JOURNAL OF COMPUTATIONAL PHYSICS **110**, 171–186 (1994)

GENSMAC: A Computational Marker and Cell Method for Free Surface Flows in General Domains

MURILO F. TOME AND SEAN MCKEE

University of Strathclyde, Department of Mathematics, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, United Kingdom

Received November 13, 1991; revised March 22, 1993

A computer program for solving two-dimensional incompressible viscous fluid flow in general domains is described. This is based on the simplified Marker and cell technique, but it has a number of novel features. A user-supplied data file of coordinates prescribes the fluid domain which can be quite general and needs only to be connected. With a view to parallelisation the momentum equations are solved explicitly, but an automatic step-changing routine optimises the stability restriction. A conjugate gradient solver is used to invert the discrete Poisson equation. An accurate approximation to the stress conditions on the free surface is adopted. The code is written in structured FORTRAN with features from FORTRAN 90. The efficacy of the code is illustrated by applying it to some industrial problems. © 1994 Academic Press, Inc.

1. INTRODUCTION

The marker-and-cell (MAC) method is a finite difference solution technique based on a staggered grid for investigating the dynamics of an incompressible viscous fluid. It was first introduced by Harlow and Welch [1]. It employs the primitive variables of pressure and velocity and has particular application to the modelling of fluid flows with free surfaces. One of the key features is the use of virtual particles whose coordinates are stored and which move from one cell to the next according to the latest computed velocity components. If a cell contains a particle it is considered to contain fluid, thus providing flow visualization of the free surface. Amsden and Harlow [2] subsequently developed a simplified MAC method (SMAC) which circumvented difficulties with the original method by splitting the calculational cycle into two parts, a provisional velocity field calculation followed by a velocity revision employing a potential function to ensure incompressibility throughout. Amsden and Harlow [2] describe a specific program, ZUNI, which embodies the SMAC methodology. ZUNI can be used to calculate two-dimensional fluid flow in rectangular or cylindrical coordinates. It can deal with free-surface flows with free-slip or no-slip conditions applying at rigid boundaries. Furthermore, provision is made for prescribed inflows and outflows, and a rectangular obstacle can be incorporated into the region. Fundamentally, though, the fluid domain is required to be rectangular and it is restricted to two dimensions.

There have been a number of developments of the SMAC code over the intervening years. Due to the limitation on the time step size for low Reynolds number Deville [3], Pracht [4], and Golafshani [5] have introduced implicit type schemes. In particular, Pracht [6] developed an implicit treatment of the velocity (and density) similar to the implicit–fluid Eulerian (ICE) method (see Harlow and Amsden [7]) using an arbitray Lagrangian–Eulerian (ALE) computational mesh. This in principle allows the calculation of flows involving curved or moving boundaries. Sicilian and Hirt [8] have also been active in developing the SMAC method. More recently, they have developed the containment atmosphere prediction (CAP) code using a particle in cell (PIC) approach to model the flow of a contaminant, in this case hydrogen. Miyata and Nishimura [9–11] have developed the MAC method for the simulation of water waves generated by ships of arbitrary configuration and breaking waves over circular and elliptical bodies. In both cases the inviscid boundary condition on the free surface was employed. A solution, based on the original MAC philosophy, for the free-slip condition imposed on an arbitrary domain was presented by Viecelli [12, 13]. This method is applicable to cases where all the boundaries are arbitrarily shaped and the mesh points do not coincide with boundary points. In his calculations, it was necessary to define and flag cells in the vicinity of the boundaries and to define the normal at each boundary cell. At cells adjacent to the boundary, he used weighted interpolation formulae to calculate momentum and pressure on the mesh. The method, named ABMAC by Viecelli, was applied to several specific problems and good results were reported. The extension of the technique using the SMAC philosophy is not clear, particularly, when no-slip conditions are imposed on boundaries of arbitrary shape. More recently, McQueen and Rutter [14] and Markham and Proctor [15] described modifications to the fluid flow code ZUNI to provide enhanced performance. Essentially they employed an automatic time-stepping routine and a preconditioned conjugate gradient solver for the Poisson equation in a rectangular region. Although the code we describe goes much further than those authors, it is fair to say that this work in its early stages was strongly influenced by them.

One feature of the SMAC (Amsden and Harlow [2]) method is that the calculational cycle is split into two parts: one for the velocity field and the other for the pressure field so that there is no iteration procedure involving velocity and pressure. We use this essential idea in the development of GENSMAC. In this report we describe modifications to the SMAC method in order to deal with free surface flows in general domains when the no-slip condition is imposed on curved boundaries. Sample calculations include the simulation of cavity filling which is a common means of moulding used in a range of industries including plastics, foods, ceramics, and metallurgy. Other more specific problems tackled with the same approach involve flow in the bottom of a circular cavity and the dual inlet problem in cavities of complex shape.

2. GENSMAC

The original motivation for considering the SMAC method was its potential applicability to a problem of simulating the high speed injection of material into a complex mould with the view to developing a useful design tool for engineers. Thus a method was required that was truely effective for general regions. In this paper we describe modifications to the SMAC method in order that it can be used to solve problems for quite arbitrary shaped domains. This method, which we call GENSMAC, is at present only two-dimensional; our intention is to develop it for three dimensions and optimise it for running on computers with large scale parallel architectures. The method, as currently implemented, has the following characteristics:

(i) The code was written from scratch in structured FORTRAN with features from FORTRAN 90.

(ii) It has the capacity for solving free surface fluid flow in arbitrary shaped domains. The user needs only supply a data file of coordinates and select interpolating functions most appropriate to each part of the curve.

(iii) It has an automatic time step changing routine (the idea is taken directly from Markham and Proctor [15]) which allows the stability restriction to be optimised.

(iv) It uses a conjugate gradient solver to solve a Poisson equation for the corrected velocity field. Our experience with the successive overrelaxation method showed it not to be efficient. Further, our experience with Markham and Proctor's code [15] showed it not be robust. However, the particular discretisation described in this paper ensures a SPD system for the discrete Poisson equation and the direct use of CG thus provides a robust solver.

(v) It contains an accurate approximation to the stress conditions on the free surface.

(vi) It can simulate flows involving many inflow and outflow boundaries.

The code, at present, only treats no-slip boundary conditions on curved boundaries.

3. THE UNDERLYING EQUATIONS AND THE SMAC METHODOLOGY

The basic equations for two dimensional time-dependent incompressible viscous flows are the Navier–Stokes equations, together with the continuity equation which in non-dimensional form become

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}= -\frac{\partial \phi}{\partial x}+\frac{1}{\mathrm{Re}}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right) + (1/\mathrm{Fr}^2)\, g_x \tag{1}$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}= -\frac{\partial \phi}{\partial y}-\frac{1}{\mathrm{Re}}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right) + (1/\mathrm{Fr}^2)\, g_y \tag{2}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{3}$$

where $\mathrm{Re}=UL/\nu$ and $\mathrm{F_r}=U/\sqrt{Lg}$ are the associated Reynolds number and Froude number, respectively. U and L are typical velocity and length scales, g is the gravitational constant with $\mathbf{g}=(g_x, g_y)^{\mathrm{T}}$ as the unit gravitational vector and $\mathbf{u}=(u, v)^{\mathrm{T}}$ are the non-dimensional components of velocity, while ϕ is the non-dimensional pressure per unit density.

The basic idea of the SMAC method (and also GENSMAC) is to solve the Navier–Stokes using a staggered grid and to "push" forward virtual particles by solving $d\mathbf{x}/dt=\mathbf{u}$ ($\mathbf{x}=(x, y)^{\mathrm{T}}$ and $\mathbf{u}=(u, v)^{\mathrm{T}}$) for the newly computed value of the velocity.

More precisely, the procedure can be stated as follows. We suppose that at a given time, say t_0, the velocity field $\mathbf{u}(\mathbf{x}, t_0)$ is known and boundary conditions for the velocity and the pressure are given. The updated velocity field $\mathbf{u}(\mathbf{x}, t)$, where $t=t_0+\delta t$, is then calculated as follows:

(i) Let $\tilde{\phi}(\mathbf{x}, t_0)$ be an arbitrary pressure field which satisfies the correct pressure condition on the free surface.

(ii) Calculate the intermediate velocity field, $\tilde{\mathbf{u}}(\mathbf{x}, t)$, from

$$\frac{\partial \tilde{u}}{\partial t}=\left[-\frac{\partial u^2}{\partial x}-\frac{\partial uv}{\partial y}-\frac{\partial \tilde{\phi}}{\partial x}+\frac{1}{\mathrm{Re}}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right)+(1/\mathrm{Fr}^2)\, g_x\right]_{t=t_0} \tag{4}$$

$$\frac{\partial \tilde{v}}{\partial t}=\left[-\frac{\partial uv}{\partial x}-\frac{\partial v^2}{\partial y}-\frac{\partial \tilde{\phi}}{\partial y}-\frac{1}{\mathrm{Re}}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right)+(1/\mathrm{Fr}^2)\, g_y\right]_{t=t_0} \tag{5}$$

with $\tilde{\mathbf{u}}(\mathbf{x}, t_0)=\mathbf{u}(\mathbf{x}, t_0)$, using the correct boundary conditions for $\mathbf{u}(\mathbf{x}, t_0)$. We now define the true velocity field

$$\mathbf{u}(\mathbf{x}, t)=\tilde{\mathbf{u}}(\mathbf{x}, t)-\nabla\psi$$

with

$$\nabla^2\psi=\nabla\cdot\tilde{\mathbf{u}}(\mathbf{x}, t).$$

The point here is that $\mathbf{u}(\mathbf{x}, t)$ now satisfies

$$\nabla\cdot\mathbf{u}(\mathbf{x}, t)=0.$$

(iii) Solve the Poisson equation

$$\nabla^2\psi=\nabla\cdot\tilde{\mathbf{u}}(\mathbf{x}, t). \tag{6}$$

(iv) Compute the velocity field

$$\mathbf{u}(\mathbf{x}, t)=\tilde{\mathbf{u}}(\mathbf{x}, t)-\nabla\psi. \tag{7}$$

(v) Compute the pressure.

Once we have computed $\mathbf{u}(\mathbf{x}, t)$ the pressure follows from (see MacQueen [32])

$$\phi=\tilde{\phi}+\psi/\delta t.$$

Thus GENSMAC solves the momentum equations explicitly and a sparse symmetric system (the discrete Poisson equation) for the potential function ψ. In the case of cavity filling the order of this system is continually increasing (since one only solves for $\mathbf{u}$ and ϕ within the bulk fluid).

Particles are created at the inlets and are injected into the containment region to represent the fluid. These are virtual particles whose coordinates are stored at each time step and then updated by solving

$$\frac{dx}{dt}=u, \qquad \frac{dy}{dt}=v \tag{8}$$

by Euler's method. This provides the new coordinates of the particle, determining whether a particle moves into a new cell or leaves the containment region through an outlet.

3.1. *Boundary Conditions*

There are several types of boundary conditions which can be applied at the mesh boundary, namely: no-slip, free-slip, prescribed inflow, prescribed outflow, and continuative outflow. Except for the no-slip condition, GENSMAC treats these boundary conditions in a similar manner to SMAC. For a detailed discussion see Tome and McKee [16]. On the mesh boundary the appropriate boundary condition for ψ (Harlow and Amsden [2]) is

$$\frac{\partial\psi}{\partial n}=0.$$

3.2. *The Basic Finite Difference Equations*

A staggered grid is employed. A typical cell is as shown in Fig. 1. The variables pressure $\tilde{\phi}_{i,j}$, the added velocity potential $\psi_{i,j}$ and the divergence $D_{i,j}$ are positioned at the cell centre while $u_{i,j}$ and $v_{i,j}$ are staggered by a translation of $\delta x/2$ and $\delta y/2$, respectively.

The momentum equations (4) and (5) are discretised and applied at the u-nodes and the v-nodes, respectively. In finite difference approximation these equations become (Amsden and Harlow [2]):

$$\begin{aligned}
&\frac{\tilde{u}^{n+1}_{i+1/2,j}-u^n_{i+1/2,j}}{\delta t}\\
&\quad=\frac{\tilde{\phi}_{i,j}-\tilde{\phi}_{i+1,j}}{\delta x}\\
&\qquad+\frac{u^n_{i+1/2,j-1/2}v^n_{i+1/2,j-1/2}-u^n_{i+1/2,j+1/2}v^n_{i+1/2,j+1/2}}{\delta y}\\
&\qquad+\frac{u^n_{i+1/2,j}u^n_{i-1/2,j}-u^n_{i+3/2,j}u^n_{i-1/2,j}}{\delta x}\\
&\qquad+\left(\frac{1}{\mathrm{Re}}\right)\left[\frac{(u^n_{i+1/2,j+1}+u^n_{i+1/2,j-1}-2u^n_{i+1/2,j})}{\delta y^2}\right.\\
&\qquad\left.-\frac{(v^n_{i+1,j+1/2}-v^n_{i+1,j-1/2}-v^n_{i,j+1/2}+v^n_{i,j-1/2})}{\delta x\,\delta y}\right]\\
&\qquad+(1/\mathrm{Fr}^2)\,g_x
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\frac{\tilde{v}^{n+1}_{i,j+1/2}-v^n_{i,j+1/2}}{\delta t}\\
&\quad=\frac{\tilde{\phi}_{i,j}-\tilde{\phi}_{i,j+1}}{\delta y}\\
&\qquad+\frac{u^n_{i-1/2,j+1/2}v^n_{i-1/2,j+1/2}-u^n_{i+1/2,j+1/2}v^n_{i+1/2,j+1/2}}{\delta x}\\
&\qquad+\frac{v^n_{i,j+1/2}v^n_{i,j-1/2}-v^n_{i,j+3/2}v^n_{i,j+1/2}}{\delta y}\\
&\qquad-\left(\frac{1}{\mathrm{Re}}\right)\left[\frac{(u^n_{i+1/2,j+1}-u^n_{i+1/2,j}-u^n_{i-1/2,j+1}+u^n_{i-1/2,j})}{\delta x\,\delta y}\right.\\
&\qquad\left.+\frac{(2v^n_{i,j+1/2}-v^n_{i+1,j+1/2}-v^n_{i-1,j+1/2})}{\delta x^2}\right]\\
&\qquad+(1/\mathrm{Fr}^2)\,g_y.
\end{aligned} \tag{10}$$

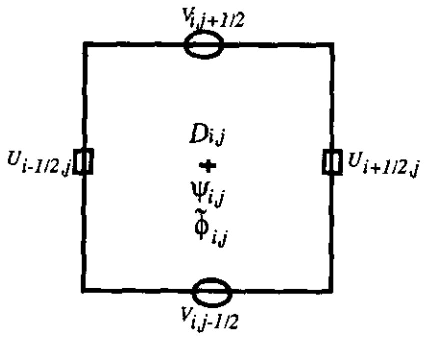

FIG. 1. Typical cell.

The Poisson equation (6) is discretised at cell centres using the five-point Laplacian which can be written as

$$4\psi_{i,j}-\psi_{i+1,j}-\psi_{i-1,j}-\psi_{i,j+1}-\psi_{i,j-1}=-h^2\tilde{D}_{i,j}, \qquad (11)$$

where

$$\tilde{D}_{i,j}=\frac{\tilde{u}_{i+1/2,j}-\tilde{u}_{i-1/2,j}}{\delta x}+\frac{\tilde{v}_{i,j+1/2}-\tilde{v}_{i,j-1/2}}{\delta y}$$

and

$$h := \text{grid size (assuming } \delta x=\delta y).$$

3.3. *Cell Flagging*

Cells within the mesh are flagged according to whether they are

1. Boundary cells (B). Cells lying on or outside the boundary domain. They play a static role by prescribing the position and curvature of the fixed boundary.

2. Empty cells (E). Cells containing no fluid.

3. Full cells (F). Cells containing fluid with no adjacent empty cell on any of their faces.

4. Surface cells (S). Cells containing the free surface. These cells have at least one adjacent empty cell on one of their faces.

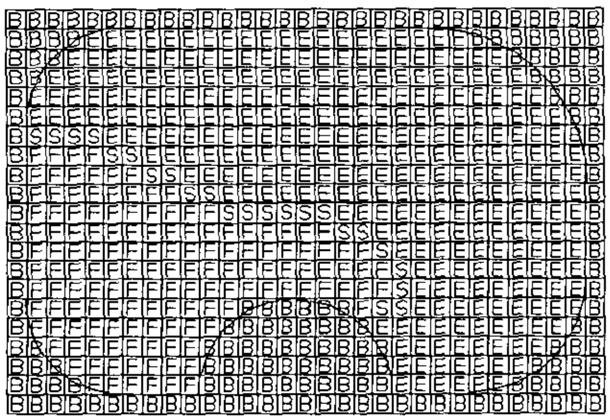

FIG. 2. Types of cells in a GENSMAC mesh.

These definitions are illustrated in Fig. 2 for a typical case. There can be some ambiguity about how one chooses whether a cell is a boundary cell as this depends upon where the boundary cuts the particular cell. This question will be treated in detail in the next section.

3.4. *Free Surface Stress Conditions*

The appropriate free-surface boundary conditions are the vanishing of the normal and tangential components of stress which can be represented by (Hirt and Shannon [17])

$$\phi-(2/\text{Re})\left[n_x n_x\frac{\partial u}{\partial x}+n_x n_y\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)+n_y n_y\frac{\partial v}{\partial y}\right]=0 \qquad (12)$$

$$\left[2n_x m_x\frac{\partial u}{\partial x}+(n_x m_y+n_y m_x)\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)+2n_y m_y\frac{\partial v}{\partial y}\right]=0, \qquad (13)$$

respectively, where $\mathbf{n}=(n_x, n_y)$ is the outward unit normal vector to the surface and $\mathbf{m}=(m_x, m_y)$ is the tangential vector.

It was pointed out (Hirt and Shannon [17]) that when these conditions are not accurately satisfied one can obtain results which are not correct, especially if the problem at hand involves a low Reynolds number. Subsequently, Nichols and Hirt [18] presented an improved treatment for the free surface stress conditions in which the normal stress (12) is applied at the actual fluid surface rather than at the centre of surface cells. They defined a special ordered set of marker particles describing the free surface and then connected these particles by line segments in order to specify the surface shape. The pressure at the surface cell centre was then obtained from a linear interpolation between the pressure at the surface and the pressure at an adjacent full cell centre. They reported that with this scheme the free surface conditions were accurately satisfied. The tangential stress was approximated in the same way as the SMAC method.

We have not yet implemented this technique in GENSMAC since our intention has always been to solve problems involving an arbitrary number of free surfaces and an arbitrary number of inflows/outflows and the resulting logical process of deleting/creating surface particles would prove to be a formidable task. However, the implementation of the normal stress condition has been performed in accordance with Hirt and Shannon [17] as follows:

Let us suppose that the grid size is small enough so that the surface will cut the cell at two edges. Then (12) and (13) can be approximated by local finite differences according to three cases:

(a) Surface cells with only one side contiguous to an empty cell. For these cells we assume that the surface will cut two opposite cell edges so that either n_x or n_y will be small. Then, (12) and (13) can be approximated by

$$\phi - \frac{2}{\text{Re}}\left(\frac{\partial u_n}{\partial n}\right) = 0 \tag{14}$$

and

$$\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0, \tag{15}$$

respectively, where n is either the x-direction or the y-direction.

For instance, considering the surface cell in Fig. 3, the corresponding finite difference approximation for the normal stress (14) is

$$\phi_{i,j} = \frac{2}{\text{Re}}\left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\delta x}\right)$$

while the tangential stress (15) is satisfied by relating the velocity outside the surface to those inside the fluid by

$$v_{i+1,j+1/2} = v_{i,j+1/2} - \frac{\delta x}{\delta y}\,(u_{i+1/2,j+1} - u_{i+1/2,j}).$$

Other types of surface cells with one side contiguous to an empty cell are treated similarly.

(b) Surface cells with two adjacent sides contiguous with empty cells. For these cells we assume that the outward normal direction lies at 45° between the open sides, in which case (12) and (13) reduce to

$$\phi = \pm\frac{1}{\text{Re}}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{16}$$

and

$$\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} = 0, \tag{17}$$

respectively. The sign in (16) is chosen to be the sign of $n_x n_y$.

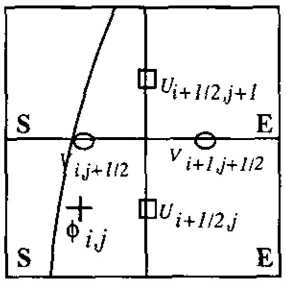

FIG. 3. Surface cell with right side open to an E-cell.

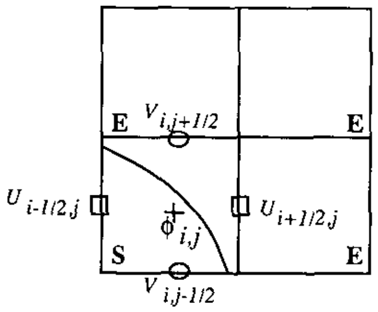

FIG. 4. Surface cell with two adjacent sides contiguous with E-cells.

For example, if we consider the surface cell as shown in Fig. 4 the finite difference approximation to the normal stress (16) would be

$$\phi_{i,j} = \frac{1}{2\,\text{Re}}\left[\frac{u_{i+1/2,j} + u_{i-1/2,j} - u_{i+1/2,j-1} - u_{i-1/2,j-1}}{\delta y} + \frac{v_{i,j+1/2} + v_{i,j-1/2} - v_{i-1,j+1/2} - v_{i-1,j-1/2}}{\delta x}\right].$$

For the tangential stress (17) we require that $\partial u/\partial x$ and $\partial v/\partial y$ vanish separately. The reason for doing this is that the mass conservation equation is also satisfied for these cells. For instance, for surface cells having the top and right sides contiguous with empty cells (see Fig. 4) we set

$$u_{i+1/2,j} = u_{i-1/2,j} \qquad v_{i,j+1/2} = v_{i,j-1/2}.$$

For other configurations of surface cells with two adjacent sides contiguous with empty cells the finite difference approximations are obtained similarly.

(c) Surface cells with three sides contiguous (or two sides which are opposite to one another) with empty cells. For these cells, a more accurate knowledge of the position and shape is required than is presently determinable with the SMAC method. In such cells the pressure is set to zero and one velocity is adjusted so that $\nabla \cdot \mathbf{u} = 0$ in the surface cell. If such cells arise it suggests that the grid should be refined until they do not exist or that the high local curvature be smoothed out.

One should note that the approximations described here coincide with the ones given by Hirt and Shannon [17]. Further, with these approximations and provided the grid is sufficiently fine, GENSMAC can solve problems involving high curvature.

3.5. *Particle Movement*

As in SMAC, marker particles are used to represent the fluid. Their essential task is to provide the position of the moving free surface so that the configuration of the surface cells can be determined. They are updated at the end of each calculational time step so as to provide the dynamics of the fluid motion. The new particle coordinates are found by solving (8) using Euler's method. Thus, after the velocity field is updated, the particles are moved according to

$$x_p^{n+1} = x_p + u_p\,\delta t^{n+1}$$
$$y_p^{n+1} = y_p + v_p\,\delta t^{n+1},$$

where (x_p, y_p) is the current particle position, δt^{n+1} is the actual time step employed, and (x_p^{n+1}, y_p^{n+1}) is its updated position. The velocities u_p, v_p are computed as in [2] by using an area weighting scheme involving the four nearest u, v velocities, respectively. In all problems this criterion for particle movement has displayed good results, with particles on the curved boundaries following the curved boundaries.

4. CURVED BOUNDARY TREATMENT

The basic idea is to decompose the curved boundary into one which coincides with mesh segments and then to solve the SMAC equations taking account of this pseudo-boundary as follows.

4.1. *Virtual Boundary Definition*

For a rectangular domain, no problems arise when the finite difference equations (9) and (10) are applied at nodes near the boundary wall because these coincide with cell edges; but a curved boundary will not, in general, coincide with cell edges and any interpolation procedure must be in accord with the staggered grid. This can be achieved first by determining where the curved boundary cuts the cell edges and then flagging this cell as a B-cell or not. Let us assume that the cells are sufficiently small so that a curved boundary will cut a cell through two of its edges. Let us denote them by x_a and y_a, where x_a is the x-coordinate of the intercept point belonging to the y-line at the bottom or top edge of the cell while y_a is the y-coordinate of the intercept point lying on the x-line at the right or left side of the cell. Then, all the possible configurations of cells cut by the curved boundary can be classified into one of two groups: cells cut on two adjacents sides (corner cell) and cells cut on sides which are opposite to one another (edge cell). Figures 5 and 6 displays examples of such cells.

Among the corner cells and the edge cells we consider those as shown in Figs. 5a, 5b, 6a, and 6b to be interior cells

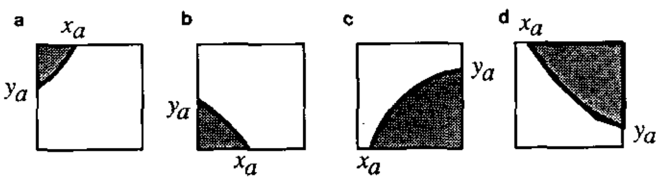

FIG. 5. Types of cells cut by a curved boundary: Corner cells.

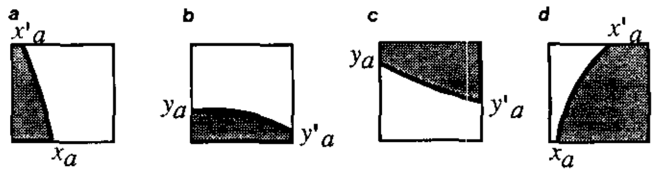

FIG. 6. Types of cells cut by a curved boundary: Edge cells.

(i.e., F, S, or E) because only a small part of these cells is occupied by the rigid boundary while the other cells (see Figs. 5c, 5d, 6c, 6d) are considered to belong to the curved boundary, and a special treatment is then required for these cases. Thus, the problem of flagging is considerably simplified: we first determine which cells are cut by the boundary and then which of these are interior ones. Let (x_c, y_c) denote the coordinates of the centre of any cell. Then we check to see if one of the following is true:

$x_a > x_c$ and $y_a < y_c$ {for left upper corner cells (see Fig. 5a)}

or

$x_a > x_c$ and $y_a > y_c$ {for left lower corner cells (see Fig. 5b)}

or

$x_a < x_c$ and $y_a > y_c$ {for right lower corner cells (see Fig. 5c)}

or

$x_a < x_c$ and $y_a < y_c$ {for right upper corner cells (see Fig. 5d)}

or

$x_a > x_c$ or $x'_a > x_c$ {for left vertical edge cells (see Fig. 6a)}

or

$y_a > y_c$ or $y'_a > y_c$ {for lower horizontal edge cells (see Fig. 6b)}

or

$y_a < y_c$ or $y'_a < y_c$ {for upper horizontal edge cells (see Fig. 6c)} or

$x_a < x_c$ or $x'_a < x_c$ {for right vertical edge cells (see Fig. 6d)}.

If one of these is true the cell is flagged as a boundary cell; otherwise it is a interior cell.

The conditions given above are designed to identify, from among those cells in the group of corner and edge cells which ones are to be considered as belonging to the curved boundary or not. The criterion is based on the fact that a cell can be divided into four quadrants (see Fig. 7) and then a test is made to identify which of these quadrants are intercepted by the curved boundary. If the curved boundary intercepts three or more quadrants of the cell we require the cell to belong to the curved boundary; otherwise it is considered to be a interior cell. For instance, the first condition for the group of corner cells is applied to cells which are cut by the curved boundary on the top and left edges. It checks to see if $x_a > x_c$ and $y_a < y_c$. If these hold then the curved boundary occupies at least some portion of quadrants II, IV, together with some fraction of quadrant I. In this case the cell is considered to belong to the curved boundary. On the other hand, if the test is not satisfied it means that the curved boundary occupies only some fraction of quadrants I and II or I and IV of the cell, in which case we consider this cell to be an interior cell. The other three conditions for testing the corner cells apply similar criteria.

The treatment for the edge cells is analogous. For example, the first condition applies to cells which are cut on the right and left edges (see Fig. 8) by a curved boundary situated on the bottom side of the cell. It checks if the curved boundary occupies any portion of the cell lying in the quadrants I and II. If so, the cell is considered to belong to the curved boundary and, if not, it is an interior cell. The other types of edge cells are treated similarly.

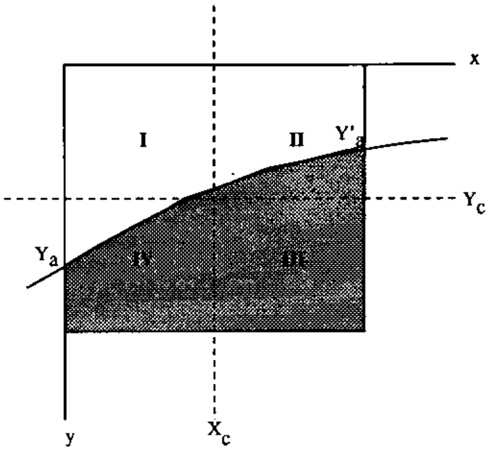

FIG. 8. Edge cell cut by a curved boundary from the bottom side.

Thus, after the cells have been flagged the boundary is approximated by piecewise continuous mesh lines. Figure 9 displays an example of such a boundary. The approximated boundary, which henceforth we shall call the "virtual boundary," will be the reference boundary for the pressure calculation. One feature of the virtual boundary is that when we are solving the Poisson equation for the corrected velocities, the Neumann condition is easily handled because it coincides with mesh lines.

4.2. *Boundary Condition on Curved Surfaces*

When the discretized Navier–Stokes equations are applied at nodes adjacent to the virtual boundary, the u and v values at boundary cell faces are required. If no-slip conditions are imposed on the curved boundary, these values can be estimated in terms of function values at internal nodes by linear (bilinear) interpolation.

It can be seen that the boundary cells may have one or two sides open to interior cells, as shown in Figs. 10 and 11.

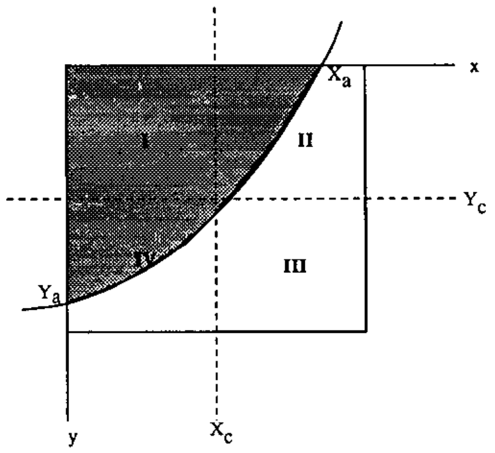

FIG. 7. Corner cell cut by a curved boundary on top and left cell edges.

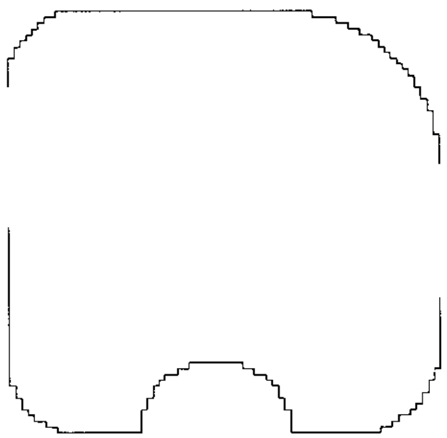
FIG. 9. Virtual boundary for a GENSMAC calculation.

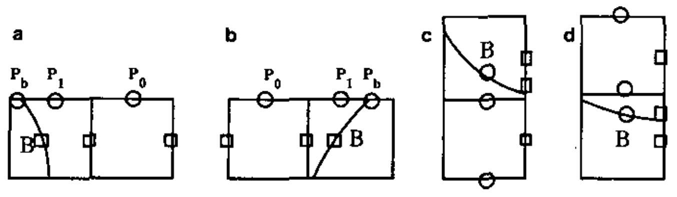

FIG. 10. Configurations of B-cells with only one side open to interior cell.

Thus, if a B-cell has one open side the u, v-values are computed using linear interpolation while for B-cells having two sides open, bilinear interpolation can be employed.

For instance, consider the B-cell as in Fig. 10a (or 10b) for the calculation of u and v. Let us denote the values of x, y, u, v at the mesh points $\mathbf{p_0}$ by x_0, y_0, u_0, v_0, with similar notations at mesh points $\mathbf{p_1}$ and $\mathbf{p_b}$. Then, it can be verified that the use of linear Lagrangian interpolation to compute u_1 gives

$$u_1 = \frac{(x_1 - x_b)}{x_0 - x_b} u_0 + \frac{(x_1 - x_0)}{x_b - x_0} u_b.$$

Noting that $u_b = 0$ (the no-slip condition states that $\mathbf{u} = \mathbf{0}$ on the boundary) leads to

$$u_1 = \frac{(x_1 - x_b)}{x_0 - x_b} u_0,$$

where u_0 is the internal value and x_0, x_1 are known multiples of the grid size h. The y-coordinates of $\mathbf{p_0}$ and $\mathbf{p_1}$ are both the same as the y-coordinate of $\mathbf{p_b}$ and x_b can thus be found from the equation describing the curved boundary

$$f(x_b, y) = 0.$$

The value of v_1 is computed in the same manner, namely

$$v_1 = \frac{(x_1 - x_b)}{x_0 - x_b} v_0.$$

For B-cells as in Figs. 10c and 10d it can be shown that the use of linear Lagrangian interpolation to compute u_1 and v_1 gives

$$u_1 = \frac{(y_1 - y_b)}{y_0 - y_b} u_0$$
$$v_1 = \frac{(y_1 - y_b)}{y_0 - y_b} v_0,$$

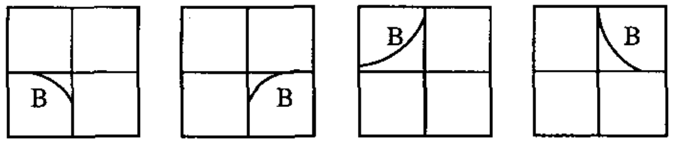

FIG. 11. Configurations of B-cells having two sides open to interior cells.

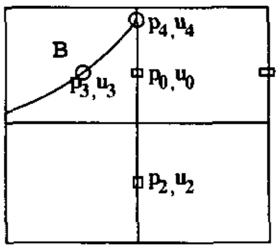

FIG. 12. Boundary cell with right and bottom sides open to interior cells.

respectively, where y_b is computed from

$$f(x, y_b) = 0.$$

In the case of B-cells having two open sides to interior cells, the u, v values are computed by using bilinear interpolation (Strang and Fix [19]). For example, related to Fig. 12, the value of u_0 is obtained by inverting an isoparametric transformation from the square $R := [0, 1] \times [0, 1]$ onto the quadrilateral defined by the points $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}$. It can be shown (Tome and McKee, [16]) that in this case u_0 can be approximated by

$$u_0 = (1 - s_0)(1 - t_0)\, u_1 + (1 - t_0)\, s_0 u_2,$$

where

$$t_0 = \frac{1 - \sqrt{(x_0 - x_3)(y_4 - y_0)/h^2}}{1 - (x_0 - x_3)(y_4 - y_0)/h^2}$$

$$s_0 = \frac{1 - t_0}{1 - t_0(1 - (x_0 - x_3)/h)}$$

and (x_i, y_i), are the coordinates of the points $\mathbf{p}_i$. The required v-value is calculated similarly.

For other configurations of B-cells with two open sides to interior cells, the u, v values are obtained in a similar manner.

4.3. *Boundary Conditions for the Poisson Equation*

For the Poisson equation (6), the rigid boundary condition

$$\frac{\partial \psi}{\partial n} = 0$$

is easily approximated by local finite differences because the "virtual boundary" coincides with mesh lines. For instance, for B-cells with only one side contiguous with a F-cell, this equation becomes either

$$\frac{\partial \psi}{\partial x} = 0 \quad \text{or} \quad \frac{\partial \psi}{\partial y} = 0,$$

depending on whether the common separating line is in the x- or in the y-direction. If a B-cell has two sides contiguous with F-cells, then the boundary condition is represented by

$$\frac{\partial \psi}{\partial x}=0 \qquad \frac{\partial \psi}{\partial y}=0.$$

As the function $\tilde{\phi}$ satisfies the correct pressure condition on the free-surface, the appropriate boundary condition for ψ on the free-surface is

$$\psi=0.$$

5. TIME-STEPPING PROCEDURE

Amsden and Harlow [2] suggested that the number of calculational cycles and hence the running time could be reduced by the use of an adaptive time-stepping routine which, at a given cycle, would automatically choose the time step most appropriate to the velocity field at that cycle. Welch *et al.* [20] discussed stability and accuracy requirements for the MAC method. They suggested that two stability restrictions are required. The first is akin to the Courant condition,

$$C\,\delta t<\frac{\delta x\,\delta y}{\delta x+\delta y},$$

where δx, δy are the mesh spacings, δt is the time step and C is the "wave speed,"

$$C=(gk\tanh k\,\delta)^{1/2},$$

k being the wave number and δ the depth of the fluid. It should be said that most of the examples in Welch *et al.* [20] involve water waves. The second stability restriction involves the viscosity:

$$2\nu\,\delta t<\frac{\delta x^2\,\delta y^2}{\delta x^2+\delta y^2}.$$

It is clear that the second criterion can be applied directly to select an appropriate time step, but that the first will only be appropriate for selected classes of problems. A more appropriate treatment used by Markham and Proctor [15], among others, is to require that no particles should cross more than one cell boundary in a given time interval, i.e., that

$$|u|\,\delta t<\delta x$$
$$|v|\,\delta t<\delta y.$$

Note that, although the fluid is modelled as incompressible, disturbances propagate through it at a speed governed by the explicit discretisation of the momentum equations.

We shall now discuss the implementation of this adaptive time-stepping procedure as described by Markham and Proctor [15]. Let us define

$$\delta t_{\text{visc}}=\frac{A_1}{2\nu}\cdot\frac{\delta x^2\,\delta y^2}{\delta x^2+\delta y^2} \tag{18}$$

$$\delta t_u=A_2\cdot\frac{\delta x}{2U_{\max}} \tag{19}$$

$$\delta t_v=A_2\cdot\frac{\delta y}{2V_{\max}}, \tag{20}$$

where $0\leqslant A_i\leqslant 1$. The extra factors of 0.5 have been introduced as a conservative measure. The time step to be used at a given point in the calculation will be

$$\delta t=\min(\delta t_{\text{visc}},\delta t_u,\delta t_v)\,A, \tag{21}$$

where $0<A\leqslant 1$; the reason for this extra factor will be made clear shortly.

We have not specified what we mean by $U_{\max}$ and $V_{\max}$. It could mean the maximum u (or v) velocity at the beginning of a cycle or at the end. Clearly, if $\max_{i,j}|U^{n+1}_{i,j}|$ were to be chosen instead of $\max_{i,j}|U^{n}_{i,j}|$ then this could lead to a substantial increase in computational effort. Markham and Proctor [15] instead make a compromise: they chose the tilde (provisional) velocities. Thus if the δt_u or δt_v is less than the time step δt adopted, then the time step is revised and the tilde velocities recalculated. The rôle of the added factor A in (21) is now seen as a means of compensating for the use of the tilde velocities rather than the $U^{n+1}_{i,j}$ $(V^{n+1}_{i,j})$ velocities. On test problems Markham and Proctor [15] found this adaptive technique led to considerable computational savings. Our experience concurs with them and this technique has been incorporated in GENSMAC.

6. THE POISSON SOLVER

At every calculational cycle we have to solve the Poisson equation on a general domain subject to Dirichlet and Neumann boundary conditions. This can be extremely time consuming for large problems, making the choice of an appropriate robust algorithm vital.

There have been several methods for solving sparse linear systems (e.g., Hageman and Young [21], Birkhoff and Linch [22]). Point iterative methods have often been used due to their inexpensive iterates and simple implementation. For linear systems arising from free surface flow problems, Botta and Ellenbroek [23] presented a modified SOR method which was implemented in the SOLA-VOF (Hirt and Nichols [24]) method and they reported good improvement for the solution of the Poisson equation. They pointed out that the standard SOR may have very slow convergence in this case as the iteration matrix has complex eigenvalues which jeopardise the convergence. This method is based on a knowledge of the exact position of the free surface so that its application to the SMAC method requires considerable additional work for a complete formulation. Ehrich [25] formulated an ad hoc SOR method for the Poisson equation in irregular domains. His method worked well compared to those obtained by Brazier [26] but lack of theoretical results makes it less attractive for tackling large problems. Takemitsu [27] presented a method which he claims is faster than the SMAC method. Essentially, he introduced an invariant into the Navier–Stokes equations and then solved the resulting Poisson equation through a SLOR [28] iterative procedure involving both the potential function and the final velocities. In his experiments, he compared the exact solution of a Poiseuille flow with the solution obtained by MAC, SMAC, streamfunction–vorticity, and his method. He reported that his method was faster than MAC and SMAC methods, although the convergence of the procedure was not guaranteed and different relaxation factors needed to be employed when solving both the SMAC equations and his own. Results for flows involving free surfaces were not presented.

Aiming to enhance the performance of the ZUNI code (Amsden and Harlow [2]), Markham and Proctor [15] described modifications to the Zuni code regarding its speedup and accuracy. In particular they replaced the SOR solver used in Zuni by a preconditioned conjugate gradient solver (PCG) for the Poisson equation. Markham and Proctor's [15] code, called COSMAC, has a switch mechanism, allowing a choice of Cartesian or cylindrical coordinates. They reported that for the most part their PCG package led to improvements, both in terms of computer time and accuracy. However, COSMAC was tested at Unilever Research Laboratories [29] for a jet intrusion problem and the results were shown to be extremely sensitive to parameter variation, often not working at all and always requiring multiple precision.

In our case the solution is sought in an arbitrary but preset region using Cartesian coordinates. Because this region is piece-wise continuous, consisting of lines either

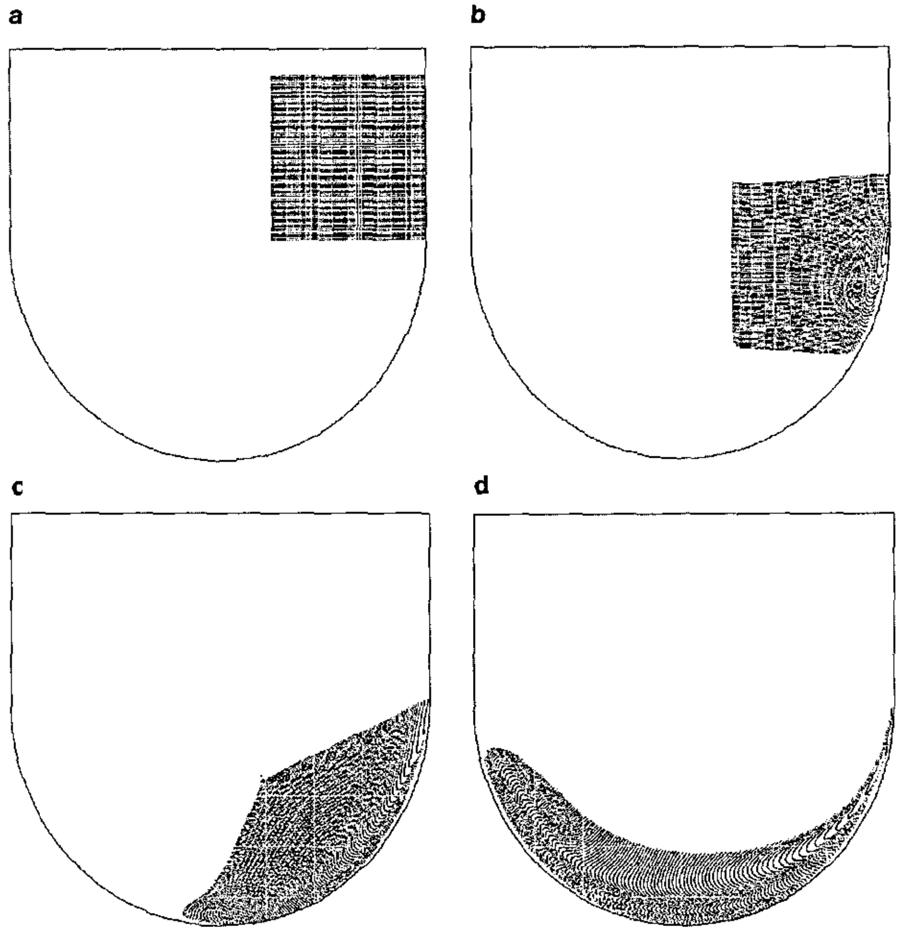

FIG. 13. Particle plot for the droplet problem at times $t = 0.0$, 2.0, 3.0, 6.0, 7.5, 9.0, 12.0, and 22.5, respectively.

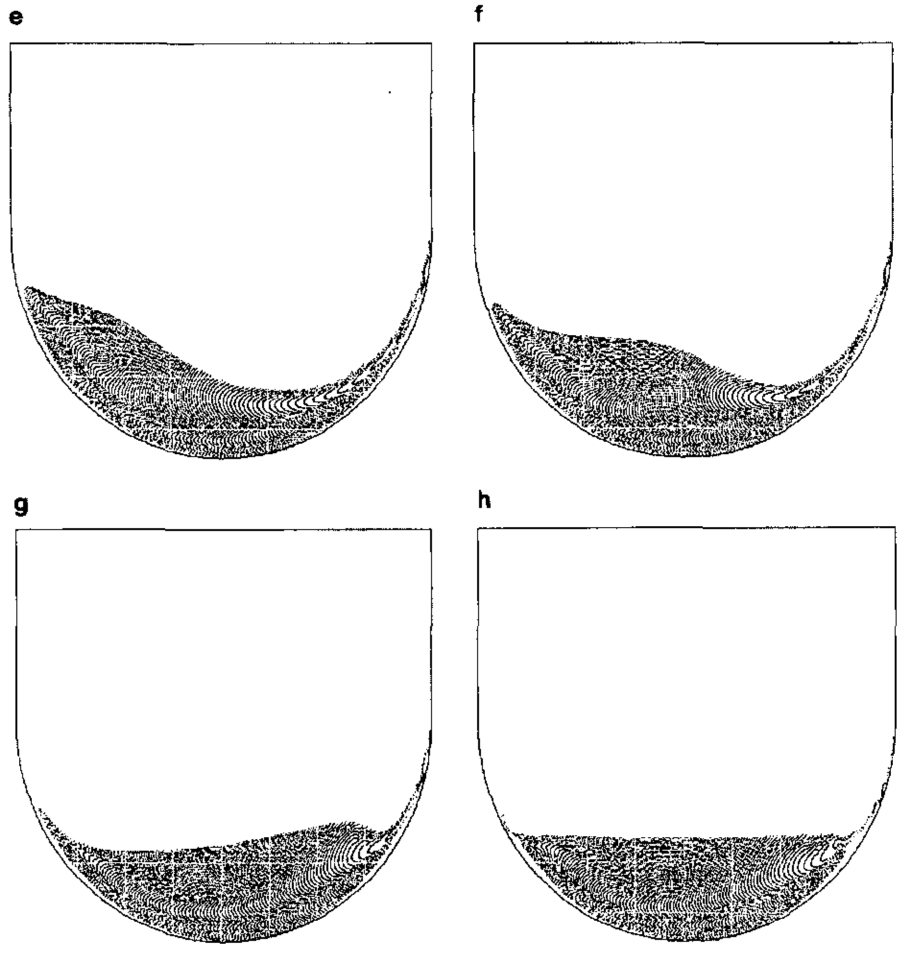

FIGURE 13—*Continued*

parallel to the x-axis or the y-axis, this leads to a system of equations for the discretised Poisson equation whose matrix is symmetric and positive definite. A conjugate gradient method (see Ortega [3]) has been implemented to solve this linear system resulting in a robust algorithm.

6.1. *Implementation of the Conjugate Gradient Method*

The discretised Poisson equation (11) leads to a linear system which can be represented by

$$AX = D,$$

where A is a sparse symmetric matrix of order n and X and D are column vectors of order n; n represents the number of full cells (F-cells) within the mesh, D is the vector of divergence $(\nabla \cdot \tilde{D})$ of each full cell, and X is the solution vector.

The matrix A is assembled by applying Eq. (11) at each full cell, row by row from right to the left. Each full cell represents a row of matrix coefficients in A and an element in D. The homogeneous Neumann condition enters via modifications to the diagonal term in rows corresponding to full cells with sides contiguous to the virtual boundary while the homogeneous Dirichlet condition on the free surface is applied by dropping the off-diagonal term in rows corresponding to full cells which have sides contiguous with surface cells. It can be shown that the presence of either a free surface or a continuative outflow boundary assures that this matrix is positive definite. After the matrix is assembled the vector X is initialised as the vector of the potential function in the previous time step and we apply the conjugate gradient algorithm as in Ortega [30] to solve this linear system.

At this point, it may be noted that a direct solver could have been employed, or that a preconditioner could have been added, which might well have speeded up the conjugate gradient method. However, direct solvers are well known not to be efficient for very large systems. On the other hand, since the order of the linear system is continually varying at each time step, any preconditioner or sequence of preconditioners would be expensive. Moreover, since our intention is to extend the method to three dimensions, in which case a vector/parallel machine would be required, it is well known that an efficient preconditioner is dependent on the particular architecture of the machine [31].

7. CALCULATIONAL EXAMPLES AND APPLICATIONS

To illustrate the capability of this newly extended version of the SMAC method we present some calculations performed by the GENSMAC code on free surface flows involving curved boundaries. These include a droplet into a cavity and jet intrusion into a container with curved walls. Although here we do not attempt any detailed comparison with experimental or analytical data, proof testing and comparison confidence in the validity of the calculations. A variety of comparisons were made with the problems treated by Amsden and Harlow [9] with results that were in good agreement.

In addition, to exhibit the applicability of this new technique in being able to handle highly complicated geometries, we used the code to simulate the filling of cavities with complex shapes used in the manufacture of novelty products.

A. *Droplet into a Circular Cavity*

This example shows the motion of a square block of fluid under the force of gravity as it falls into a circular cavity. No-slip conditions are applied at the curved wall while free-slip conditions were applied at the other walls. This example was considered by Viecelli [12] with the free-slip condition imposed on the curved wall. In this example we used a "diffusion coefficient" $(1/\mathrm{Re}) = 0.15$ and the gravitational force in the y-direction was $(1/\mathrm{F}_r^2) = -1.0$. The parameters for the time step routine were $A = 0.8$, $A_1 = A_2 = 0.5$ and a distribution of 16 particles per cell was initially employed. The whole calculation took 2274 cycles and the CG solver took an average of 25 iterations per cycle to satisfy a convergence criterion $\varepsilon = 10^{-5}$ for a linear system of 555 equations (on average) at each time step. The test for convergence was based on the L_∞ norm of the residual, and the final residual $\|D - AX\|_\infty$ was also required to satisfy

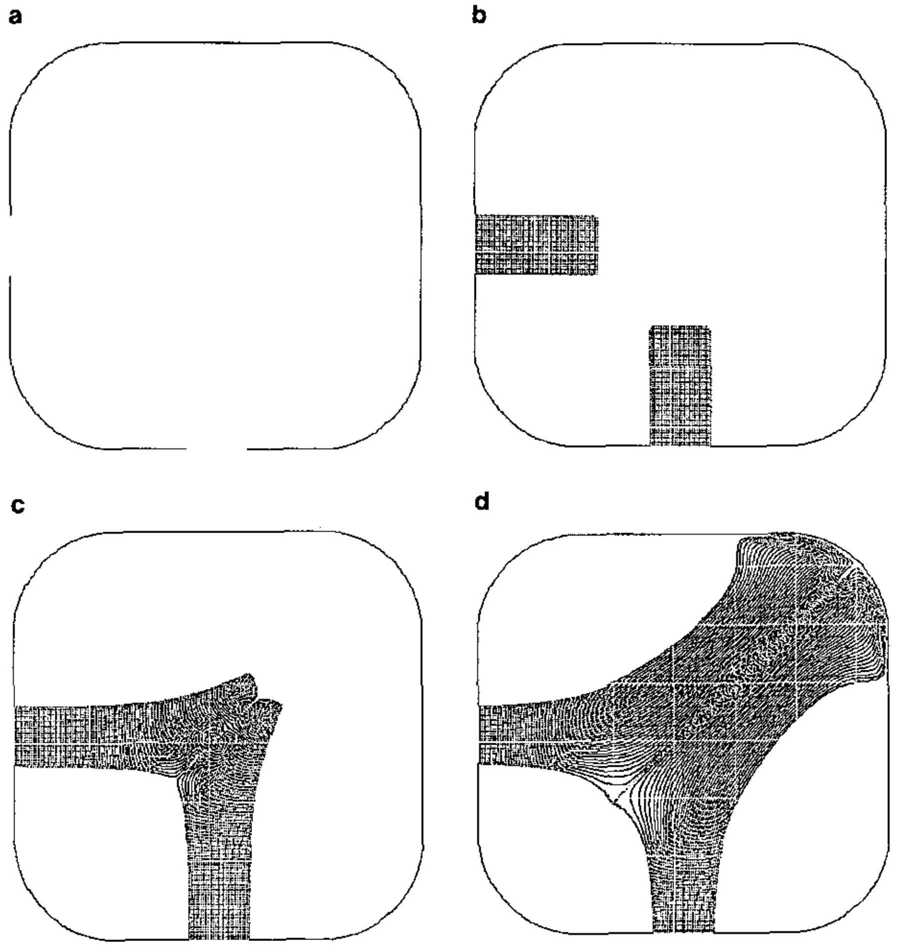

FIG. 14. Particle plot for the jet filling at times $t = 0.0$, 6.0, 12.0, 27.0, 56.0, 59.0, 62.0, and 62.68, respectively.

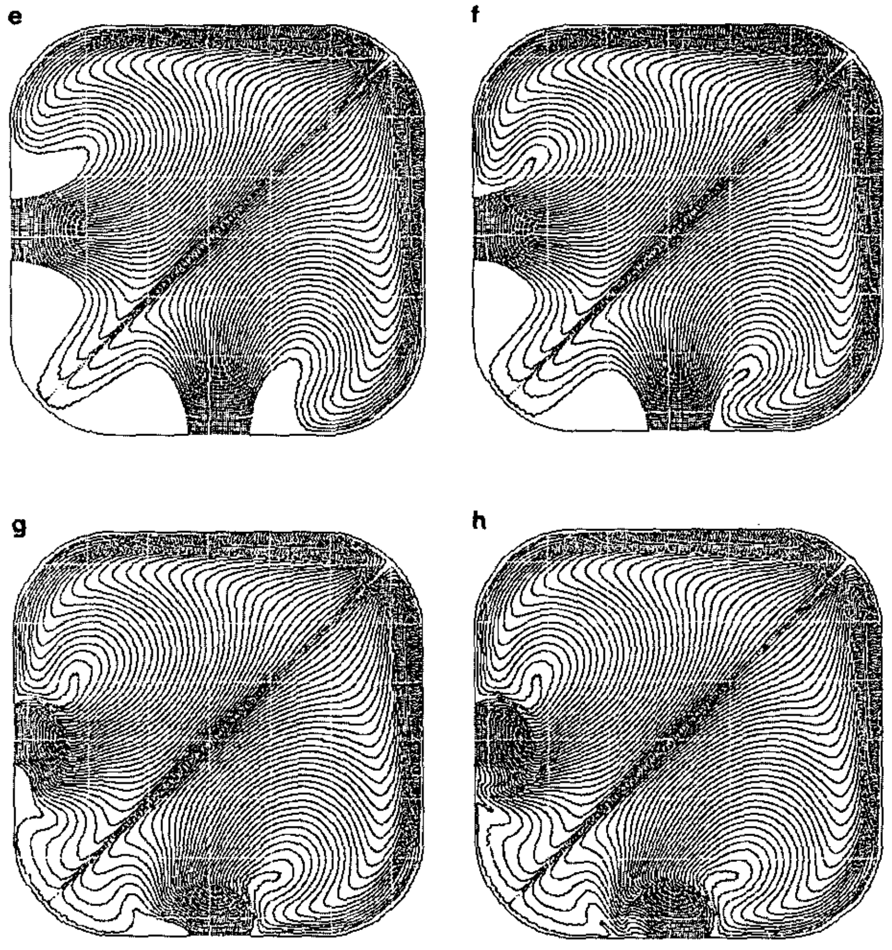

FIGURE 14—*Continued*

the convergence criterion. Figure 13 displays the particle configuration at different time intervals. Figure 13 shows the dynamics of the fluid motion under the force of gravity as it sloshes back and forth. The final picture shows the approaching stationary configuration of the free surface. As in all incompressible flow calculations, volume or mass conservation is necessary for accurate results. The CG solver as implemented ensures that $\tilde{D}_{i,j}$ remains negligible for each cell under a user-specified criterion and, as we can see from the plots, the total volume also remains constant throughout the run to within the accuracy that can be measured from the particle configuration plots. Therefore, we conclude that momentum and mass were rigorously conserved, validating the treatment of the boundary conditions on the cavity wall.

B. *Jet Filling of a Curved Container*

In this example we exhibit the feature of the GENSMAC code in being able to deal with several inlets. This problem was investigated (see [29]) with the COSMAC code [15] to simulate the filling of a square container with a highly viscous fluid. We employ a container with curved corners and use a "diffusion coefficient" of $(1/\text{Re}) = 2.0$ and switch off gravity. One inlet is set on the left wall and another inlet is set on the bottom wall. A mesh of 80×80 cells is employed and a convergence criterion $\varepsilon = 10^{-6}$ is used. The run started with an empty cavity and finished when the cavity was completely full. Particle plots are shown in Fig. 14.

C. *Simulation of a Moulding Process for Novelty Products*

To further illustrate the capabilities of GENSMAC two complex mould shapes were selected which are normally used to manufacture novelty products: The "foot" mould and the "finger" mould are both asymmetric and have a complex boundary in two dimensions.

The problem described here represents the high speed injection moulding of relatively large cavities through small inlet ports. The product material (e.g., molten plastic, liquid metal, or slurry) enters the cavity with high velocity and initially as a free jet impinging on the surface of the mould

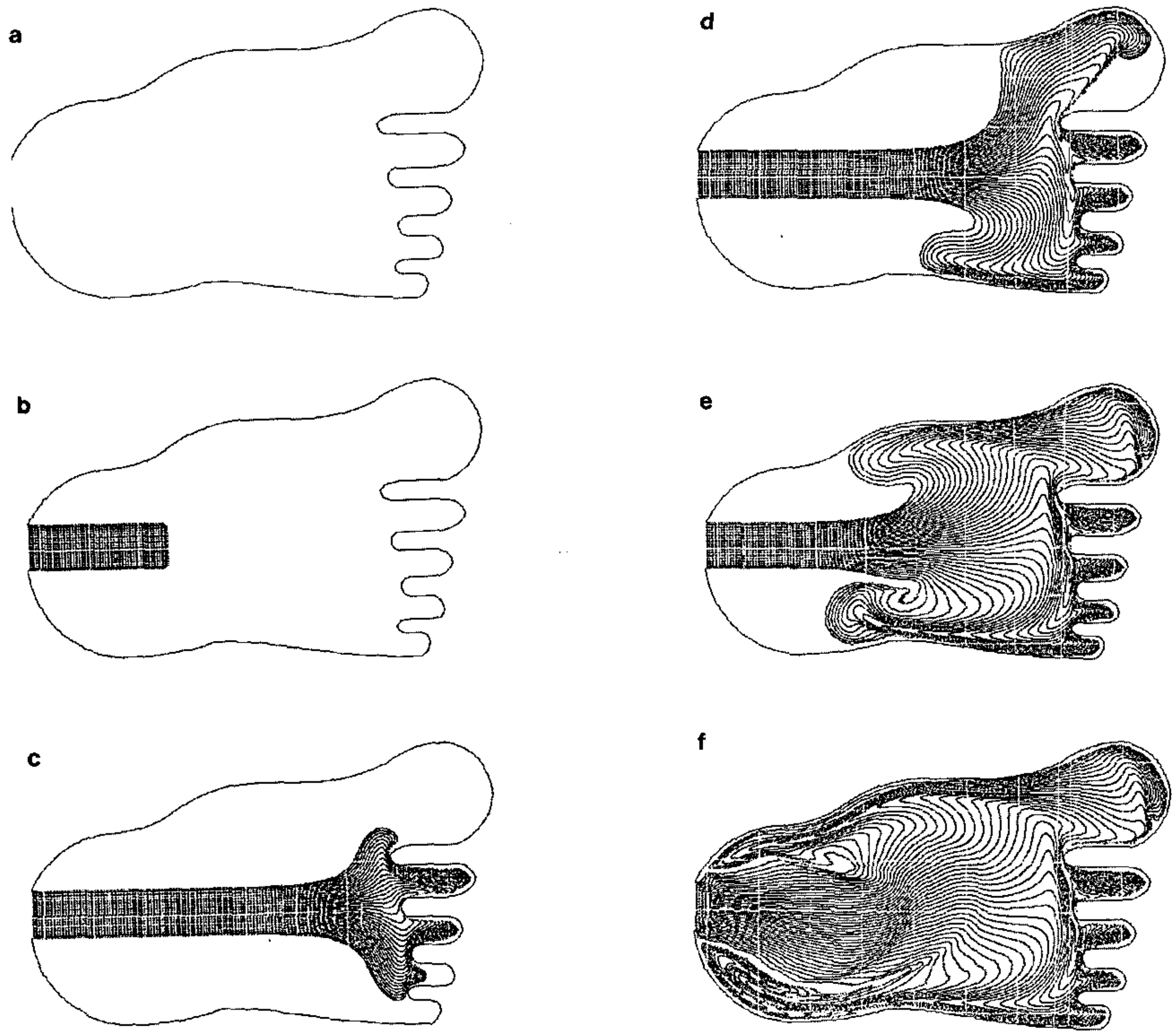

FIG. 15. The simulation of the filling process for the "foot" mould at times $t = 0.0$, 3.0, 12.0, 21.0, 31.40, and 38.72, respectively.

opposite the inlet port. The Reynolds number which is an important determinant of the filling behaviour and, based on the port size, entry velocity, and material rheology, is typically small, whereas the Froude number is large showing that the effect of gravity is negligible. The first calculation shows the filling sequence of the "foot" mould. In this case the complex morphology of the mould and the high curvature of the flow domain required a fine mesh. The second calculation simulates the filling of the "finger" mould (a closed hand with the forefinger pointing forward) which also involves a complicated flow domain and consequently fine meshing.

Sample input data [29] were used for the simulation in each case. These were in turn used to calculate both the Reynolds number and the Froude number and to set the initial timestep and overall fill time. Usually injection moulding materials have complex rheologies which can be sensitively dependent on other process parameters such as temperature, rate of shear, etc. For simplicity all thermal effects are ignored (a valid assumption, provided the rate of filling is much greater than the rate of heat conduction into the mould and that heat generation effects such as viscous dissipation are negligible) and an appropriate "average" value of Reynolds number ($\mathrm{Re} \approx 8$ for the "foot" mould and $\mathrm{Re} \approx 6.666$ for the "finger" mould) was used in each case. The effect of gravity was neglected since the Froude number was large for both problems ($1/\mathrm{Fr}^2 \approx O(10^{-3})$). The mesh sizes for the "foot" and "finger" moulds were 6200 and 18,972 cells, respectively. No stability problems were encountered and the convergence characteristics of GENSMAC were generally good for both these problems. The results are displayed in Figs. 15 and 16, respectively.

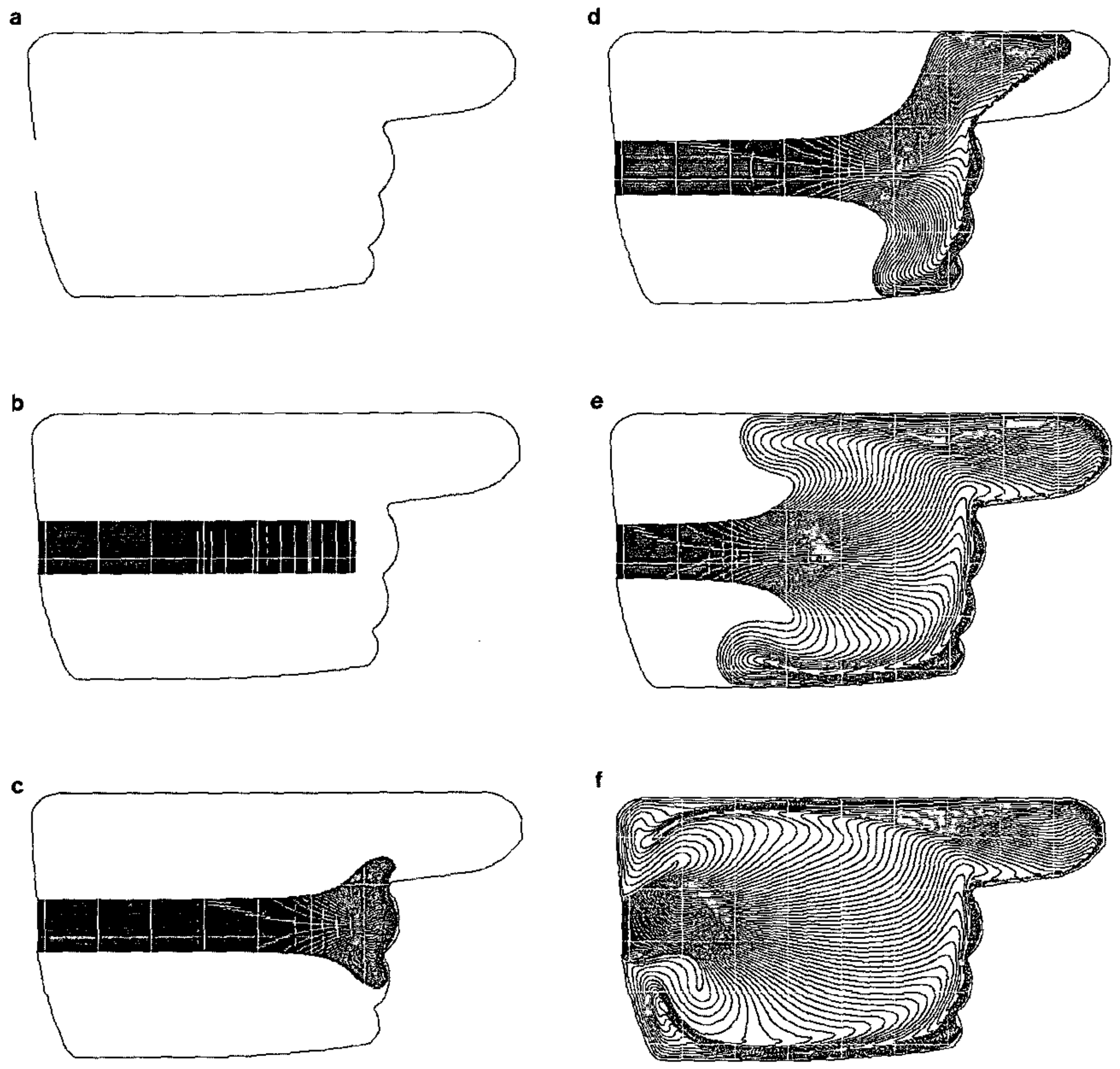

FIG. 16. The simulation of the filling process for the "finger" mould at times $t = 0.0$, 6.0, 8.0, 14.0, 26.0, and 35.21, respectively.

ACKNOWLEDGMENTS

The authors are indebted to CEGB for the COSMAC code. The authors are also indebted to Dr. B. Duffy of the University of Strathclyde and Dr. J. Crilly of Unilever Research for their careful reading of the manuscript and their helpful comments. The first author was supported by CNP_q–Conselho Nacional de Desenvolvimento Cientifico e Tecnologico under Grant 201550/88-1.

REFERENCES

1. F. Harlow and J. E. Welch, *Phys. Fluids* **8**, 2182 (1965).
2. A. A. Amsden and F. H. Harlow, Los Alamos Scientific Lab. Report LA-4370, Los Alamos, New Mexico, 1970 (unpublished).
3. M. O. Deville, *J. Comput. Phys.* **15**, 362 (1974).
4. W. E. Pracht, *J. Comput. Phys.* **7**, 46 (1971).
5. M. Golafshani, *Int. J. Numer. Methods Fluids* **8**, 897 (1988).
6. W. E. Pracht, *J. Comput. Phys.* **17**, 132 (1975).
7. F. H. Harlow and A. A. Amsden, *J. Comput. Phys.* **8**, 197 (1971).
8. J. M. Sicilian and C. W. Hirt, *J. Comput. Phys.* **56**, 428 (1984).
9. H. Miyata and S. Nishimura, *J. Comput. Phys.* **60**, 391 (1985).
10. H. Miyata and S. Nishimura, *J. Fluid Mech.* **157**, 397 (1985).
11. H. Miyata, *J. Comput. Phys.* **65**, 179 (1986).
12. J. A. Viecelli, *J. Comput. Phys.* **4**, 543 (1969).
13. J. A. Viecelli, *J. Comput. Phys.* **8**, 119 (1971).
14. J. F. MacQueen and P. Rutter, C.E.G.B. Report LM/PHYS/258, 1981.

15. G. Markham and M. V. Proctor, C.E.G.B. Report TPRD/L/0063/M82, 1983.
16. M. Tome and S. McKee, GENSMAC, Department of Mathematics Research Report 25, University of Strathclyde, 1991.
17. C. W. Hirt and J. P. Shannon, *J. Comput. Phys.* **2**, 403 (1969).
18. B. D. Nichols and C. W. Hirt, *J. Comput. Phys.* **8**, 434 (1971).
19. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method* (Prentice–Hall, Englewood Cliffs, NJ, 1973), p. 156.
20. J. E. Welch, F. H. Harlow, J. P. Shannon, and B. J. Daly, Los Alamos Scientific Lab. Report LA-3425, Los Alamos, 1966.
21. L. A. Hageman and D. M. Young, *Applied Iterative Methods* (Academic Press, New York, 1981).
22. G. Birkhoff and R. E. Lynch, *Numerical Solution of Elliptic Problems* (SIAM, Philadelphia, 1984).
23. E. F. F. Botta and M. H. M. Ellenbroek, *J. Comput. Phys.* **60**, 119 (1985).
24. C. W. Hirt and B. D. Nichols, *J. Comput. Phys.* **39**, 201 (1981).
25. L. W. Ehrich, *J. Comput. Phys.* **44**, 31 (1981).
26. P. H. Brazier, *Comput. Methods Appl. Mech. Eng.* **3**, 335 (1974).
27. N. Takemitsu, *J. Comput. Phys.* **61**, 499 (1985).
28. R. S. Varga, *Matrix Iterative Analysis* (Prentice–Hall, Englewood Cliffs, NJ, 1962), p. 199.
29. J. F. Crilly, Private communication, Unilever Research, Colworth House, Sharnbrook, Bedfordshire, 1988.
30. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum, New York, 1988), p. 191.
31. J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers* (SIAM, Philadelphia, 1991).
32. J. F. MacQueen, CERL Note No. RD/L/N 15/71, 1971 (unpublished).