ORNL/TM-1999/292

VOLVOF: An Update of the CFD Code, SOLA-VOF

James E. Park

December 1999

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
LOCKHEED MARTIN ENERGY RESEARCH CORP.
for the
DEPARTMENT OF ENERGY

under contract DE-AC05-96OR22464

CONTENTS

Page

ABSTRACT

The SOLA-VOF code developed by the T-3 (Theoretical Physics, Fluids) group at the Los Alamos National Laboratory (LANL) has been extensively modified at the Oak Ridge National Laboratory (ORNL). The modified and improved version has been dubbed "VOLVOF," to acknowledge the state of Tennessee, the Volunteer state and home of ORNL.

Modifications include generalization of boundary conditions, additional flexibility in setting up problems, addition of a problem interruption and restart capability, segregation of graphics functions to allow utilization of modern commercial graphics programs, and addition of time and date stamps to output files. Also, the pressure iteration has been restructured to exploit the much greater system memory available on modern workstations and personal computers. A solution monitoring capability has been added to utilize the multi-tasking capability of modern computer operating systems.

These changes are documented in the following report. NAMELIST input variables are defined, and input files and the resulting output are given for two test problems.

Modification and documentation of a working technical computer program is almost never complete. This is certainly true for the present effort. However, the impending retirement of the writer dictates that the current configuration and capability be reported.

1. INTRODUCTION

1.1 BACKGROUND

1.1.1 History

A family of computational fluid dynamics (CFD) codes was developed at the Los Alamos National Laboratory (LANL) during the 1970's.[1] Dubbed the SOLA (for **SOL**ution **A**lgorithm) series, they were developed for teaching CFD and to allow ease of modification, but possessed substantial utility for solving interesting fluid flow problems.

These codes built on the unique capabilities of the T-3 group,[*] which traces its history to the work of John Von Neumann during the second world war. The codes were conceived by C. W. Hirt; algorithm refinement and the reduction from concept to working codes was accomplished by a family of collaborators, including among others A. A. Amsden, T. D. Butler, L. D. Cloutman, B. J. Daly, R. S. Hotchkiss, R. C. Mjolsness, B. D. Nichols, H. M. Ruppel, and M. D. Torrey.

Since the codes were developed, dramatic changes have occurred in the computing environments available. On the hardware front, capacity and capability have exploded, and moderately-priced desktop systems (workstations and personal computers) can outperform the multi-million dollar computers for which the SOLA family was developed. On the software front, the FORTRAN language has acquired a high level of commonality between vendors. Operating systems now allow multi-tasking so that solution progress can be monitored while a problem is being run. Taken together, these developments have made standardization of file handling and coding practices practical and economical. Thus codes can be structured for portability (easy transfer between platforms) regardless of the maker of the hardware.

1.1.2 Utility and Economics

Despite their age, the inherent simplicity of the SOLA algorithms makes them attractive for continued use. In particular, the VOF (Volume of Fluid) algorithm for tracking the movement of a free surface or of interfaces between two fluids has emerged as the standard method of simulating such motions. Thus, even a simple implementation has utility for many problems. The SOLA-VOF code[2] contains the first implementation of the VOF algorithm.

Thus, the utility of the SOLA-VOF code and the availability of powerful workstations make the upgrading of the original a viable proposition. The end product of such an effort is an effective complement to commercially-available CFD codes, such as FLOW-3D,[†] CFX,[‡] FIDAP,[**] and Fluent,[††] as well as several others. The two approaches are compared as followed:

---

[*] See http://gnarly.lanl.gov/home.html

[†] Flow Science, Inc., Los Alamos, NM

[‡] AEA Technology Engineering Software, Inc, Bethel Park, PA

[**] Fluid Dynamics International, Evanston, IL

[††] Fluent Inc., Lebanon, NH

1a. Commercial codes require an up-front investment plus an annual support/upgrade fee but are available “immediately” upon payment; development, training, and support are provided as long as licenses are kept current. Development is targeted to the needs of a customer community with diverse needs.

1b. In-lab codes require ongoing funding to support development and maintenance, but availability of the code continues if funding for development is suspended. Development can be targeted to the specific needs of the user. The ‘developer’ of an in-lab code will likely have an advanced degree in addition to experience in CFD and numerical analysis.

2a. Commercial codes must be run on systems specified by their license, and the number of simultaneous users is limited by the license or by the capacity of the system. If several people wish to use the code at any one time, most must wait.

2b. In-lab codes can be installed on as many systems as desired and problems may be moved from system to system during solution to utilize temporary excess capacity.

3a. Usually, access to the source code for commercial products is provided at substantial extra cost and is subject to use only on platforms specified by the license. Unless the source code is purchased, simulation of unique physical phenomena must be shoe-horned into the structure for “user access” provided by the vendor.

3b. In-lab codes *are* source codes, allowing unique physics sub-models to be accommodated more readily.

4a. Details of algorithms, “tricks of the trade” that give an algorithm robustness and efficiency, are protected as proprietary trade secrets by commercial entities. Without access to the details of a code feature, the purchaser encounters the real possibility that the computational tool is marginal for the required application. Often, a person contemplating licensing of a commercial code must rely on the advice of the vendor in such matters.

4b. The algorithms used for in-lab codes are open to the extent that the source code can be understood by the user. The authors usually have no commercial interest in their code and can therefore offer candid advice to a potential user.

Clearly, the choice between using a commercial or an in-lab code depends on a number of factors, including availability of personnel and computing resources, time, cost, and other detailed requirements for the simulation to be undertaken. In the writer’s experience, having an in-lab code available does not replace timely access to a commercial code, but the in-lab code does increase the options available to the analyst.

1.1.3 Potential Modifications

The SOLA codes were developed for particular sponsors; the emphasis in development was to meet the particular requirements of each sponsor. As a result, the types of boundary conditions available were limited. The underlying algorithm has a more general applicability than that provided by those specific coded conditions. Similarly, input-output options were tailored to the needs of the sponsor, and graphics routines that depend on particular hardware were included. The ability to closely time execution to allow optimization was not provided. Finally, the ability to interrupt and restart a problem solution was not always provided. These are

Finally, expanding the range of physical problems that a code can simulate can sometimes be done with very little coding effort.

1.1.4 Source

In order to obtain feedback on the performance of their CFD codes, members of the T-3 group at the Los Alamos National Laboratory have in the past provided development copies of CFD source codes to "Friendly Users." Acquistion of source-language code in this way implies that the source of the code would be acknowledged when reporting extensions. Additionally, any problems that might be encountered would be reported to the providers.

The author obtained the SOLA-VOF source code in 1986. As received, the code was as described in an LANL report.[2] Additional information is contained in Hirt and Nichols.[3,4] Further development of the VOF algorithm at Los Alamos is described in, for example, Torrey et al[5,6] and Kothe, Mjolsness, and Torrey.[7] According to the vendors, the VOF algorithm is used in the commercial CFD codes, FLOW-3D, CFX, and Fluent referenced above.

1.2 REVISIONS

The original Fortran code has been replaced with Fortran 77 syntax whenever algorithm modifications were introduced. The latest version of Fortran (90) has not been used because it was not available when this work was undertaken.

2. REVIEW OF SOLA-VOF FEATURES AND NOTATION

The SOLA-VOF code is well-documented in the original manual[2] and subsequent reports. In this chapter, parts of that work are summarized as a preamble to describing the modifications to be introduced.

The SOLA-VOF code was designed to simulate the flow of an incompressible fluid. The flow is to be represented in two spatial dimensions (x, y - Cartesian) or (r, z - cylindrical) and will usually be time-dependent. Transient flows possessing a free surface or flows in which two fluids are separated by a sharp interface are especially amenable to solution by this code.

2.1 EQUATIONS TO BE SOLVED

The equations are two components of the momentum equation (Newton's Second Law applied to a fluid), an equation expressing the conservation of mass, and an equation describing the motion of the free surface or fluid interface, if present. These momentum equations and the continuity equation will be given here because some notational changes have been made in anticipation of the addition of an equation for the rotational velocity. The

$$\nu = \mu/\rho \tag{1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + g_r + \nu\left[\frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + \xi\left(\frac{1}{r}\frac{\partial u}{\partial r} - \frac{u}{r^2}\right)\right] \tag{2}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial r} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + g_z + \nu\left[\frac{\partial^2 w}{\partial r^2} + \frac{\partial^2 w}{\partial z^2} + \frac{\xi}{r}\frac{\partial w}{\partial r}\right] \tag{3}$$

$$\frac{1}{kc^2}\frac{\partial p}{\partial t} + \frac{\partial u}{\partial r} + \frac{\partial w}{\partial z} + \hat{\imath}\frac{u}{r} = 0 \qquad \textbf{(4)}$$

The first term on the left is used to incorporate a slight compressibility effect, such as acoustic waves, into the formulation. For a truly incompressible fluid, the acoustic velocity c is infinite. In that case, $(1/c^2)$ is set to 0 in the input to SOLA-VOF, and Eq. (4) reduces to the usual incompressible continuity equation.

2.2 MESH LAYOUT

The partial differential equations describing the flow are solved on a mesh of rectangular computational cells. Earlier examples of this technique include the MAC (Marker-and-Cell);[8,9] SMAC (Simplified Marker-and-Cell)[10] and ICE (Implicit Compressible Eulerian).[11]

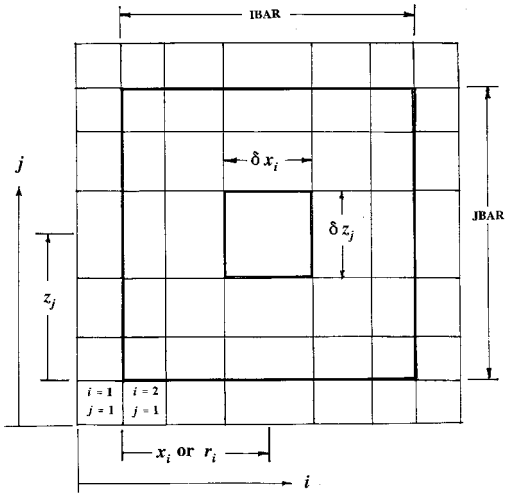


Fig. 1. Finite difference mesh with variable width rectangular cells.

A typical mesh, which is surrounded with a ring of image or "ghost" cells, is shown in Fig. 1. The image cells are employed for convenience in imposing boundary conditions on the equations. The index "*i*" is used to count the cells in the r direction; similarly, "*j*" is used to count in the z direction. As implied by the sketch, the cells are rectangular but the width of each column and row can be unique. The region on which the flow equations are solved consists of "IBAR" cells in the x,r direction and "JBAR" cells in the z direction. The cell widths are designated by δx_i and δz_j for the *i*,*j* cell. The coordinates of the cell center are designated by r_i and z_j.

When deriving the finite difference representation of the partial differential equations, the continuous variation of the dependent variables (velocities and pressure) is replaced by discrete values at specified locations on the mesh.

3. CHANGES TO SOLA-VOF

At ORNL, modifications to the code were made over a number of years as particular requirements were encountered. Changes fell into four categories: cosmetic, input modification, output modification, and physical model enhancements. The changes that have been made are summarized in this section.

3.1 OVERVIEW

3.1.1 Cosmetic Changes

When convenient, cosmetic changes [substituting 'do' loops for 'if' tests, substituting Fortran77 'if-then-else' syntax for 'if (...) go to xx' syntax] were made during the course of changes in the code. The objective of such changes was to increase readability and efficiency while producing a program that gave solutions that were identical to those obtained before the changes were made.

The most extensive cosmetic changes were made in the solution of the pressure correction equation. SOLA-VOF was originally written for the Control Data Corporation CDC-7600 computing system. That system rewarded programs that used small memory arrays. To exploit this trait, the pressure correction routine performed an 'if test' on every face of every computational cell during every iteration of the pressure correction. This arrangement not only repeated the tests for every iteration but inhibited hardware efficiencies by requiring multiple branching logic to be executed for every iteration. In the last decade, computer memory has fallen drastically in price and the concept of 'virtual memory' has been incorporated into operating systems. Currently, huge arrays may be used in codes without imposing the penalties encountered when using the CDC-7600. Exploiting these changes in technology, the results of the 'if' tests performed on each cell face were captured in four arrays (four words of memory for each computational cell) before the pressure iteration process was started. The benefit for this change is proportional to the number of iterations required, but in practice yields about a 30% reduction in processor time together with identical results for the pressure corrections.

A more visible but less extensive cosmetic change was introduced in the main program. The major processing loop was originally programmed with multiple intertwined 'if' tests, making reading and modification difficult. One major 'do loop' was substituted for several of the 'ifs,' highlighting the top-down nature of that section of code and easing the insertion of multiple output options.

Other less extensive cosmetic changes appear throughout the new coding.

3.1.2 Output Changes

Changes in output available have been introduced in the areas of graphics, timing, and tabular output. Input and output file names and purposes are listed in Table 1.

Table 1. Input and output files used by SOLA-VOF

File name	Type	Input/output	Function
soldata	text	input	Initial conditions, mesh generation, fluid properties, boundary conditions, output and restart control, iteration and time step control.
solprt.out	text	output	Echo input, periodic time and iteration statistics, occasional field variable (velocities, pressure) output, final timing summary.
vofile.out	binary	output	Image of all common blocks; with the appropriate ‘soldata’ changes, contains sufficient information to continue a previous calculation.
vofile.in	binary	input	Restart file for continuing calculations; normally created by renaming ‘vofile.out’ to ‘vofile.in.’
startstop.in	text	input	A file one character in length. The program reads this file at intervals. If the character = ‘s’ or ‘S’, the calculation is terminated.
	text	standard output	Periodic record of time and iteration information, printer plots of fluid surface or interface shape, sampling of acceleration parameters.
history.out	text	output	A record of the flow variables at specified points in the flow field.
tecfile.out	text	output	Flow variables at the corner of every computational cell; formatted for input into Tecplot for visualization of the solution.
monitor.out	text	output	At selected time step increments, prints one line of timing and iteration information.

3.1.2.1 Graphics Output

As received, the code contained a graphics capability implemented with specific calls to a Stromberg-Carlson cathode ray tube device (the SC-4020). These calls and the fortran used to fill the plotting arrays have been removed. A variety of commercial graphics packages are available that require only ASCII text files as input. These packages are designed to be used interactively after the CFD calculation has been finished. Therefore, the original graphics calls have been replaced with short routines that produce various ASCII files depending on the setting of input parameters. Header information in the files is structured specifically for the commercial program Tecplot.*

In addition, a very crude "printer plot" is generated and placed in its own output file at a specified increment of problem time. By monitoring this file, the user can obtain a rough idea of surface shape. It also allows a quick visual check of boundary condition settings so that the calculation can be stopped if an incorrect specification has been made.

3.1.3 Input

Input is provided from two sources. Initial conditions, including fluid conditions, mesh structure, and boundary conditions, as well as control variables involving output timing, etc., are provided by reading a file named **soldata** from "standard input." Restart of a calculation from a previous run is accomplished by reading a restart file named **vofile.in**. Input and output file names and purposes are listed in Table 1.

The 'namelist' input technique used by Nichols, Hirt, and Hotchkiss[2] has been expanded to allow more control over the problem definition without custom coding. Variables have been defined to control frequency of tabular output, graphical output, and problem duration. Arrays have been defined and coding modified to allow a mix of boundary conditions to be applied on each of the four problem edges. Namelists have been added to allow definition of obstacles embedded in the flow field and to allow some initial condition specification without custom coding.

3.2 RESTART CONTROL AND ITERATION CONTROL

A restart capability has been added to SOLA-VOF. This allows simulations to be stopped, the intermediate solution examined, and the calculation continued if desired. Whenever a simulation reaches a normal end, either by covering a specified time span, by covering a specified number of time steps, or by encountering a "stop" instruction, the contents of the common blocks are written to a disk file in the directory in which the problem is running. This file is named "**vofile.out**." If the computer operating system is case sensitive (most versions of UNIX for example), the file must be lower case.

The restart flag is specified in the namelist *abegin*. Besides specifying the incremental physical time to be simulated and the number of time steps to be added to the solution (the lesser of these determines when the problem is terminated), a simulation can be stopped by editing the file **startstop.in** in the directory in which the problem is running. This filename must be lower case. The file consists of one character. If that character is "s" or "S" when the file is read by the program, output files will be written and closed, the restart file

---

*A product of Amtec Engineering, Inc., Bellevue, WA.

vofile.out will be written and closed, and the program will terminate execution. This of course requires

that the operating system under which SOLA-VOF is running be multi-tasking. For example, UNIX, LINUX, Windows (95, 98, and NT) are multi-tasking systems.

To restart a simulation from a previous run, the output file **vofile.out** from that run should be renamed to **vofile.in**, the character in the file **startstop.in** should be changed to "q" (for example), and the restart flag in name list *abegin* set to "1."

3.3 SETTING BOUNDARY CONDITIONS

All of the boundary condition options described in the original LANL report are available. In VOLVOF, the capability of splitting boundary conditions along boundaries has been added. For example, portions of a boundary may be blocked, creating a wall or hole to allow inflow or outflow of various types, depending on the details of the boundary flags set in the namelist input. Use of these flags is described in Appendix B.

3.4 OUTPUT MANAGEMENT

To allow monitoring of variable changes at specific points on the mesh, history files may be established using the namelist *history*. Variables to be monitored as well as the locations and the times are defined in the namelist. ASCII files are created that may be viewed, edited, and plotted to give a visual assessment of the convergence of the calculation or the time-dependent development of the flow.

The file *solprt.out* is a comprehensive record of a computer run. The input files are echoed, the initial and boundary conditions and periodic snapshots of the flow field are recorded. Also, periodic information about the pressure iteration, the time step being used, and other flow development information is reported. Finally, a summary table of the iterations, computer time, and wall clock time is printed. The file is ASCII and can be examined using a text editor.

The file *monitor.out* contains a single line of timing and iteration information for particular time steps. The file is ASCII to allow the user to monitor the calculation during execution.

An ASCII file *tecfile.out* is produced periodically. Sufficient information is provided to produce contour plots of stream function, vorticity, and pressure, as well as mesh velocity vector plots. This file is formatted with the proper headers for input to the Tecplot* graphics program. Using a text editor, this file is easily modified to meet the input specifications of other commercial graphics programs.

A monitoring file is also written to the "standard output" of the computing system being used. This file contains some information about the pressure iteration and the time step being used as well as a periodic coarse character-based plot of the current flow configuration.

The update frequency of each of these files is controlled by namelist variables specified by the user, allowing a balance to be struck between file size and timeliness of the output.

---

*A production of Amtec Engineering Inc., Bellevue, Washington.

4. DEMONSTRATION PROBLEMS

4.1 RECIRCULATING FLOW DRIVEN BY A MOVING WALL - THE DRIVEN CAVITY PROBLEM

The flow in a closed cavity, driven by the motion of one wall across the cavity, has been extensively studied for many years. A comprehensive study[12] illustrates the complexity of the flows that develop in numerical solutions to the problem. Results for one driven cavity problem, a square cavity with a Reynolds Number of 100, are presented here as a demonstration problem.

The stream function ø is given by

$$u = \frac{\partial \phi}{\partial z}, \quad w = -\frac{\partial \phi}{\partial x} \tag{5}$$

The stream function has the property (in two dimensions) that it satisfies steady-state continuity equation (Eq. 4) identically; that is

$$\nabla \cdot \bar{v} = \frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = \frac{\partial}{\partial x}\frac{\partial \phi}{\partial z} - \frac{\partial}{\partial z}\frac{\partial \phi}{\partial x} = 0 \tag{6}$$

In addition, the velocity vector $\bar{v}$ is everywhere tangent to curves of constant value of the stream function (*streamlines*). Thus contour plots of the stream function give a rapid qualitative view of a two-dimensional flow field.

Contour plots of the fluid vorticity also give a rapid qualitative view of a two-dimensional flow field; in two dimensions, the vorticity is given by

$$\omega = |\nabla \times \bar{v}| = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \tag{7}$$

The fluid, with a kinematic viscosity ν, fills a cavity that is a square R units on a side. The lid of the cavity moves across the cavity with a velocity u. The Reynolds Number for this problem is

$$Re = \frac{uR}{\nu}, \; with \; R = 1.0, \; u = 1.0, \; \nu = 0.01, \; Re = 100 \tag{8}$$

Solutions were obtained on a 20 × 20 uniform mesh (cell size = 0.05 × 0.05), a 40 × 40 uniform mesh (cell size = 0.025 × 0.025), and 50 × 50 non-uniform mesh (mesh spacing at the walls = 0.01 and gradually growing to 0.0308 at the center of the cavity). The problem was also solved for the 20 × 20 uniform mesh

using the commercial CFD code CFX 4.1.* In addition, midplane velocity profiles from the solution of Ghia, Ghia, and Shin[12] (129 × 129 cells, uniform mesh) are available for comparison.

4.1.1 Input

The input file for the 50 × 50 cell problem is reproduced on the following page. The problem length is specified to be 10 time units, with graphics output (*tecfile.out*) to occur every 2 time units. A complete listing of the solution (*solprt.out*) is requested only at the end (after 10 time units).

4.1.2 Results

Solutions for the 20 × 20 cases are presented in the next two figures; Fig. 2 shows velocity vectors from the CFX solution. Figure 3 shows the stream lines for the VOLVOF case. Clearly these solutions are very similar, with the "center" of the circulating fluid mass located at the approximate coordinates x = 0.60, z = 0.75. The streamlines in the lower left and right corners show rough counter-rotating circulation cells that normally develop in the driven cavity simulation. The cells also appear in the tabular output from CFX, but the vectors are too small to appear on the graphical output.

Figures 4!5 show the stream lines for the 40 × 40 and 50 × 50 cell cases. The streamlines are progressively smoother and the corner circulation cells are much smoother. Overall, the 50 × 50 cell solution is visually much more refined than the 20 × 20 solutions (Figs. 2!3).

Figures 6! 8 are comparisons of several scalar quantities across the horizontal midplane (z = 0.5, 0.0 < x < 1.0) for each of the mesh resolutions. Taken together, the comparisons show a consistent convergence of the solutions as the mesh is refined.

Figure 6 shows the vertical velocity at the midplane. First, the x coordinate at which the velocity changes sign is virtually the same for each solution. The two 20 × 20 cell solutions are somewhat different, with the CFX solution showing smaller peaks than the 20 × 20 cell VOLVOF solution. This damping of the peaks suggests that the CFX solution is more diffusive for the discretization scheme selected for this CFX run. The 20 × 20, 40 × 40, and 50 × 50 cell solutions differ by smaller amounts as the number of cells increases. Individual velocity values from the Ghia, Ghia, and Shin[12] solution are included for comparison. Clearly, all of the vertical velocity solutions are qualitatively the same. The 40 × 40 and 50 × 50 cell solutions are nearly identical and essentially the same as the Ghia et al solutions.

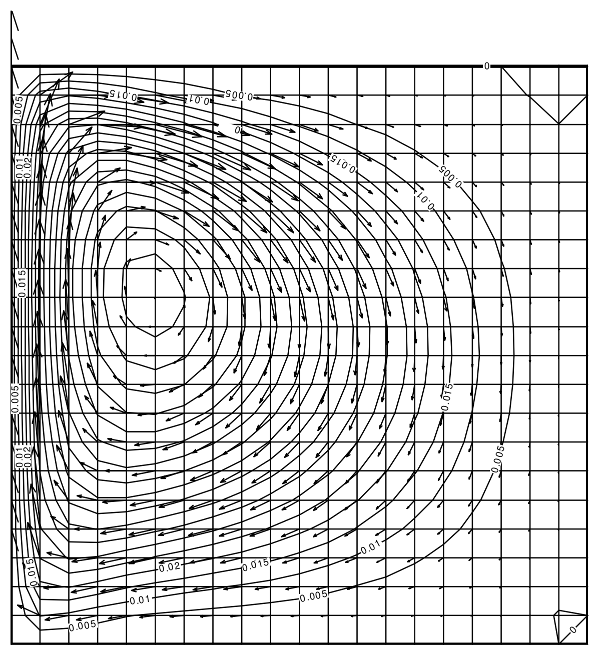
Figure 7 shows the midplane values for the stream function that was obtained using VOLVOF. The convergence of the sequence of solutions is evident here also.

---

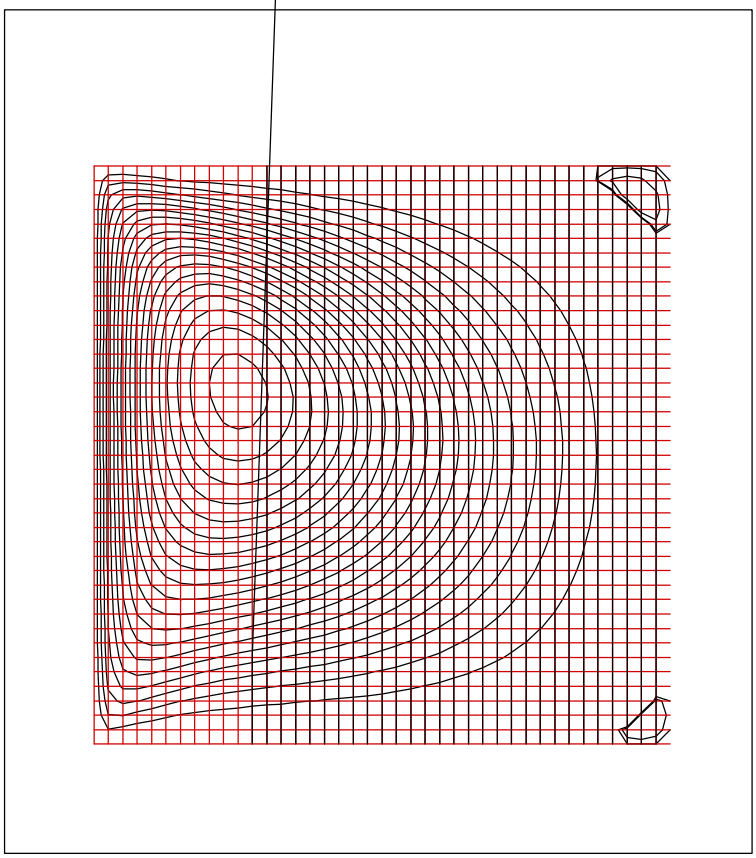
*A product of AEA Technology plc.

Input File for Driven Cavity Problem

```
Driven Cavity Problem - Re = 100 - nonuniform mesh - check case 8/27/97
 &abegin
 nresta=0,   incycl=6001,   deltim=10.0,   inquir=10,  fsaft=0.8d0,
/  &end
&xput
 alpha=1.0,          autot=1.0,       cangle=90.0,       csq=-1.0,
 delt=.002,          epsi=1.0e-4,     flht=1.1,          gx=0.0,
 gy=0.0,             icyl=0,          isymplt=0,
 nmat=1,             isurf10=0,       npx=0,             npy=0,
 omg=1.8,            pltdt=2.00,      prtdt=10.0,        sigma=0.0,
 twfin=10.0,         ui=1.0,          wi=0.0,
 nwb=52*2,           nwl=52*2,        nwt=52*2,          nwr=52*2,
 xpl=0.49,           xpr=0.51,        ypb=0.01,          ypt=0.99,
 nu=0.010,           rhof=1.0,        rhofc=1.0,         uln=52*0.0,
 wbn=52*0.0,         wtn=52*0.0,      utt=52*1.0,        ubt=52*0.0,
 wlt=52*0.0,         wrt=52*0.0,
/  &end
&mshset
 nkx=2,     xl=0.0,      0.5, 1.0,
            nxl=1,        24,
            nxr=24,        1,
            xc=0.01,   0.99,
            dxmn=0.01, 0.01,
 nky=2,     yl=0.0,      0.5, 1.0,
            nyl=1,        24,
            nyr=24,        1,
            yc=0.01,   0.99,
            dymn=0.01, 0.01,
/  &end
 &obsics
 moblk=0,
 mincs=0,
/   &end
&history
 mhist=1, nhist=101, ihist(1)=21, jhist(1)=26,
/  &end
```



0
0.005
0.01
0.015
0.02



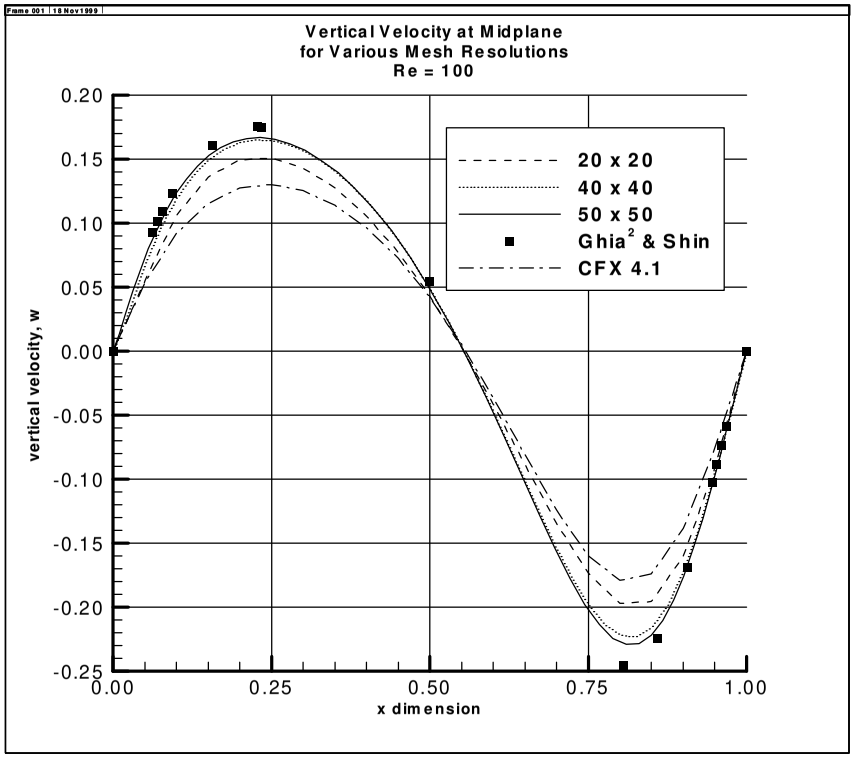


Fig. 6. Midplane axial velocity from VOLVOF at various mesh resolutions, from CFX, v. 4.1, and from Ghia, Ghia, and Shin (1982).

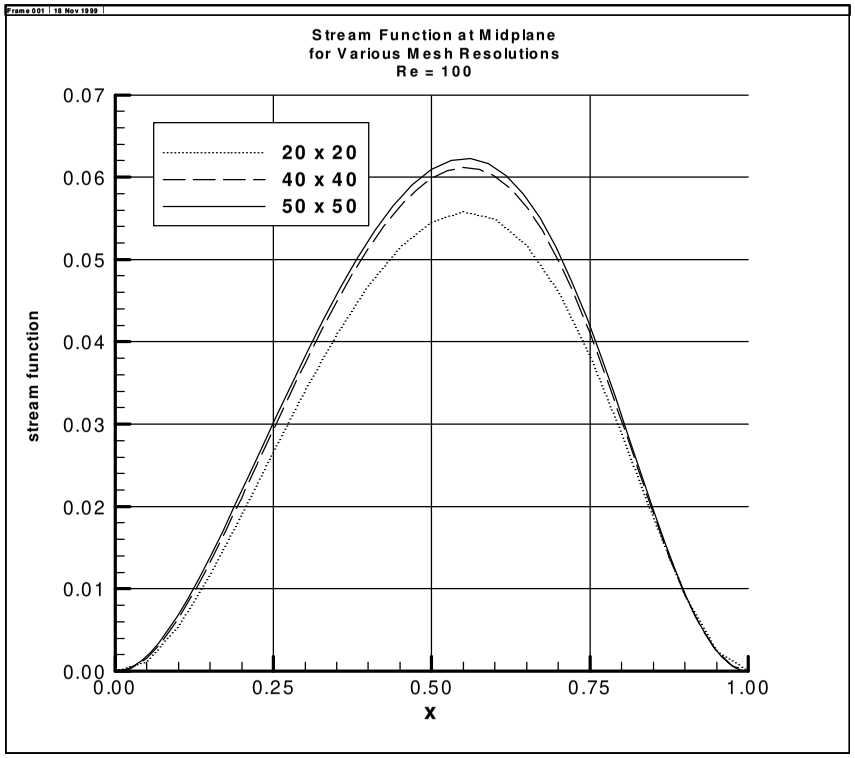


Fig. 7. Midplane values of stream function from VOLVOF at various mesh resolutions.

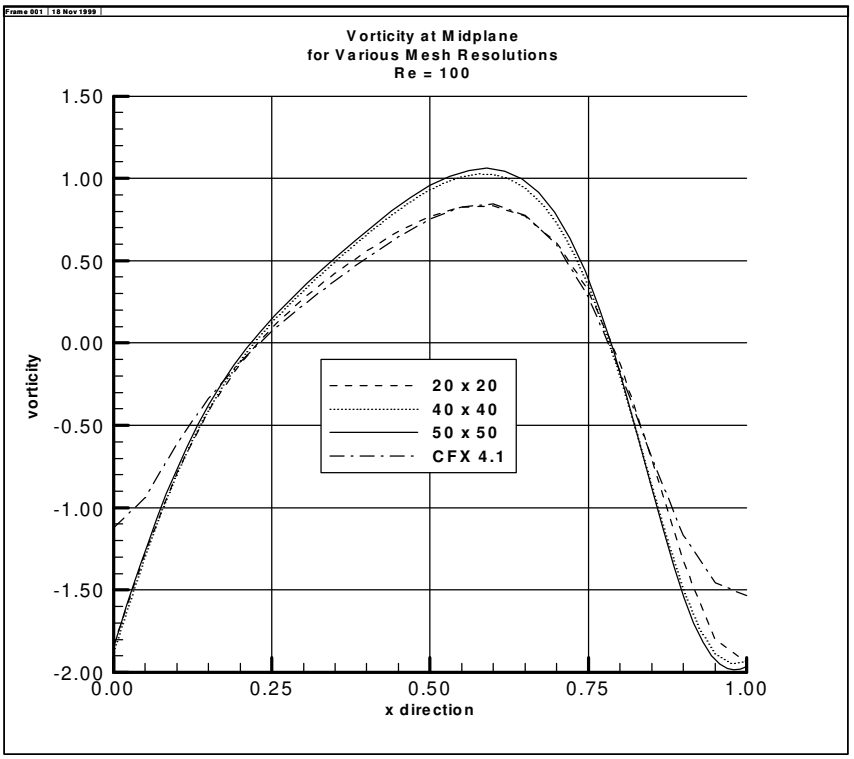


Fig. 8. Midplane values of vorticity from VOLVOF and CFX v. 4.1.

Stream function results from CFX require user-programmed post processing of the numerical solution, and satisfactory results were not obtained. This is not a flaw in the CFX product.

Figure 8 shows the vorticity calculated at the horizontal midplane of the driven cavity. Again, the convergence to a well-resolved solution as the mesh is refined is evident. The vorticity curve for the CFX results matches the VOLVOF results for the 20 × 20 cell solution except at the walls. Again, the details of the CFX algorithm prevented an accurate determination of the vorticity at the walls. Since CFX does not use the vorticity as a part of its solution schemes, this is not a problems for CFX users.

Figures 9–10 show contours of the fluid pressure and vorticity from the VOLVOF 50 × 50 cell solution. These are included for completeness.

In summary, three VOLVOF solutions to the flow in a square driven cavity at a Reynolds number of 100 have been presented. The solutions show a consistent convergence towards a mesh-free solution for the problem. The coarse mesh solution (20 × 20 cells) compares favorably with a solution on the same mesh obtained using the commercial CFD code CFX. The fine mesh solution (50 × 50) compares favorably with a solution obtained by Ghia, Ghia, and Shin.[12]

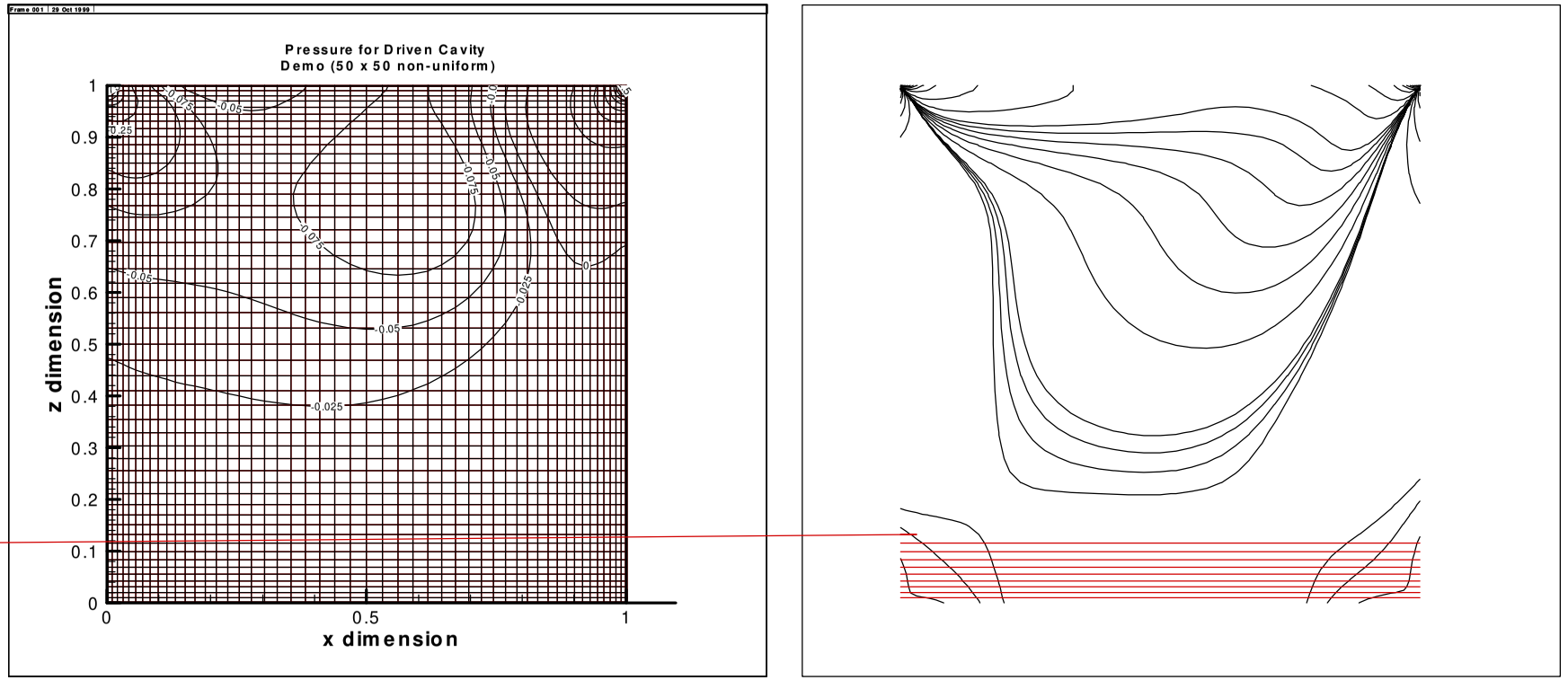


Fig. 9. Relative pressure contours in the driven cavity from VOLVOF.

4.2 DETERMINING THERMOCOUPLE IMMERSION SEQUENCE USING COMPUTATIONAL FLUID DYNAMICS

A series of experiments was run to determine the surface heat transfer rates during quenching of a hot, instrumented cylinder fabricated from nickel tubing.[13] The cylinder is heated to 800EC in the oven and then dropped into the cold (20EC) water in the barrel beneath the oven. Data from the thermocouples are recorded during the rapid cooling (15 s from release to cessation of boiling).

In order to interpret the measurements, the order of wetting of the thermocouples must be known. The interaction of the water with the moving cylinder determines if the thermocouples on the surface of the cylinder are wetted simultaneously, sequentially from bottom to top or, possibly, sequentially from top to bottom. The present simulation of the immersion revealed the order of wetting.

The problem is axisymmetric about the axis of the cylinder and barrel. The computing mesh is shown on the reverse side. The mesh has been concentrated along the edges of the cylinder. Rather than integrate the equations of motion of the cylinder and its deceleration as it hits the water, a constant inlet velocity was assigned to the water entering the barrel and around the fixed cylinder. The water was assigned an upward velocity of 2.44 m/s, the velocity that the cylinder would have if dropped from a height of 2 ft (0.61 m) before hitting the water.

The input data for this demonstration problem is shown on the following page. A mesh of 20 cells radially (plus two image cells) by 41 cells axially (also with image cells added top and bottom) is set up. The mesh is graduated to give the greatest resolution (finest spacings) along the bottom and along the outer wall of the test cylinder. The cylinder is treated as an obstacle, defined to include cells whose centers are numbered (1,16) through (8,36). Note that, in the input file, the obstacle is defined by the upper right corner of the specified cells.

4.2.1 monitor.out

The file showing summary information useful in monitoring the progress of the flow development is reproduced on the following page. Each line contains the cycle number at which the line was printed, the number of iterations required to converge the pressure equation for that cycle, the total processor time used to that point in the run, the problem simulation time, the computational time step (dt) for that cycle, the contents of the *startstop.in* file, the radial and axial indices for the cell dictating the minimum size time step, and a code indicating the governing stability criterion. The codes are: 1 => convection limited in radial direction, 2 => convection limited in the axial direction, 3 => diffusion limited, 4 => surface tension limited.

From the *monitor.out* file, the CPU time used was over 2900 s. This was on an obsolete IBM RISC/6000-320 workstation. On a DEC 500au Personal Workstation, the calculation required about 57 s. By contrast, a single experiment required several days to set up and execute.

Input File for Cylindrical Splash Problem

```
 Splashing the cylinder into a barrel of water
&abegin
 nresta=0, incycl=4500, deltim=2.0d-3, inquir=80, fsaft=0.8d0,
 ifstf=1,
/   &end
&xput
 alpha=1.0d0,   autot=1.0d0,   cangle=90.0d0,   csq=-1.0d0,
 delt=2.0d-3,   epsi=1.0d-3,   flht=2.919d2,    gx=0.0d0,
 gy=-9.8d3,     icyl=1,        isymplt=0,       nmat=1,
 isurf10=0,     npx=0,         npy=0,           omg=1.8d0,
 pltdt=.025,    prtdt=0.05,    sigma=0.0d0,     twfin=0.35,
 ui=0.0d0,      wi=2444.2,
 xpl=0.0d1,     xpr=2.70d2,    ypb=2.7d2,       ypt=2.85d2,
 nu=1.0d0,      rhof=1.0d-3,   rhofc=0.0d0,
 nwl=43*1,      nwr=43*1,      nwt=43*3,        nwb=43*1,
 wbn=43*2444.2,
/   &end
&mshset
 nkx  = 2,
 xl   =   0.0d0, 76.0d0, 280.0d0,
 xc   =  68.0d0, 84.0d0,
 nxl  =    6,      1,
 nxr  =    1,     12,
 dxmn =    8.0d0,  8.0d0,
 nky  = 3,
 yl   = 0.0d0,  3.0d2,  6.04d2, 8.5d2,
 yc   = 2.92d2, 3.08d2, 6.29d2,
 nyl  =   14,      1,      1,
 nyr  =    1,     18,      6,
 dymn = 8.0d0,  8.0d0, 25.0d0,
/   &end
&obsics
 moblk    = 1,
 iobll(1) = 1, jobll(1)=16, iobur(1)=8,  jobur(1)=36,
 mincs    = 1,
 iicll(1) = 1, jicll(1)=2,  jicur(1)=15, icvar(1)=2, varic(1)=2444.2,
/   &end
&history
 mhist=0,
/   &end
```

monitor.out File for Cylindrical Splash Problem

```
date and time are Fri May 22 15:13:20 1998
 Cy=    80/ it=   25/ CPU=    74.0/ PTime= 1.3748D-02/ dt= 1.18D-04/q   9  15 3
 Cy=   160/ it=   17/ CPU=   122.8/ PTime= 2.3140D-02/ dt= 1.20D-04/q   9  15 3
 Cy=   240/ it=   22/ CPU=   172.3/ PTime= 3.3936D-02/ dt= 1.49D-04/q   9  15 3
 Cy=   320/ it=  110/ CPU=   223.7/ PTime= 4.7782D-02/ dt= 1.97D-04/q   9  15 3
 Cy=   400/ it=   22/ CPU=   278.2/ PTime= 6.7336D-02/ dt= 2.87D-04/q   9  15 3
 Cy=   480/ it=   40/ CPU=   333.6/ PTime= 9.2485D-02/ dt= 3.42D-04/q   9  15 3
 Cy=   560/ it=   19/ CPU=   389.5/ PTime= 1.2431D-01/ dt= 4.08D-04/q   9  15 3
 Cy=   640/ it=   27/ CPU=   445.9/ PTime= 1.5993D-01/ dt= 4.96D-04/q   9  15 3
 Cy=   720/ it=  162/ CPU=   618.4/ PTime= 1.9525D-01/ dt= 3.12D-04/q   9  15 3
 Cy=   800/ it=   13/ CPU=   820.0/ PTime= 2.0755D-01/ dt= 9.05D-05/q   9  15 3
 Cy=   880/ it=  181/ CPU=   978.1/ PTime= 2.1582D-01/ dt= 3.29D-05/q   9  15 3
 Cy=   960/ it=    3/ CPU=  1072.7/ PTime= 2.1832D-01/ dt= 3.85D-05/q   9  15 3
 Cy=  1040/ it=    5/ CPU=  1137.0/ PTime= 2.2214D-01/ dt= 5.23D-06/q   9  15 3
 Cy=  1120/ it=    4/ CPU=  1158.4/ PTime= 2.2279D-01/ dt= 1.16D-05/q   9  15 3
 Cy=  1200/ it=    4/ CPU=  1179.8/ PTime= 2.2421D-01/ dt= 2.57D-05/q   9  15 3
 Cy=  1280/ it=    9/ CPU=  1214.6/ PTime= 2.2716D-01/ dt= 4.66D-05/q   9  15 3
 Cy=  1360/ it=   10/ CPU=  1328.4/ PTime= 2.3082D-01/ dt= 1.80D-05/q   9  15 3
 Cy=  1440/ it=    7/ CPU=  1375.6/ PTime= 2.3253D-01/ dt= 2.84D-05/q   9  15 3
 Cy=  1520/ it=    5/ CPU=  1402.0/ PTime= 2.3590D-01/ dt= 6.06D-05/q   9  15 3
 Cy=  1600/ it=  229/ CPU=  1777.3/ PTime= 2.3963D-01/ dt= 3.13D-08/q   9  15 3
 Cy=  1680/ it=   43/ CPU=  1989.3/ PTime= 2.3963D-01/ dt= 1.58D-08/q   9  15 3
 Cy=  1760/ it=   23/ CPU=  2061.6/ PTime= 2.3963D-01/ dt= 1.29D-08/q   9  15 3
 Cy=  1840/ it=    2/ CPU=  2096.0/ PTime= 2.3963D-01/ dt= 2.69D-08/q   9  15 3
 Cy=  1920/ it=    7/ CPU=  2162.9/ PTime= 2.3963D-01/ dt= 2.70D-08/q   9  15 3
 Cy=  2000/ it=   24/ CPU=  2208.7/ PTime= 2.3964D-01/ dt= 4.17D-08/q   9  15 3
 Cy=  2080/ it=   14/ CPU=  2252.1/ PTime= 2.3964D-01/ dt= 8.03D-08/q   9  15 3
 Cy=  2160/ it=    2/ CPU=  2284.5/ PTime= 2.3965D-01/ dt= 1.78D-07/q   9  15 3
 Cy=  2240/ it=    2/ CPU=  2306.7/ PTime= 2.3967D-01/ dt= 3.95D-07/q   9  15 3
 Cy=  2320/ it=    2/ CPU=  2328.1/ PTime= 2.3972D-01/ dt= 8.75D-07/q   9  15 3
 Cy=  2400/ it=    4/ CPU=  2349.0/ PTime= 2.3983D-01/ dt= 1.94D-06/q   9  15 3
 Cy=  2480/ it=    4/ CPU=  2369.8/ PTime= 2.4007D-01/ dt= 4.30D-06/q   9  15 3
 Cy=  2560/ it=    5/ CPU=  2390.6/ PTime= 2.4060D-01/ dt= 9.53D-06/q   9  15 3
 Cy=  2640/ it=    3/ CPU=  2411.8/ PTime= 2.4177D-01/ dt= 2.11D-05/q   9  15 3
 Cy=  2720/ it=    8/ CPU=  2433.5/ PTime= 2.4436D-01/ dt= 4.68D-05/q   9  15 3
 Cy=  2800/ it=   12/ CPU=  2463.1/ PTime= 2.5012D-01/ dt= 1.04D-04/q   9  15 3
 Cy=  2880/ it=   14/ CPU=  2518.5/ PTime= 2.6090D-01/ dt= 1.77D-04/q   9  15 3
 Cy=  2960/ it=   17/ CPU=  2574.0/ PTime= 2.7843D-01/ dt= 2.74D-04/q   9  15 3
 Cy=  3040/ it=   16/ CPU=  2740.9/ PTime= 2.9700D-01/ dt= 5.52D-05/q   9  15 3
 Cy=  3120/ it=   14/ CPU=  2861.1/ PTime= 2.9902D-01/ dt= 1.75D-05/q   9  15 3
 Cy=  3153/ it=  627/ CPU=  2913.8/ PTime= 1.0000D+10/ dt= 1.09D-05/q   9  15 3
 date and time are Fri May 22 16:06:09 1998
```

On the following four pages, character maps of the solution are shown as the cylinder penetrates the water; the elapsed time since release of the cylinder is given at the top of each frame. Clearly the surface is swamped at about 0.15 s. Thus, the simulation reveals that the thermocouples were all inundated with cold water within about 0.025 s. With a data-logging frequency of 250 Hz/sensor (0.004 s between points), this has been confirmed experimentally.

The character maps consist of one character for each computational cell, arranged to correspond to the mesh indices. The characters correspond to the cell type at each location. For each image cell, the boundary condition code is shown. For the present example, the boundaries at the left, right, and bottom of the maps are "1" for specified normal velocity. The condition "3" at the top corresponds to a continuative boundary. The cylinder, an `obstacle' to the flow, is set out by the `B' characters (for "blocked"). The cells marked by the "F" are full of fluid, while the cells that are blank (" ") indicate either empty cells or, if a two-fluid problem is being run, cells that are filled with the second fluid. Surface cells, cells dividing two fluids or dividing the full cells from empty cells, are marked with an "s."

Figure 11 is a composite of the flow configuration at several times during the transient. The surface location is more precise, and velocity vectors indicate the direction as well as the magnitude of the fluid motion. Further, the pictures are proportionate to the actual experimental equipment. This figure was produced using the commercial graphics program Tecplot based on the code output in the *tecfile.out* file.

```
cycle/time =   0 0.000   cycle/time = 176 0.025   cycle/time = 331 0.050

3333333333333333333333   3333333333333333333333   3333333333333333333333
1                    1   1                    1   1                    1
1                    1   1                    1   1                    1
1                    1   1                    1   1                    1
1                    1   1                    1   1                    1
1                    1   1                    1   1                    1
1                    1   1                    1   1                    1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB             1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB    ssss     1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB   sFFFFssss 1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB   sFFFFFFFFs1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB   sFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB             1   1BBBBBBB   sFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB  sssss      1   1BBBBBBB   sFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB  sFFFFsssss 1   1BBBBBBB   sFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB sFFFFFFFFFFs1   1BBBBBBB   sFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB ssFFFFFFFFFF1
1BBBBBBB             1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB             1   1BBBBBBBsFFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB             1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBB             1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB             1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1FFFFFFFsssssssssssss1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1111111111111111111111   1111111111111111111111   1111111111111111111111
```

```
cycle/time = 426 0.075
33333333333333333333
1                    1
1                    1
1                    1
1                    1
1                    1
1                    1
1BBBBBBB             1
1BBBBBBB             1
1BBBBBBB             1
1BBBBBBB     sss     1
1BBBBBBB    sFFFssss 1
1BBBBBBB    sFFFFFFFs1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
```

```
cycle/time = 501 0.100
33333333333333333333
1                    1
1                    1
1                    1
1                    1
1                    1
1                    1
1BBBBBBB      ssss   1
1BBBBBBB     sFFFFsss1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
```

```
cycle/time = 562 0.125
33333333333333333333
1                    1
1                    1
1                    1
1                    1
1             sssssss1
1            sFFFFFFF1
1BBBBBBB     sFFFFFFF1
1BBBBBBB     sFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB    sFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1
1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
```

```
cycle/time = 620 0.150   cycle/time = 670 0.175   cycle/time = 737 0.200

3333333333333333333333   3333333333333333333333   3333333333333333333333
1                    1   1               sFFFF1   1             sFFFFFF1
1                    1   1             ssFFFFF1   1           ssFFFFFFF1
1             sssssss1   1            sFFFFFFF1   1          sFFFFFFFFF1
1            sFFFFFFF1   1           sFFFFFFFF1   1         sFFFFFFFFFF1
1            sFFFFFFF1   1           sFFFFFFFF1   1       ssFFFFFFFFFFF1
1            sFFFFFFF1   1          sFFFFFFFFF1   1      sFFFFFFFFFFFFF1
1BBBBBBB    sFFFFFFFF1   1BBBBBBB  sFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB    sFFFFFFFF1   1BBBBBBB  sFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1   1BBBBBBB  sFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB   sFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB  sFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1   1BBBBBBB sFFFFFFFFFFF1
1BBBBBBBsFFFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1   1BBBBBBBsFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1   1BBBBBBBFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1   1FFFFFFFFFFFFFFFFFFFF1
1111111111111111111111   1111111111111111111111   1111111111111111111111
```

```
cycle/time =1227 0.225
33333333333333333333
1           sFFFFFFFFF1
1    ssssssFFFFFFFFFF1
1    sFFFFFFFFFFFFFF1
1    sFFFFFFFFFFFFFF1
1     sFFFFFFFFFFFFF1
1     sFFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
cycle/time =2799 0.250
33333333333333333333
1  sFFFFFFFFFFFFFFFF1
1  sFFFFFFFFFFFFFFFF1
1   sFFFFFFFFFFFFFFF1
1    sFFFFFFFFFFFFFF1
1     sFFFFFFFFFFFFF1
1      sFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
cycle/time =2947 0.275
33333333333333333333
1  sFFFFFFFFFFFFFFFF1
1  sFFFFFFFFFFFFFFFF1
1   ssFFFFFFFFFFFFFF1
1     sFFFFFFFFFFFFF1
1      sFFFFFFFFFFFF1
1      sFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1BBBBBBBFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
1FFFFFFFFFFFFFFFFFFF1
11111111111111111111
```

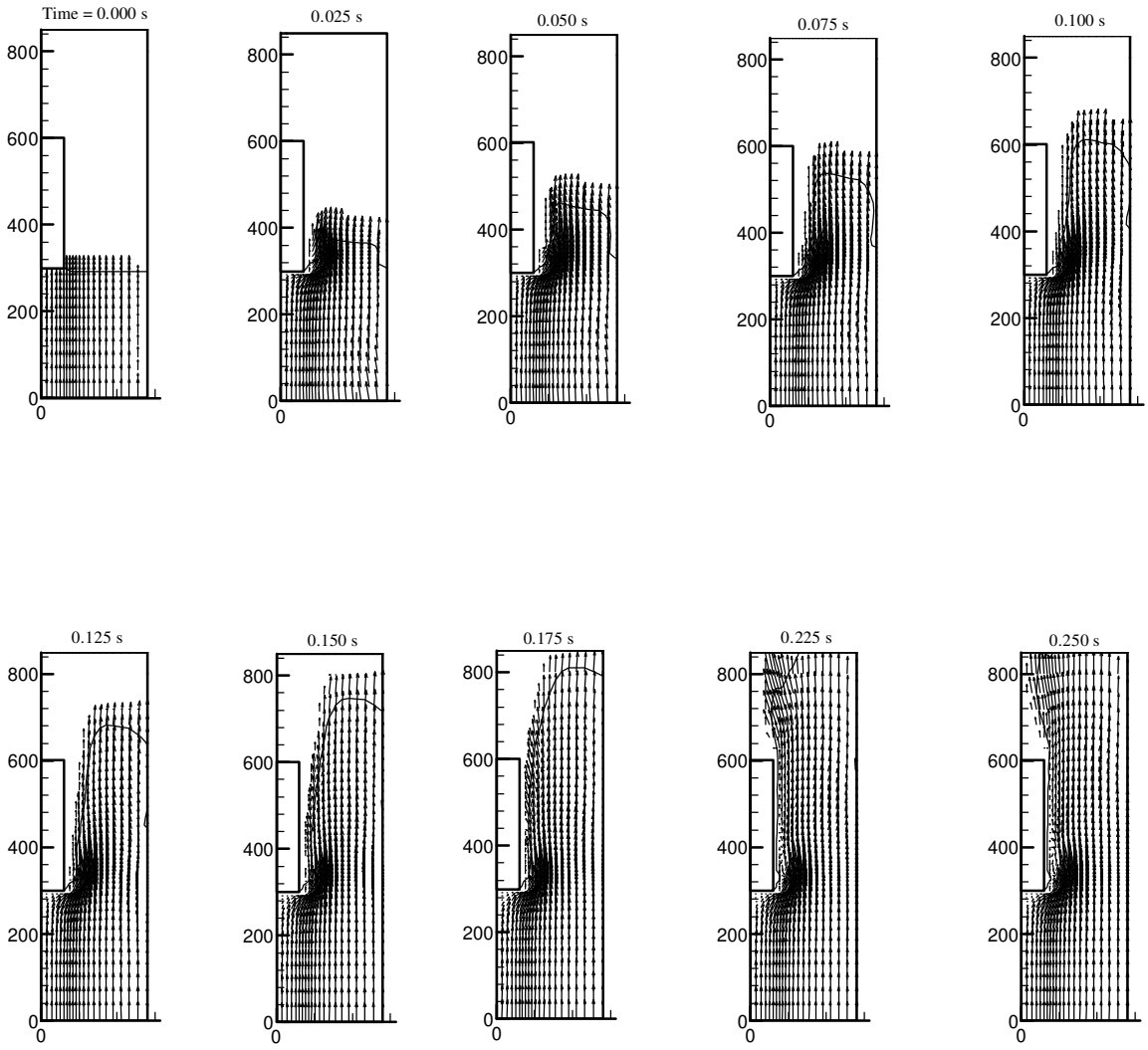


Fig. 11. Fluid configuration at several times following release of the cylinder into the quench tank.

5. CONCLUSIONS

A partial upgrading and modernization of the computational fluid dynamics code SOLA-VOF is reported. Two test cases demonstrate that the present configuration of the code is stable and useful:

1. The well known driven cavity problem has been solved on three distinct finite difference meshes. The results on one grid compare favorably with results using the same grid and the commercial CFD code CFX. The results on a different grid compare well with a fine-grid solution obtained in 1982 by Ghia, Ghia, and Shin.[12]

2. A second, transient, problem simulates the entry of an instrumented cylinder into a barrel of water. Details of wave action within the barrel during the entry (the cylinder was dropped from an elevation of two feet above the water) are presented graphically. The elapsed time and the order in which thermocouples attached to the cylinder are wetted by the waves agree with data recorded during an experiment.

Therefore, the CFD code VOLVOF is stable and a useful vehicle for an interested individual to employ as an alternative to commercial CFD codes.

ACKNOWLEDGMENTS

T. D. Butler, A. A. Amsden and other members of the T-3 group at the Los Alamos National Laboratory have been generous with their time and advice over a period of twenty five years. The “friendly user” source from which the present code began was provided by Tony Amsden. Dr. Larry Cloutman of the Lawrence Livermore National Laboratory has served as consultant-without-fee, friend, and critic for a similar time period. Dr. C. W. Hirt, president of Flow Science, Inc, has generously shared his CFD experience and his opinions on current CFD usage and practice over many years.

The support of D. M. Hetrick, manager of the Computational Modeling and Simulation Section, Computational Physics and Engineering Division, Oak Ridge National Laboratory, in providing the funds to prepare this documentation is gratefully acknowledged.

REFERENCES

1. C. W. Hirt, "Simplified Solution Algorithms for Fluid Flow Problems," proc. *Numerical Methods for Partial Differential Equations Seminar*, Univ. of Wis., Academic Press, Inc., 1978.

2. B. D. Nichols, C. W. Hirt, and R. S. Hotchkiss, "SOLA-VOF: A Solution Algorithm for Transient Fluid Flow with Multiple Free Boundaries," LA-8355, Los Alamos National Laboratory, Los Alamos, New Mexico, August 1980.

3. C. W. Hirt and B. D. Nichols, "A Computational Method for Free Surface Hydrodynamics," presented at the ASME 1980 Pressure Vessel and Piping Conference, San Francisco, CA, 1980.

4. C. W. Hirt and B. D. Nichols, "Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries," *J. Comp. Physics*, **39**, pp. 201 ! 225, 1981.

5. Martin D. Torrey, Lawrence D. Cloutman, Raymond C. Mjolsness, and C. W. Hirt, "NASA-VOF2D: A Computer Program for Incompressible Flows with Free Surfaces," LA-10612-MS, Los Alamos National Laboratory, Los Alamos, NM, December 1985.

6. Martin D. Torrey, Raymond C. Mjolsness, and Leland R. Stein, "NASA-VOF3D, A Three-Dimensional Computer Program for Incompressible Flows with Free Surfaces," LA-11009-MS, Los Alamos National Laboratory, Los Alamos, NM, July 1987.

7. Douglas B. Kothe, Raymond C. Mjolsness, and Martin D. Torrey, "RIPPLE: A Computer Program for Incompressible Flows with Free Surfaces," LA-12007-MS, Los Alamos National Laboratory, Los Alamos, NM, April 1991.

8. F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow," *Phys. Fluids*, **8**, p. 2182, 1965.

9. 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 Surface," LA-3425, Los Alamos Scientific Laboratory, Los Alamos, NM, March 1966.

10. Anthony A. Amsden and Francis H. Harlow, "The SMAC Method: A Numerical Technique for Calculating Incompressible Fluids Flows," LA-4370, Los Alamos Scientific Laboratory, Los Alamos, NM, February 1970.

11.Francis H. Harlow and Anthony A. Amsden, "A Numerical Fluids Dynamics Calculation Method for All Flow Speeds," *J. Comp. Physics*, **8**, pp. 197 ! 213.

12. U. Ghia, K. N. Ghia, and C. T. Shin, "High-Re Solutions for Incompressibe Flow Using the Navier-Stokes Equations and a Multigrid Method," *J. Comp. Physics*, **48**, pp. 387 ! 411, 1982.

13. J. E. Park and G. M. Ludtka, "Calculation of Surface Heat Transfer Parameters During Quenching From Surface Thermocouple Signals," *Heat Transfer in High Energy/High Heat Flux Applications*, ASME HTD-Vol. 119, Book No. 00522, pp. 31 ! 38, 1989.

APPENDIX A. TENTATIVE DIFFERENCE EQUATIONS FOR THE AZIMUTHAL MOMENTUM EQUATION

The description of an axisymmetric rotating flow using a cylindrical coordinate system requires the addition of an azimuthal momentum equation and the addition of a centrifugal "force" term to the radial momentum equation. Specification of boundary conditions must be expanded to include the new variable. Finally, the relationship between the explicit and implicit stages of the semi-implicit algorithm must be adjusted.

At the time of this writing, the equation had been added and the centrifugal term had been added to the radial equation. However, some instability has been encountered in the solution of these coupled equations that has not yet been defeated.

The program logic has been added to impose body forces ("gravity" terms) as functions of time. This allows simulations of earthquake forces and other acceleration of the coordinate system in which the equations are solved. The acceleration terms are isolated in a small subroutine so that this specification is isolated from the complex flow of the main control program.

The momentum equations in the cylindrical coordinate system, including the azimuthal momentum equation and of the details of some of the finite difference templates used, are presented in this appendix.

A.1 Momentum Equations

The axial momentum equation (Eq. 2), has not been changed but is repeated here for completeness.

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial r} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + g_z + \nu\left[\frac{\partial^2 w}{\partial r^2} + \frac{\partial^2 w}{\partial z^2} + \frac{1}{r}\frac{\partial w}{\partial r}\right] \qquad \textbf{(9)}$$

The radial momentum equation (Eq. 3) is modified by the addition of the "centrifugal force" term, as follows:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + w\frac{\partial u}{\partial z} - \frac{v^2}{r} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + g_r + \nu\left[\frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + \nu\left(\frac{1}{r}\frac{\partial u}{\partial r} - \frac{u}{r^2}\right)\right] \qquad \textbf{(10)}$$

Finally, the equation added for the azimuthal momentum is:

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial r} + w\frac{\partial v}{\partial z} + \frac{uv}{r} = \nu\left[\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial (vr)}{\partial r}\right) + \frac{\partial}{\partial z}\left(\frac{\partial v}{\partial z}\right)\right] \qquad \textbf{(11)}$$

These equations require that the kinematic viscosity ν be a constant.

A.1.1 Finite Difference Approximation for Azimuthal Momentum Equation

Following conventional notation, variables that are evaluated on cell faces are assigned fractional subscripts. To4 Tf-0.0-0.0ORTj5on,ion4 Tf-0.F2 7.F2 7.F019019001900J 8 0 c (&) T3 113 T64 -15.12/F1/F1/ /F2 1

$$v_{i,j}^{n\%1} \text{ ' } v_{i,j}^{n} \% \ddot{a}t[\&FCOR \& FVR \& FVY \% VISTH] \tag{12}$$

$$\begin{aligned} FVR \text{ ' } (ubar/dral)([\ddot{A}x_i(\tfrac{\mathbb{M}v}{\mathbb{M}r})_{i\&½} \% \ddot{A}x_{i\&1}(\tfrac{\mathbb{M}v}{\mathbb{M}r})_{i\%½}] \\ \% \acute{a}\, sign(1.0, ubar)[\ddot{A}x_i(\tfrac{\mathbb{M}v}{\mathbb{M}r})_{i\&½} \& \ddot{A}x_{i\&1}(\tfrac{\mathbb{M}v}{\mathbb{M}r})_{i\%½}]) \end{aligned} \tag{13}$$

$$\ddot{A}x_i \text{ ' } x_{i\%1} \& x_i \tag{14}$$

$$ubar \text{ ' } (u_{i,j} \% u_{i\&1,j})/2 \; with \; dral \text{ ' } [\ddot{A}x_i \% \ddot{A}x_{i\&1}] \% \acute{a}\, sign(1.0, ubar)[\ddot{A}x_i \& \ddot{A}x_{i\&1}] \tag{15}$$

$$(\tfrac{\mathbb{M}v}{\mathbb{M}r})_{i\%½} \text{ ' } \frac{v_{i\%1} \& v_i}{\ddot{A}x_i} \tag{16}$$

$$\begin{aligned} VWL \text{ ' } (\ddot{A}x_i v_{i\&1,j} \% \ddot{A}x_{i\&1} v_{i,j})/(\ddot{A}x_i \% \ddot{A}x_{i\&1}), \\ VWR \text{ ' } (\ddot{A}x_i v_{i\%1,j} \% \ddot{A}x_{i\%1} v_{i,j})/(\ddot{A}x_i \% \ddot{A}x_{i\%1}) \end{aligned} \tag{17}$$

$$FCOR = \frac{VWL\, u_{i-\frac{1}{2},j} + VWR\, u_{i+\frac{1}{2},j}}{2x_i} \qquad \textbf{(18)}$$

Equation 13 utilizes the same upwind differencing used for the radial and axial momentum equations. In Hirt and Nichols,[4] this differencing algorithm is shown to be first-order accurate on a non-uniform mesh.

The centrifugal “force” term in the radial momentum equation was differenced as shown by Torrey et al.[6]

$$\frac{v^2}{r} = \frac{VWL^2 + VWR^2}{x_{i+\frac{1}{2}}} \qquad \textbf{(19)}$$

The viscous term is given by

$$VISTH = \tilde{o}\left[\frac{\partial}{\partial r}\frac{\partial(rv)}{r\partial r} + \frac{\partial}{\partial z}\frac{\partial w}{\partial z}\right] \qquad \textbf{(20)}$$

This expression is evaluated using central differences in direct analogy with the viscous terms in the radial and axial momentum equations.

APPENDIX B. DEFINITION OF INPUT VARIABLES

input to control restart and problem duration (namelist /abegin/)

nresta	restart control (0 = a new case from initial conditions, 1 = restart from a previous result)
incycl	the number of time steps to be completed for the current restart run
deltim	amount of simulated time to be added to the simulation on this run - the run will be terminated by cycles (time steps) or simulated time, whichever occurs first
inquir	frequency (in time steps) at which file 'startstop' will be interrogated to stop the run
ifstf	free surface or two fluid simulation (= 1) or single fluid, no free surface (= 0)
idefm	if idefm =1 turns on special treatment to suppress spurious voids as described on p. 20 of the NASAVOF-2D report, LA-10612-MS (see reference list)
adefm	parameters for void surpression
bdefm	
nthresh	number of iterations before time step is cut in pressure equation - default = 25
nocolim	number of times non-convergence of the pressure equation is allowed
fsaft	actual time step / maximum stable time step - "safety factor" on time step
irot	rotating flow (= 1, not implemented), no angular momentum solution (= 0)
dtmult	reserved for accelerated steady-state solutions - multiplies stable time step. Not implemented
advtim	implicit/explicit fraction for tilde step - convention = 0.0
mshake	options for earthquake simulation - not documented
gnorm	
nintrp	

input parameters (namelist /xput/)

alpha	controls amount of donor cell fluxing (=1.0 for full donor cell differencing, =0.0 for central differencing)
autot	automatic time step flag (=1.0 for automatic delt adjustment, =0.0 for constant delt)
cangle	contact angle, in degrees, between fluid and wall
csq	material sound speed squared (= -1.0 for incompressible material)
delt	time step
epsi	pressure iteration convergence criterion
flht	fluid height, in y-direction
gx	body acceleration in positive x-direction
gy	body acceleration in positive y-direction
icyl	mesh geometry indicator (=1 for cylindrical coordinates, = 0 for plane coordinates)
isurf10	surface tension indicator (=1 for surface tension, = 0 for no surface tension)
isymplt	symmetry plot indicator (=1 for symmetry plot, = 0 for no symmetry plot)
nmat	number of materials
npx	number of particles in x-direction in rectangular setup
npy	number of particles in y-direction in rectangular setup
nu	coefficient of kinematic viscosity
omg	over-relaxation factor used in pressure iteration
pltdt	time increment between file update for plots
prtdt	time increment between file update for printing of field variables
rhof	fluid density (for f=1.0 region)
rhofc	fluid density in complement of f region

sigma surface tension coefficient
twfin problem time to end calculation
ui x-direction velocity used for initializing mesh
wi y-direction velocity used for initializing mesh

nwb, nwl, nwr, & nwt are the indicator arrays for boundary condition to be used along the edges (b = bottom, l = left, r = right, t = top) of the mesh
=1 for rigid free-slip wall,
=2 for rigid no-slip wall,
=3 for continuitive boundary,
=4 for periodic boundary,
=5 for constant pressure boundary

A 'periodic boundary' must have value of "4" for each active element in the boundary arrays on both sides.

uln normal velocity on the left boundary, different value is allowed for each cell
(ie. uln(j), j = 1, jm1)
urn normal velocity on the right boundary
utt tangential velocity on the top boundary
ubt tangential velocity on the bottom boundary
wtn normal velocity on top boundary
wbn normal velcoity on bottom boundary
wlt tangential velocity on left boundary
wrt tangential velocity on right boundary (for rotating flows), only some of the options make sense:
if boundary = no flow, then convection flux is zero;
if boundary = free slip, then shear is zero;
if boundary = no slip, then shear is applied, wall velocity is specified;
if boundary = pressure, treat as continuitive;
if boundary = continuitive, then [r × velocity] = constant across the boundary;.
vlt tangential rotational velocity on left
vrt tangential rotational velocity on right
vbt tangential rotational velocity on bottom
vtt tangential rotational velocity on top
xpl location of left side of rectangular particle region
xpr location of right side of rectangular particle region
ypb location of bottom of rectangular particle region
ypt location of top of rectangular particle region
pervel peripheral velocity for rotating flows, specified at the peripheral radius
perrad the peripheral radius
omega the angular velocity associated with pervel & perrad

input to set up computational mesh (namelist /mshset/)

Note: this mesh generator input is the same as that described in the orignal SOLA-VOF documentation.[2] The description is not repeated here.

dxmn(n) minimum space increment in x-direction in submesh n
dymn(n) minimum space increment in y-direction in submesh n
nkx number of submesh regions in x-direction
nxl(n) number of cells between locations xl(n) and xc(n) in submesh n
nxr(n) number of cells between locations xc(n) and xl(n+1) in submesh n
nyl(n) number of cells between locations yl(n) and yc(n) in submesh n
nyr(n) number of cells between locations yc(n) and yl(n+1) in submesh n
xc(n) x-coordinate of the convergence point in submesh n
xl(n) location of the left edge of submesh n (nkx+1 values of xl(n) are necessary because the right edge (xr) of submesh n is determined by the left edge of submesh n+1)
yc(n) y-coordinate of the convergence point in submesh n
yl(n) location of the bottom of submesh n (nky+1 values of yl(n) are necessary because the top edge (yr) of submesh n is determined by the bottom edge of submesh n+1)

input to specify initial condition blocks and obstacle locations (namelist /obsics/)

mincs number of patches for initial u, w, & p fields
moblk number of patches for obstacles
iicll i (value) for (Initial Condition), lower left corner
jicll j (value) ...
iicur i (value) for (Initial Condition), upper right corner
jicur j (value) ...
iobll same for obstacle
jobll same for obstacle
iobur same for obstacle
jobur same for obstacle
varic(m) starting value for the mth patch
icvar(m) identifier for the mth variable
1 = u, 2 = w, 3 = v (azimuthal velocity), 4 = p, 5 = f
note that this routine can be used to set pressure boundary conditions by selecting a block that sits on the portion of the boundary of interest. Recommended rather than hard-coding special bc's.
iprll i index for lower left corner of region to be printed
iprur i index for upper right corner of region to be printed
jprll j index for lower left corner of region to be printed
jprur j index for upper right corner of region to be printed
i and j indices for corners of rectangular region printed for processing by Tecplot - defaults are 1 to im1 and 1 to jm1

input to produce history files (namelist /history/)

ihist(nt,npt) x (or r) index of point #npt to be monitored
jhist(nt,npt) y (or z) index of point #npt to be monitored
mhist number of locations to be specified
nhist number of time points to be recorded for each location

INTERNAL DISTRIBUTION

1. K. W. Childs
2. G. E. Giles
3. D. M. Hetrick
4. J. R. Kirkpatrick
5. M. A. Kuliasha
6. J. E. Park
7. M. W. Wendel
8. P. T. Williams
9. M. W. Yambert
10. Central Research Library

EXTERNAL DISTRIBUTION

11. Prof. A. J. Baker, Mechanical and Aerospace Engineering and Engineering Science, 316A Perkins Hall, Knoxville, TN 37996

12. Mr. T. D. Butler, Fluid Dynamics Group (T-3), MS-B216, Los Alamos National Laboratory, Los Alamos, NM 87544

13. Dr. L. D. Cloutman, P.O. Box 808, Lawrence Livermore National Laboratory, Livermore, CA 94550