(I,J-1))/DYB
      SGV=DSIGN(DP1,VAV)
      DYA=DYT+DYB+ALPHA*SGV*(DYT-DYB)
      FUY=(VAV/DYA)*(DYB*DUDT+DYT*DUDB+ALPHA*SGV*(DYT*DUDB-DYB*DUDT))
      UBDYT=(DELY(J)*UN(I,J+1)+DELY(J+1)*UN(I,J))/(DELY(J)+DELY(J+1))
      UBDYB=(DELY(J-1)*UN(I,J)+DELY(J)*UN(I,J-1))/(DELY(J)+DELY(J-1))
      DUDXSQ=2.0*(UN(I-1,J)*RDX(I)/(DELX(I)+DELX(I+1))+UN(I+1,J)*RDX(I+1
     1 )/(DELX(I)+DELX(I+1))-UN(I,J)*RDX(I)*RDX(I+1))
      DUDYT=(UN(I,J+1)*DELY(J)*RDY(J+1)-UN(I,J)*DELY(J+1)*RDY(J)-UBDYT*
     1 (DELY(J)*RDY(J+1)-DELY(J+1)*RDY(J)))/(0.5*(DELY(J)+DELY(J+1)))
      DUDYB=(UN(I,J)*DELY(J-1)*RDY(J)-UN(I,J-1)*DELY(J)*RDY(J-1)-UBDYB*
     1 (DELY(J-1)*RDY(J)-DELY(J)*RDY(J-1)))/(0.5*(DELY(J-1)+DELY(J)))
      DUDYSQ=(DUDYT-DUDYB)*RDY(J)
      DUDXL=(UN(I,J)-UN(I-1,J))*RDX(I)
      DUDXR=(UN(I+1,J)-UN(I,J))*RDX(I+1)
      RXDUDX=RX(I)*(DELX(I+1)*DUDXL+DELX(I)*DUDXR)/(DELX(I)+DELX(I+1))
      RXSQU=UN(I,J)*RX(I)**2
      VISX=NU*(DUDXSQ+DUDYSQ+CYL*RXDUDX-CYL*RXSQU)
      U(I,J)=UN(I,J)+DELT*((P(I,J)-P(I+1,J))*2.0*RDELX+GX-FUX-FUY+VISX)
   10 CONTINUE
      V(I,J)=0.0
      IF (P(I,J)+P(I,J+1).LT.EMF) GO TO 20
      IF (BETA(I,J).LT.0.0.OR.BETA(I,J+1).LT.0.0) GO TO 20
      UBR=(DELY(J+1)*UN(I,J)+DELY(J)*UN(I,J+1))*RDELY
      UBL=(DELY(J+1)*UN(I-1,J)+DELY(J)*UN(I-1,J+1))*RDELY
      UAV=0.5*(UBR+UBL)
      DXR=0.5*(DELX(I)+DELX(I+1))
      DXL=0.5*(DELX(I)+DELX(I-1))
      SGU=DSIGN(DP1,UAV)
      DXA=DXR+DXL+ALPHA*SGU*(DXR-DXL)
      DVDR=(VN(I+1,J)-VN(I,J))/DXR
      DVDL=(VN(I,J)-VN(I-1,J))/DXL
      FVX=(UAV/DXA)*(DXL*DVDR+DXR*DVDL+ALPHA*SGU*(DXR*DVDL-DXL*DVDR))
      SGV=DSIGN(DP1,VN(I,J))
```

```
915         DYA=DELY(J+1)+DELY(J)+ALPHA*SGV*(DELY(J+1)-DELY(J))
916         DVDT=(VN(I,J+1)-VN(I,J))*RDY(J+1)
917         DVDB=(VN(I,J)-VN(I,J-1))*RDY(J)
918         FVY=(VN(I,J)/DYA)*(DELY(J)*DVDT+DELY(J+1)*DVDB+ALPHA*SGV*(DELY(J+1
919        1 )*DVDB-DELY(J)*DVDT))
920         VBDYR=(DELX(I+1)*VN(I,J)+DELX(I)*VN(I+1,J))/(DELX(I)+DELX(I+1))
921         VBDYL=(DELX(I)*VN(I-1,J)+DELX(I-1)*VN(I,J))/(DELX(I)+DELX(I-1))
922         DVDXR=(VN(I+1,J)*DELX(I)*RDX(I+1)-VN(I,J)*DELX(I+1)*RDX(I)-VBDYR*
923        1 (DELX(I)*RDX(I+1)-DELX(I+1)*RDX(I)))/(0.5*(DELX(I+1)+DELX(I)))
924         DVDXL=(VN(I,J)*DELX(I-1)*RDX(I)-VN(I-1,J)*DELX(I)*RDX(I-1)-VBDYL*
925        1 (DELX(I-1)*RDX(I)-DELX(I)*RDX(I-1)))/(0.5*(DELX(I)+DELX(I-1)))
926         DVDXSQ=(DVDXR-DVDXL)*RDX(I)
927         DVDYSQ=2.0*(VN(I,J-1)*RDY(J)/(DELY(J+1)+DELY(J))-VN(I,J)*RDY(J+1)
928        1 *RDY(J)+VN(I,J+1)*RDY(J+1)/(DELY(J+1)+DELY(J)))
929         DVDXRX=(VBDYR-VBDYL)*RDX(I)*RXI(I)
930         VISY=NU*(DVDXSQ+DVDYSQ+CYL*DVDXRX)
931         V(I,J)=VN(I,J)+DELT*((P(I,J)-P(I,J+1))*2.0*RDELY+GY-FVX-FVY+VISY)
932      20 CONTINUE
933         RETURN
934         END
```

```
C**$$**$$**BEGINNING OF ELEMENT PRESIT/            **$$**$$**C
      SUBROUTINE PRESIT
      INCLUDE COMDECK,LIST
   10 CONTINUE
C
C     HAS CONVERGENCE BEEN REACHED
C
      IF (FLG.EQ.0.) GO TO 140
      ITER=ITER+1
      ITMAX=1000
      IF (CYCLE.LT.5) ITMAX=4000
      IF (ITER.LT.ITMAX) GO TO 20
      IF (CYCLE.LT.10) GO TO 140
      T=EP10
      GO TO 140
   20 FLG=0.0
C
C     COMPUTE UPDATED CELL PRESSURE AND VELOCITIES
C
      DO 130 J=2,JM1
      DO 130 I=2,IM1
      IF (BETA(I,J).LT.0.0) GO TO 130
      IF (F(I,J).LT.EMF) GO TO 130
      IF (NF(I,J).EQ.0) GO TO 80
C
C     CALCULATE PRESSURE FOR SURFACE CELLS
C
      NFF=NF(I,J)
      L=I
      M=J
      GO TO (30,40,50,60,130), NFF
   30 L=I-1
      GO TO 70
   40 L=I+1
      GO TO 70
   50 M=J-1
      GO TO 70
   60 M=J+1
   70 CONTINUE
      PLM=P(L,M)
      IF (NF(L,M).NE.0.AND.BETA(I,J).GT.0.0) PLM=PS(I,J)
      DELP=(1.0-BETA(I,J))*PLM+BETA(I,J)*PS(I,J)-P(I,J)
      GO TO 90
   80 CONTINUE
      D(I,J)=RDX(I)*(U(I,J)-U(I-1,J))+RDY(J)*(V(I,J)-V(I,J-1))+CYL*0.5
     1 *RXI(I)*(U(I,J)+U(I-1,J))
C
C     TEST FOR PRESSURE CONVERGENCE
C
      IF (DABS(D(I,J)/DZRO).GE.EPSI) FLG=1.0
      DELP=-BETA(I,J)*D(I,J)*BETA(I,J)
   90 CONTINUE
      P(I,J)=P(I,J)+DELP
      IF (BETA(I+1,J).LT.0.0) GO TO 100
      U(I,J)=U(I,J)+DELT*DELP*2.0/(DELX(I)+DELX(I+1))
  100 IF (BETA(I-1,J).LT.0.0) GO TO 110
      U(I-1,J)=U(I-1,J)-DELT*DELP*2.0/(DELX(I)+DELX(I-1))
  110 IF (BETA(I,J+1).LT.0.0) GO TO 120
      V(I,J)=V(I,J)+DELT*DELP*2.0/(DELY(J)+DELY(J+1))
  120 IF (BETA(I,J-1).LT.0.0) GO TO 130
```

```
      V(I,J-1)=V(I,J-1)-DELT*DELP*2.0/(DELY(J)+DELY(J-1))
  130 CONTINUE
      CALL BC
      GO TO 10
  140 CONTINUE
      RETURN
      END
```

```
1002 C**$$**$$**BEGINNING OF ELEMENT PARMOV/            **$$**$$**C
1003       SUBROUTINE PARMOV
1004       INCLUDE COMDECK,LIST
1005 C
1006 C      * SOLA-VM PARTICLE MOVEMENT SECTION
1007 C
1008       NPT=0
1009       NPN=0
1010       K=1
1011       KN=1
1012       PFLG=1.0
1013    10 IF (NP.EQ.NPT) GO TO 150
1014 C      * CALC. U WEIGHTED VELOCITY OF PARTICLE
1015       I=IP(K)
1016       J=JP(K)
1017       IF (YP(K).GT.YJ(J)) GO TO 20
1018       HPX=X(I)-XP(K)
1019       HMX=DELX(I)-HPX
1020       HPY=YJ(J)-YP(K)
1021       NORMY=(DELY(J)+DELY(J-1))*0.5
1022       HMY=NORMY-HPY
1023       UTOP=(U(I-1,J)*HPX+U(I,J)*HMX)*RDX(I)
1024       UBOT=(U(I-1,J-1)*HPX+U(I,J-1)*HMX)*RDX(I)
1025       UPART=(UTOP*HMY+UBOT*HPY)/NORMY
1026       GO TO 30
1027    20 HPX=X(I)-XP(K)
1028       HMX=DELX(I)-HPX
1029       HPY=YJ(J+1)-YP(K)
1030       NORMY=(DELY(J+1)+DELY(J))*0.5
1031       HMY=NORMY-HPY
1032       UTOP=(U(I-1,J+1)*HPX+U(I,J+1)*HMX)*RDX(I)
1033       UBOT=(U(I-1,J)*HPX+U(I,J)*HMX)*RDX(I)
1034       UPART=(UTOP*HMY+UBOT*HPY)/NORMY
1035 C      CALC. V WEIGHTED VELOCITY OF PARTICLE
1036    30 IF (XP(K).GT.XI(I)) GO TO 40
1037       NORMX=(DELX(I)+DELX(I-1))*0.5
1038       RNORMX=1.0/NORMX
1039       HPX=XI(I)-XP(K)
1040       HMX=NORMX-HPX
1041       HPY=Y(J)-YP(K)
1042       HMY=DELY(J)-HPY
1043       VTOP=(V(I-1,J)*HPY+V(I,J)*HMX)*RNORMX
1044       VBOT=(V(I-1,J-1)*HPX+V(I,J-1)*HMX)*RNORMX
1045       VPART=(VTOP*HMY+VBOT*HPY)*RDY(J)
1046       GO TO 50
1047    40 NORMX=(DELX(I)+DELX(I+1))*0.5
1048       RNORMX=1.0/NORMX
1049       HPX=XI(I+1)-XP(K)
1050       HMX=NORMX-HPX
1051       HPY=Y(J)-YP(K)
1052       HMY=DELY(J)-HPY
1053       VTOP=(V(I,J)*HPX+V(I+1,J)*HMX)*RNORMX
1054       VBOT=(V(I,J-1)*HPX+V(I+1,J-1)*HMX)*RNORMX
1055       VPART=(VTOP*HMY+VBOT*HPY)*RDY(J)
1056    50 XPART=XP(K)+UPART*DELT
1057       YPART=YP(K)+VPART*DELT
1058       IF (XPART.GT.X(I)) IP(KN)=IP(K)+1
1059       IF (XPART.LT.X(I-1)) IP(KN)=IP(K)-1
1060       IF (YPART.GT.Y(J)) JP(KN)=JP(K)+1
1061       IF (YPART.LT.Y(J-1)) JP(KN)=JP(K)-1
```

```
      XP(KN)=XPART
      YP(KN)=YPART
      IF (WL.EQ.3) GO TO 70
   60 IF (WB.EQ.3) GO TO 100
   70 IF (WR.EQ.3) GO TO 110
   80 IF (WT.EQ.3) GO TO 120
      GO TO 130
   90 IF (XP(KN).LT.X(1)) PFLG=0.0
      GO TO 60
  100 IF (YP(KN).LT.Y(1)) PFLG=0.0
      GO TO 70
  110 IF (XP(KN).GT.X(IM1)) PFLG=0.0
      GO TO 80
  120 IF (YP(KN).GT.Y(JM1)) PFLG=0.0
  130 IF (PFLG.NE.1.0) GO TO 140
      KN=KN+1
      NPN=NPN+1
  140 K=K+1
      NPT=NPT+1
      PFLG=1.0
      GO TO 10
  150 NP=NPN
      RETURN
      END
```

```
1086 C**$**$$**BEGINNING OF ELEMENT VFCONV/            **$$**$$**C
1087       SUBROUTINE VFCONV
1088       INCLUDE COMDECK,LIST
1089 C
1090 C     CONVECT THE VOLUME - FRACTION FUNCTION F
1091 C
1092       DO 30 J=1,JM1
1093       DO 30 I=1,IM1
1094       VX=U(I,J)*DELT
1095       VY=V(I,J)*DELT
1096       IA=I+1
1097       ID=I
1098       IDM=MAXO(I-1,1)
1099       RB=XI(I)+0.5*DELX(I)
1100       RA=XI(I+1)
1101       RD=XI(I)
1102       IF (VX.GE.0.0) GO TO 10
1103       IA=I
1104       ID=I+1
1105       IDM=MINO(I+2,IMAX)
1106       RA=XI(I)
1107       RD=XI(I+1)
1108    10 CONTINUE
1109       IAD=IA
1110       IF (NF(ID,J).EQ.3.OR.NF(ID,J).EQ.4) IAD=ID
1111 C     IF(NF(ID,J).EQ.0.AND.NF(IA,J).EQ.0) IAD=ID
1112       IF (FN(IA,J).LT.EMF.OR.FN(IDM,J).LT.EMF) IAD=IA
1113       FX1=FN(IAD,J)*DABS(VX)+DMAX1((DP1-FN(IAD,J))*DABS(VX)-(DP1-FN(ID,J
1114      1 ))*DELX(ID),0.000)
1115       FX=DMIN1(FX1,FN(ID,J)*DELX(ID))
1116       F(ID,J)=F(ID,J)-FX*RDX(ID)*((DABS(RB/RD))*CYL+(1.0-CYL))
1117       F(IA,J)=F(IA,J)+FX*RDX(IA)*((DABS(RB/RA))*CYL+(1.0-CYL))
1118       JA=J+1
1119       JD=J
1120       JDM=MAXO(J-1,1)
1121       IF (VY.GE.0.0) GO TO 20
1122       JA=J
1123       JD=J+1
1124       JDM=MINO(J+2,JMAX)
1125    20 CONTINUE
1126       JAD=JA
1127       IF (NF(I,JD).EQ.1.OR.NF(I,JD).EQ.2) JAD=JD
1128 C     IF(NF(I,JD).EQ.0.AND.NF(I,JA).EQ.0) JAD=JD
1129       IF (FN(I,JA).LT.EMF.OR.FN(I,JDM).LT.EMF) JAD=JA
1130       FY1=FN(I,JAD)*DABS(VY)+DMAX1((DP1-FN(I,JAD))*DABS(VY)-(DP1-FN(I,JD
1131      1 ))*DELY(JD),0.000)
1132       FY=DMIN1(FY1,FN(I,JD)*DELY(JD))
1133       F(I,JD)=F(I,JD)-FY*RDY(JD)
1134       F(I,JA)=F(I,JA)+FY*RDY(JA)
1135    30 CONTINUE
1136       DO 70 J=2,JM1
1137       DO 70 I=2,IM1
1138       IF (BETA(I,J).LT.0.0) GO TO 70
1139       VCHG=0.0
1140       IF (F(I,J).GT.EMF.AND.F(I,J).LT.EMF1) GO TO 50
1141       IF (F(I,J).GE.EMF1) GO TO 40
1142       VCHG=F(I,J)
1143       F(I,J)=0.0
1144       GO TO 50
1145    40 CONTINUE
```

```
1146        VCHG=-(1.0-F(I,J))
1147        F(I,J)=1.0
1148     50 CONTINUE
1149        VCHGT=VCHGT+VCHG*DELX(I)*DELY(J)*(XI(I)*2.0*PI*CYL+(1.0-CYL))
1150        IF (F(I,J).LT.1.0-EMF) GO TO 70
1151        IF (F(I+1,J).LT.EMF) GO TO 60
1152        IF (F(I-1,J).LT.EMF) GO TO 60
1153        IF (F(I,J+1).LT.EMF) GO TO 60
1154        IF (F(I,J-1).LT.EMF) GO TO 60
1155        GO TO 70
1156     60 F(I,J)=F(I,J)-1.1D0*EMF
1157        VCHG=1.1D0*EMF
1158        VCHGT=VCHGT+VCHG*DELX(I)*DELY(J)*(XI(I)*2.0*PI*CYL+(1.0-CYL))
1159     70 CONTINUE
1160 C
1161 C      SPECIAL BOUNDARY CONDITIONS FOR F
1162 C
1163        RETURN
1164        END
```

```
1165 C**$$**$$**BEGINNING OF ELEMENT PETACL/            **$$**$$**C
1166       SUBROUTINE PETACL
1167       INCLUDE COMDECK,LIST
1168 C
1169 C     DETERMINE THE PRESSURE INTERPOLATION FACTOR PETA
1170 C
1171       DO 10 I=1,IMAX
1172       DO 10 J=1,JMAX
1173       PS(I,J)=0.0
1174    10 PETA(I,J)=1.0
1175       IPASS=0
1176       DO 160 I=2,IM1
1177       DO 160 J=2,JM1
1178       NF(I,J)=0
1179       IF (BETA(I,J).LT.0.0) GO TO 160
1180       IF (F(I,J).LT.EMF.OR.F(I,J).GT.1.0-EMF) GO TO 160
1181       IF (F(I+1,J).LT.EMF) GO TO 20
1182       IF (F(I,J+1).LT.EMF) GO TO 20
1183       IF (F(I-1,J).LT.EMF) GO TO 20
1184       IF (F(I,J-1).LT.EMF) GO TO 20
1185       GO TO 160
1186    20 CONTINUE
1187 C
1188 C     CALCULATE THE PARTIAL DERIVATIVES OF F
1189 C
1190       DXAV=0.5*DELX(I-1)+DELX(I)+0.5*DELX(I+1)
1191       DYAV=0.5*DELY(J-1)+DELY(J)+0.5*DELY(J+1)
1192       FIPJM=F(I+1,J-1)
1193       IF (I.EQ.IM1.OR.J.EQ.2.OR.BETA(I+1,J-1).LT.0.0) FIPJM=1.0
1194       FIPJ=F(I+1,J)
1195       IF (I.EQ.IM1.OR.BETA(I+1,J).LT.0.0) FIPJ=1.0
1196       FIPJP=F(I+1,J+1)
1197       IF (I.EQ.IM1.OR.J.EQ.JM1.OR.BETA(I+1,J+1).LT.0.0) FIPJP=1.0
1198       FIJM=F(I,J-1)
1199       IF (J.EQ.2.OR.BETA(I,J-1).LT.0.0) FIJM=1.0
1200       FIMJM=F(I-1,J-1)
1201       IF (J.EQ.2.OR.BETA(I-1,J-1).LT.0.0) FIMJM=1.0
1202       FIMJ=F(I-1,J)
1203       IF (BETA(I-1,J).LT.0.0) FIMJ=1.0
1204       FIMJP=F(I-1,J+1)
1205       IF (BETA(I-1,J+1).LT.0.0) FIMJP=1.0
1206       FIJP=F(I,J+1)
1207       IF (J.EQ.JM1.OR.BETA(I,J+1).LT.0.0) FIJP=1.0
1208       AVFCX=FIJM*DELY(J-1)+F(I,J)*DELY(J)+FIJP*DELY(J+1)
1209       AVFCY=FIMJ*DELX(I-1)+F(I,J)*DELX(I)+FIPJ*DELX(I+1)
1210       AVFR=FIPJM*DELY(J-1)+FIPJ*DELY(J)+FIPJP*DELY(J+1)
1211       AVFL=FIMJM*DELY(J-1)+FIMJ*DELY(J)+FIMJP*DELY(J+1)
1212       AVFT=FIMJP*DELX(I-1)+FIJP*DELX(I)+FIPJP*DELX(I+1)
1213       AVFB=FIMJM*DELX(I-1)+FIJM*DELX(I)+FIPJM*DELX(I+1)
1214 C
1215 C       BOUNDARY CONDITIONS FOR WALL ADHESION
1216 C
1217       IF (BETA(I+1,J).GE.0.0.AND.I.NE.IM1) GO TO 30
1218       AVFR=AVFCX+0.5*(DELX(I)+DELX(I+1))/TANCA
1219       IF (F(I,J+1).LT.EMF) AVFT=AVFCY-0.5*(DELY(J)+DELY(J+1))*TANCA
1220       IF (F(I,J-1).LT.EMF) AVFB=AVFCY-0.5*(DELY(J)+DELY(J-1))*TANCA
1221    30 IF (BETA(I,J+1).GE.0.0.AND.J.NE.JM1) GO TO 40
1222       AVFT=AVFCY+0.5*(DELY(J)+DELY(J+1))/TANCA
1223       IF (F(I+1,J).LT.EMF) AVFR=AVFCX-0.5*(DELX(I)+DELX(I+1))*TANCA
1224       IF (F(I-1,J).LT.EMF) AVFL=AVFCX-0.5*(DELX(I)+DELX(I-1))*TANCA
```

```
   40 IF (BETA(I,J-1).GE.0.0.AND.J.NE.2) GO TO 50
      AVFB=AVFCY+0.5*(DELY(J)+DELY(J-1))/TANCA
      IF (F(I+1,J).LT.EMF) AVFR=AVFCX-0.5*(DELX(I)+DELX(I+1))*TANCA
      IF (F(I-1,J).LT.EMF) AVFL=AVFCX-0.5*(DELX(I)+DELX(I-1))*TANCA
   50 IF (BETA(I-1,J).GE.0.0) GO TO 60
      AVFL=AVFCX+0.5*(DELX(I)+DELX(I-1))/TANCA
      IF (F(I,J+1).LT.EMF) AVFT=AVFCY-0.5*(DELY(J)+DELY(J+1))*TANCA
      IF (F(I,J-1).LT.EMF) AVFB=AVFCY-0.5*(DELY(J)+DELY(J-1))*TANCA
   60 CONTINUE
      PFX=(AVFR-AVFL)/DXAV
      PFY=(AVFT-AVFB)/DYAV
      PF=PFX**2+PFY**2
      IF (PF.GT.EM10) GO TO 70
      NF(I,J)=5
      P(I,J)=0.25*(P(I+1,J)+P(I,J+1)+P(I-1,J)+P(I,J-1))
      GO TO 160
   70 CONTINUE
C
C     DETERMINE THE PRESSURE INTERPOLATION CELL NF
C
      ABPFX=DABS(PFX)
      ABPFY=DABS(PFY)
      L=I
      M=J
      IF (ABPFY.GE.ABPFX) GO TO 80
      DXDYR=DELY(J)*RDX(I)
      PFMN=ABPFY
      NF(I,J)=2
      L=I+1
      DMX=DELX(I)
      DMIN=0.5*(DMX+DELX(I+1))
      IF (PFX.GT.0.0) GO TO 90
      NF(I,J)=1
      L=I-1
      DMX=DELX(I)
      DMIN=0.5*(DMX+DELX(I-1))
      GO TO 90
   80 CONTINUE
      DXDYR=DELX(I)*RDY(J)
      PFMN=ABPFX
      NF(I,J)=4
      M=J+1
      DMX=DELY(J)
      DMIN=0.5*(DMX+DELY(J+1))
      IF (PFY.GT.0.0) GO TO 90
      NF(I,J)=3
      M=J-1
      DMX=DELY(J)
      DMIN=0.5*(DMX+DELY(J-1))
   90 CONTINUE
      TANTH=PFMN
      IF (ISRF10.LT.1) GO TO 140
      IF (NF(I,J).LE.2) GO TO 100
      RIGHT=2.0*(AVFR-AVFCX)/(DELX(I)+DELX(I+1))
      XLEFT=2.0*(AVFCX-AVFL)/(DELX(I)+DELX(I-1))
      GPP=2.0*(RIGHT-XLEFT)/DXAV
      GP=0.5*(RIGHT+XLEFT)
      GO TO 110
  100 CONTINUE
      IF (NF(I,J).EQ.2.AND.F(I+1,J).GT.EMF.AND.F(I+1,J).LT.EMF1) AVFCY
```

```
     1 =FIMJ*DELX(I-1)+F(I,J)*DELX(I)+DELX(I+1)
      TOP=2.0*(AVFT-AVFCY)/(DELY(J)+DELY(J+1))
      BOTTOM=2.0*(AVFCY-AVFB)/(DELY(J)+DELY(J-1))
      GPP=2.0*(TOP-BOTTOM)/DYAV
      GP=0.5*(TOP+BOTTOM)
  110 CURVCY=0.0
      IF (CYL.LT.1.0) GO TO 120
      XLITLR=XI(I)
      IF (NF(I,J).EQ.1) XLITLR=X(I-1)+F(I,J)*DELX(I)
      IF (NF(I,J).EQ.2) XLITLR=X(I+1)-F(I,J)*DELX(I)
      RLITLR=1.0/XLITLR
      TRIG=DABS(DSIN(DATAN(TANTH)))
      IF (NF(I,J).LE.2) TRIG=DABS(DCOS(DATAN(TANTH)))
      CURVCY=-CYL*TRIG*DSIGN(DP1,PFX)*RLITLR
  120 CURVXY=-GPP/(1.0+GP**2)**1.5
      CURV=CURVXY+CURVCY
      PS(I,J)=SIGMA*CURV
      IF ((AVFL.LT.EM6.OR.AVFR.LT.EM6).AND.(AVFT.LT.EM6.OR.AVFB.LT.EM6))
     1 GO TO 130
      GO TO 140
  130 PS(I,J)=EP10
      IPASS=1
  140 CONTINUE
      IF (F(I,J).LT.EMF) GO TO 150
      NFSB=0
      IF (F(I+1,J).LT.EMF.OR.I.EQ.IM1.OR.BETA(I+1,J).LT.0.0) NFSB=NFSB+1
      IF (F(I,J+1).LT.EMF.OR.BETA(I,J+1).LT.0.0) NFSB=NFSB+2
      IF (F(I-1,J).LT.EMF.OR.BETA(I-1,J).LT.0.0) NFSB=NFSB+4
      IF (F(I,J-1).LT.EMF.OR.BETA(I,J-1).LT.0.0) NFSB=NFSB+8
      IF (NFSB.EQ.15) PS(I,J)=0.0
  150 CONTINUE
      DFS=(0.5-F(I,J))*DMX
      IF (F(I,J).LT.0.5*TANTH*DXDYR) DFS=0.5*DMX*(1.0+DXDYR*TANTH-DSQRT(
     1 8.0*F(I,J)*DXDYR*TANTH))
      BETA(I,J)=1.0/(1.0-DFS/DMIN)
      IF (L.EQ.1.OR.L.EQ.IMAX) BETA(I,J)=1.0
      IF (M.EQ.1.OR.M.EQ.JMAX) BETA(I,J)=1.0
      IF (BETA(L,M).LT.0.0) BETA(I,J)=1.0
  160 CONTINUE
      IF (IPASS.LT.1) GO TO 230
      DO 220 J=2,JM1
      DO 220 I=2,IM1
      IF (NF(I,J).LT.1.OR.NF(I,J).GT.4.OR.BETA(I,J).LT.0.0) GO TO 220
      IF (PS(I,J).LT.EP9) GO TO 220
      PS(I,J)=0.0
      NFF=NF(I,J)
      GO TO (170,180,190,200), NFF
  170 L=I-1
      M=J
      GO TO 210
  180 L=I+1
      M=J
      GO TO 210
  190 L=I
      M=J-1
      GO TO 210
  200 L=I
      M=J+1
      GO TO 210
  210 IF (PS(L,M).LT.EP9) PS(I,J)=PS(L,M)
```

```
  220 CONTINUE
  230 CONTINUE
C
C * * SET PETA IN ADJACENT FULL CELL
C
      DO 290 J=1,JMAX
      DO 290 I=1,IMAX
      NFF=NF(I,J)
      IF (NFF.EQ.0) GO TO 290
      L=I
      M=J
      GO TO (240,250,260,270,290), NFF
  240 L=I-1
      DMX=DELX(I)
      DMIN=0.5*(DMX+DELX(I-1))
      GO TO 280
  250 L=I+1
      DMX=DELX(I)
      DMIN=0.5*(DMX+DELX(I+1))
      GO TO 280
  260 M=J-1
      DMX=DELY(J)
      DMIN=0.5*(DMX+DELY(J-1))
      GO TO 280
  270 M=J+1
      DMX=DELY(J)
      DMIN=0.5*(DMX+DELY(J+1))
  280 CONTINUE
      IF (NF(L,M).GT.0) GO TO 290
      BPD=1.0-BETA(L,M)*(1.0-PETA(I,J))*DELT/(DMIN*DMX)
      PETA(L,M)=1.0/BPD
  290 CONTINUE
C
C     SET EMPTY CELL PRESSURES
C
      DO 300 I=1,IMAX
      DO 300 J=1,JMAX
      IF (F(I,J).GT.EMF.OR.BETA(I,J).LT.0.0) GO TO 300
      P(I,J)=0.0
  300 CONTINUE
      RETURN
      END
```

```
1367 C**$$**$$**BEGINNING OF ELEMENT DRAW/            **$$**$$**C
1388       SUBROUTINE DRAW
1389       INCLUDE COMDECK,LIST
1390 C
1391 C      VELOCITY VECTOR PLOT
1392 C
1393       IF (MOVY.EQ.0) GO TO 10
1394       IF (T.LT.TMVP-EM10) GO TO 110
1395       TMVP=TMVP+DTMVP
1396    10 CONTINUE
1397       CALL PAGEG (Z,0,0,1)
1398       CALL FRAME (XMIN,XMAX,YMAX,YMIN)
1399       IF (MOVY.EQ.1) GO TO 20
1400       WRITE (17,130) NAME
1401       WRITE (17,120) T,CYCLE
1402    20 CONTINUE
1403       CALL DRWOBS
1404       DO 30 J=2,JM1
1405       DO 30 I=2,IM1
1406       IF (CYCLE.GT.0.AND.F(I,J).LT.EMF) GO TO 30
1407       IF (F(I,J).LT.0.5) GO TO 30
1408       IF (BETA(I,J).LT.0.0) GO TO 30
1409       XCC=XI(I)
1410       YCC=0.5*(Y(J)+Y(J-1))
1411       UVEC=(U(I-1,J)+U(I,J))*0.5*VELMX1+XCC
1412       VVEC=(V(I,J-1)+V(I,J))*0.5*VELMX1+YCC
1413       CALL DRWVEC (XCC,YCC,UVEC,VVEC,1)
1414       CALL PLTPT (XCC,YCC,ICH,1)
1415    30 CONTINUE
1416 C
1417 C      DRAW FREE SURFACE
1418 C
1419       FPL=0.5
1420       DO 50 I=2,IM1
1421       DO 50 J=2,JM1
1422       IF (BETA(I,J).LT.0.0) GO TO 50
1423       FATR=0.25*(F(I,J)+F(I+1,J)+F(I,J+1)+F(I+1,J+1))
1424       FXTR=0.5*(F(I+1,J+1)+F(I+1,J)-F(I,J+1)-F(I,J))/(XI(I+1)-XI(I))
1425       FYTR=0.5*(F(I,J+1)+F(I+1,J+1)-F(I,J)-F(I+1,J))/(YJ(J+1)-YJ(J))
1426       FTRS=FXTR**2+FYTR**2
1427       IF (FTRS.EQ.0.0) FTRS=EP10
1428       XTR=0.5*(XI(I+1)+XI(I))+(FPL-FATR)*FXTR/FTRS
1429       XTR=DMAX1(XTR,XI(I))
1430       XTR=DMIN1(XTR,XI(I+1))
1431       XTRM=-XTR
1432       YTR=0.5*(YJ(J)+YJ(J+1))+(FPL-FATR)*FYTR/FTRS
1433       YTR=DMAX1(YTR,YJ(J))
1434       YTR=DMIN1(YTR,YJ(J+1))
1435       IF (F(I,J).GT.0.5.AND.F(I+1,J).GT.0.5) GO TO 40
1436       IF (F(I,J).LT.0.5.AND.F(I+1,J).LT.0.5) GO TO 40
1437       FABR=0.25*(F(I,J)+F(I+1,J)+F(I,J-1)+F(I+1,J-1))
1438       FXBR=0.5*(F(I+1,J)+F(I+1,J-1)-F(I,J)-F(I,J-1))/(XI(I+1)-XI(I))
1439       FYBR=0.5*(F(I,J)+F(I+1,J)-F(I,J-1)-F(I+1,J-1))/(YJ(J)-YJ(J-1))
1440       FBRS=FXBR**2+FYBR**2
1441       IF (FBRS.EQ.0.0) FBRS=EP10
1442       XBR=0.5*(XI(I+1)+XI(I))+(FPL-FABR)*FXBR/FBRS
1443       XBR=DMAX1(XBR,XI(I))
1444       XBR=DMIN1(XBR,XI(I+1))
1445       YBR=0.5*(YJ(J)+YJ(J-1))+(FPL-FABR)*FYBR/FBRS
1446       YBR=DMAX1(YBR,YJ(J-1))
```

```
      YBR=DMIN1(YBR,YJ(J))
      CALL DRWVEC (XBR,YBR,XTR,YTR,1)
   40 CONTINUE
      IF (F(I,J).GT.0.5.AND.F(I,J+1).GT.0.5) GO TO 50
      IF (F(I,J).LT.0.5.AND.F(I,J+1).LT.0.5) GO TO 50
      FATL=0.25*(F(I,J)+F(I,J+1)+F(I-1,J)+F(I-1,J+1))
      FXTL=0.5*(F(I,J+1)+F(I,J)-F(I-1,J+1)-F(I-1,J))/(XI(I)-XI(I-1))
      FYTL=0.5*(F(I-1,J+1)+F(I,J+1)-F(I-1,J)-F(I,J))/(YJ(J+1)-YJ(J))
      FTLS=FXTL**2+FYTL**2
      IF (FTLS.EQ.0.0) FTLS=EPIO
      XTL=0.5*(XI(I-1)+XI(I))+(FPL-FATL)*FXTL/FTLS
      XTL=DMAX1(XTL,XI(I-1))
      XTL=DMIN1(XTL,XI(I))
      YTL=0.5*(YJ(J)+YJ(J+1))+(FPL-FATL)*FYTL/FTLS
      YTL=DMAX1(YTL,YJ(J))
      YTL=DMIN1(YTL,YJ(J+1))
      CALL DRWVEC (XTL,YTL,XTR,YTR,1)
   50 CONTINUE
C
C     MESH PLOT
C
      IF (MOVY.EQ.1) GO TO 110
      IF (T.GT.0.0) GO TO 80
      CALL PAGEG (Z,0,0,1)
      CALL DRWOBS
      DO 60 J=1,JM1
      YCC=Y(J)
      CALL DRWVEC (XMIN,YCC,XMAX,YCC,0)
   60 CONTINUE
      DO 70 I=1,IM1
      XCC=X(I)
      CALL DRWVEC (XCC,YMIN,XCC,YMAX,1)
   70 CONTINUE
   80 CONTINUE
C
C     PLOT PARTICLES
C
      IF (NP.EQ.0) GO TO 100
      CALL PAGEG (Z,0,0,1)
      CALL DRWOBS
      CALL FRAME (XMIN,XMAX,YMAX,YMIN)
      WRITE (17,130) NAME
      WRITE (17,120) T,CYCLE
      DO 90 N=1,NP
      CALL PLTPT (XP(N),YP(N),ICH,1)
   90 CONTINUE
  100 CONTINUE
  110 CONTINUE
      IF (MOVY.EQ.0) CALL PAGEG (Z,0,0,1)
      RETURN
C
  120 FORMAT (1H+,90X,2HT=,1PD10.3,4X,6HCYCLE=,I4)
  130 FORMAT (1H ,1PX,20A4,1X,A10,2(1X,A8))
      END
```

```
1501 C**$**$**$**BEGINNING OF ELEMENT FRAME/            **$**$**$**C
1502       SUBROUTINE FRAME (XXL,XXR,YYT,YYB)
1503       IMPLICIT DOUBLE PRECISION (X-Y)
1504       CALL DRWVEC (XXL,YYT,XXR,YYT,0)
1505       CALL DRWVEC (XXL,YYT,XXL,YYB,0)
1506       CALL DRWVEC (XXL,YYB,XXR,YYB,0)
1507       CALL DRWVEC (XXR,YYB,XXR,YYT,0)
1508       RETURN
1509       END
```

```
1510 C**$$**$$**BEGINNING OF ELEMENT DRWOBS/            **$$**$$**C
1511       SUBROUTINE DRWOBS
1512       INCLUDE COMDECK,LIST
1513       XONE=1.0
1514       YONE=Y(JM1)
1515       XTWO=1.0
1516       YTWO=YCENTR
1517       CALL DRWVEC (XONE,YONE,XTWO,YTWO,1)
1518       THSTAR=-1.47062890500
1519       TH1=0.0
1520       XONE=1.0
1521       YONE=YCENTR
1522       DTH=THSTAR*0.0500
1523       DO 10 I=1,20
1524       TH2=TH1+DTH
1525       XTWO=DCOS(TH2)
1526       YTWO=DSIN(TH2)+YCENTR
1527       CALL DRWVEC (XONE,YONE,XTWO,YTWO,1)
1528       TH1=TH2
1529       XONE=XTWO
1530       YONE=YTWO
1531    10 CONTINUE
1532       XTWO=0.100
1533       YTWO=0.0
1534       CALL DRWVEC (XONE,YONE,XTWO,YTWO,1)
1535 C
1536 C      DRAW AROUND ALL OBSTACLES
1537 C
1538       DO 30 I=2,IM1
1539       DO 30 J=2,JM1
1540       IF ((BETA(I,J).LT.0.0.AND.BETA(I+1,J).LT.0.0).OR.(BETA(I,J).GT.0.0
1541      1 .AND.BETA(I+1,J).GT.0.0)) GO TO 20
1542       XONE=X(I)
1543       XTWO=XONE
1544       YONE=Y(J-1)
1545       YTWO=Y(J)
1546       CALL DRWVEC (XONE,YONE,XTWO,YTWO,1)
1547    20 IF ((BETA(I,J).LT.0.0.AND.BETA(I,J+1).LT.0.0).OR.(BETA(I,J).GT.0.0
1548      1 .AND.BETA(I,J+1).GT.0.0)) GO TO 30
1549       XONE=X(I-1)
1550       XTWO=X(I)
1551       YONE=Y(J)
1552       YTWO=YONE
1553       CALL DRWVEC (XONE,YONE,XTWO,YTWO,1)
1554    30 CONTINUE
1555       RETURN
1556       END
```

```
C**$$**$$**BEGINNING OF ELEMENT PLTPT/            **$$**$$**C
      SUBROUTINE PLTPT (XONE,YONE,ICHAR,ISYM)
      INCLUDE COMDECK,LIST
      IC=0
      X1=XONE
      Y1=YONE
   10 XO1=(X1-XMIN)*SF+XSHFT
      YO1=(Y1-YMIN)*SF+YSHFT
      CALL SETSMG (Z,84,149)
      CALL POINTG (Z,1,XO1,YO1)
      IF (ISYMPL.EQ.0.OR.ISYM.EQ.0) GO TO 20
      IC=IC+1
      IF (IC.GT.1) GO TO 20
      X1=-X1
      GO TO 10
   20 RETURN
      END
```

```
1574 C**$$**$$**BEGINNING OF ELEMENT DRWVEC/            **$$**$$**C
1575       SUBROUTINE DRWVEC (XONE,YONE,XTWO,YTWO,ISYM)
1576       INCLUDE COMDECK,LIST
1577       IC=0
1578       X1=XONE
1579       Y1=YONE
1580       X2=XTWO
1581       Y2=YTWO
1582    10 XO1=(X1-XMIN)*SF+XSHFT
1583       YO1=(Y1-YMIN)*SF+YSHFT
1584       XO2=(X2-XMIN)*SF+XSHFT
1585       YO2=(Y2-YMIN)*SF+YSHFT
1586       CALL SEGMTG (Z,1,XO1,YO1,XO2,YO2)
1587       IF (ISYMPL.EQ.0.OR.ISYM.EQ.0) GO TO 20
1588       IC=IC+1
1589       IF (IC.GT.1) GO TO 20
1590       X1=-X1
1591       X2=-X2
1592       GO TO 10
1593    20 RETURN
1594       END
```

☆ U.S. Government Printing Office: 1979 — 677—115/247