Dr-932

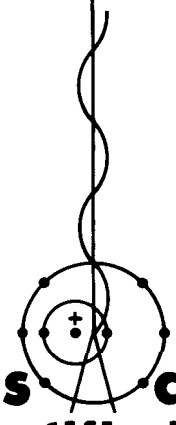
LA-5584

UC-37
Reporting Date: March 1974
Issued: July 1974

A Stochastic/Analog Simulator for Viscous, Incompressible Fluid Flow

by

James P. Shipley



los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy $4.00 Microfiche $1.45

CONTENTS

A STOCHASTIC/ANALOG SIMULATOR FOR VISCOUS, INCOMPRESSIBLE FLUID FLOW

by

James P. Shipley

ABSTRACT

A restricted form of the simplified marker-and-cell (SMAC) algorithm for calculating incompressible fluid flows is described. The SMAC algorithm is suitable for hardware implementation. The necessary hardware for building a simulator is discussed in terms of block diagrams and cell circuitry. Such a simulator is feasible if stochastic processing techniques are used. Some cost estimates are given.

I. INTRODUCTION

In recent years, many formidable problems of fluid dynamics have been solved by high-speed digital computers using various solution techniques. In fact, there seem to be almost as many solution techniques as there are problems. One unifying aspect is that all of these techniques require a large digital computer. Here we describe how a special purpose hybrid (stochastic/analog) computer might be built for solving problems of transient, viscous, incompressible fluid flows. This simulator is composed of separate blocks of circuitry, each performing the functions of a cell in a typical fluid flow mesh. The blocks can be physically arranged and interconnected in any manner to approximate the desired region of fluid flow. The simulator may prove useful to those without ready access to a suitable digital computer.

Such a simulator has become feasible because of the advent of new computing methods and circuitry called stochastic processing.[1-3] A stochastic multiplier is simpler, smaller, and less costly than either analog or digital multipliers, thus making it possible to put several multipliers in one cell.

The restricted simplified marker-and-cell (SMAC) algorithm upon which the simulator is based is described in Sec. II. The block diagrams and some of the circuitry required are discussed in Sec. III. Some of the limitations and possible extensions are listed in Sec. IV. The basic operation of the stochastic multiplier is described in the Appendix.

II. COMPUTATIONAL ALGORITHM

A. Basic Description

We calculate fluid flows here based on a special SMAC method developed by Amsden and Harlow.[4] The basic idea is to divide the region of fluid flow into fixed cells and to perform the necessary calculations for each cell in two parts, similar to a technique developed by Chorin.[5]

The first part of a calculational cycle consists of computing tentative advanced-time fluid velocities for each cell using an arbitrary pressure distribution, but with the correct velocity boundary conditions. These boundary conditions ensure that the vorticity is correct at all interior points in the fluid, but the continuity equation, $\nabla \cdot u = 0$, is not necessarily satisfied.

The second part of the cycle consists of modifying the tentative velocities to their final values so that the correct vorticity is preserved and the incompressibility (continuity) condition is satisfied.

B. The Equations

The SMAC method uses the following set of differential equations written in Cartesian coordinates:

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + a_x + \nu \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) \quad , \quad (1)$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + a_y - \nu \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) \quad , \quad (2)$$

and

$$D \equiv \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad . \quad (3)$$

The quantities u and v are the velocity components in the x- and y-directions, respectively, t is the time, ν is the fluid viscosity, and a_x and a_y are the x- and y-direction body accelerations, respectively. The pressure p has been normalized to unit fluid density. The fluid divergence D must be everywhere zero at the conclusion of each computational cycle.

The SMAC algorithm solves these differential equations by finite-difference approximation based on dividing the fluid flow region into cells as shown in Fig. 1.

The pressures are cell-centered, whereas the velocities are defined at the middle of each cell wall. Using this scheme, the finite-difference approximations to the differential Eqs. (1) - (3) can be written

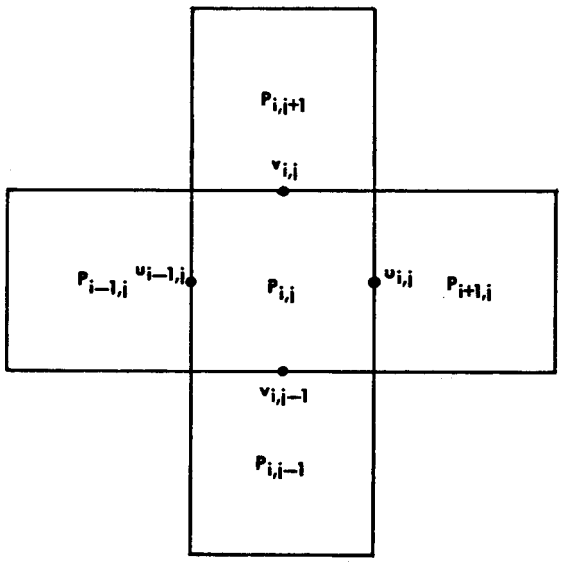


Fig. 1. Cells and variables in the SMAC algorithm.

and

$$\tilde{D}^{n+1}_{i,j} = \frac{\tilde{u}^{n+1}_{i,j} - \tilde{u}^{n+1}_{i-1,j}}{\delta x} + \frac{\tilde{v}^{n+1}_{i,j} - \tilde{v}^{n+1}_{i,j-1}}{\delta y} \quad . \quad (6)$$

The subscripts refer to locations in the cell mesh, and the superscripts count time cycles with δt being the length of one cycle. The tildes indicate tentative quantities that are present before the velocities have been corrected to satisfy the continuity equation. The tilde pressure field is arbitrary. The velocity terms in Eqs. (4) and (5) that have half-integral subscripts are calculated as simple averages.

$$\frac{\tilde{u}^{n+1}_{i,j} - u^n_{i,j}}{\delta t} = -\frac{u^n_{i,j}\, u^n_{i-1,j} - u^n_{i+1,j}\, u^n_{i,j}}{\delta x} + \frac{u^n_{i,j-1/2}\, v^n_{i+1/2,j-1} - u^n_{i,j+1/2}\, v^n_{i+1/2,j}}{\delta y} + \frac{\tilde{p}_{i,j} - \tilde{p}_{i+1,j}}{\delta x} + a_x$$

$$+ \nu \left[\frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{\delta y^2} - \frac{v^n_{i+1,j} - v^n_{i+1,j-1} - v^n_{i,j} + v^n_{i,j-1}}{\delta x \delta y} \right] \quad , \quad (4)$$

$$\frac{\tilde{v}^{n+1}_{i,j} - v^n_{i,j}}{\delta t} = -\frac{v^n_{i,j}\, v^n_{i,j-1} - v^n_{i,j+1}\, v^n_{i,j}}{\delta y} + \frac{u^n_{i-1,j+1/2}\, v^n_{i-1/2,j} - u^n_{i,j+1/2}\, v^n_{i+1/2,j}}{\delta x} + \frac{\tilde{p}_{i,j} - \tilde{p}_{i,j+1}}{\delta y} + a_y$$

$$+ \nu \left[\frac{v^n_{i+1,j} + v^n_{i-1,j} - 2v^n_{i,j}}{\delta x^2} - \frac{u^n_{i,j+1} - u^n_{i-1,j+1} - u^n_{i,j} + u^n_{i-1,j}}{\delta x \delta y} \right] \quad , \quad (5)$$

$$u^n_{i,j-1/2} = \frac{1}{2}\left(u^n_{i,j} + u^n_{i,j-1}\right) \tag{7}$$

$$u^n_{i,j+1/2} = \frac{1}{2}\left(u^n_{i,j} + u^n_{i,j+1}\right) \tag{8}$$

$$u^n_{i-1,j+1/2} = \frac{1}{2}\left(u^n_{i-1,j} + u^n_{i-1,j+1}\right) \tag{9}$$

$$v^n_{i+1/2,j} = \frac{1}{2}\left(v^n_{i,j} + v^n_{i+1,j}\right) \tag{10}$$

$$v^n_{i-1/2,j} = \frac{1}{2}\left(v^n_{i,j} + v^n_{i-1,j}\right) \tag{11}$$

$$v^n_{i+1/2,j-1} = \frac{1}{2}\left(v^n_{i,j-1} + v^n_{i+1,j-1}\right) \tag{12}$$

Next, correct the tentative velocity field by adding the gradient of a potential to it. This procedure leaves the vorticity unchanged but ensures that the velocity field satisfies the incompressibility condition. The new velocities are calculated from

$$u^{n+1}_{i,j} = \tilde{u}^{n+1}_{i,j} - \frac{\psi_{i+1,j} - \psi_{i,j}}{\delta x} \tag{13}$$

and

$$v^{n+1}_{i,j} = \tilde{v}^{n+1}_{i,j} - \frac{\psi_{i,j+1} - \psi_{i,j}}{\delta y} \quad , \tag{14}$$

where $\psi_{i,j}$ is the potential function corresponding to cell i,j. From Eqs. (6), (13), and (14)

$$D^{n+1}_{i,j} = \tilde{D}^{n+1}_{i,j} - \frac{\psi_{i+1,j} + \psi_{i-1,j} - 2\psi_{i,j}}{\delta x^2} - \frac{\psi_{i,j+1} + \psi_{i,j-1} - 2\psi_{i,j}}{\delta y^2} \quad . \tag{15}$$

Equation (15), together with appropriate boundary conditions and the requirement that $D^{n+1}_{i,j} \equiv 0$, determines the potential in every cell.

C. Boundary Conditions

The SMAC algorithm can handle five different types of boundaries. The variables at a left boundary are shown in Fig. 2.

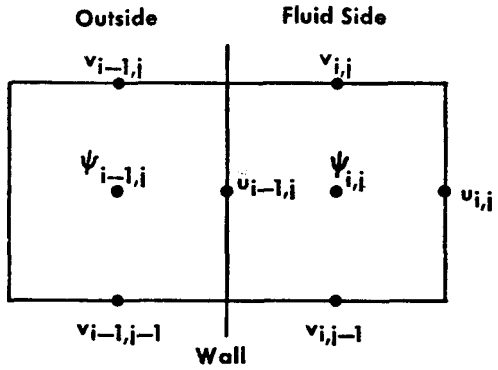


Fig. 2. Variables at a left SMAC wall.

The five types of boundaries are (refer to Fig. 2):

(1) Free-slip: A free-slip boundary represents a plane of symmetry or nonadhering surface that exerts no drag on the fluid. The boundary conditions are

$$u_{i-1,j} = 0$$

$$v_{i-1,j} = v_{i,j}$$

$$v_{i-1,j-1} = v_{i,j-1}$$

$$\psi_{i-1,j} = \psi_{i,j} \quad .$$

(2) No-slip: A no-slip boundary represents a viscous boundary that exerts a drag on the fluid forcing the tangential velocity to zero at the wall.

$$u_{i-1,j} = 0$$

$$v_{i-1,j} = -v_{i,j}$$

$$v_{i-1,j-1} = -v_{i,j-1}$$

$$\psi_{i-1,j} = \psi_{i,j} \quad .$$

(3) In-flow: An in-flow boundary allows fluid to move into the system at a prescribed inflow.

$$u_{i-1,j} = \text{prescribed inflow}$$

$$v_{i-1,j} = -v_{i,j}$$

$$v_{i-1,j-1} = v_{i,j-1}$$

$$\psi_{i-1,j} = \psi_{i,j} \quad .$$

(4) Out-flow (prescribed): A prescribed out-flow boundary allows fluid to be removed from the systems in a predetermined fashion, like a pump.

$$u_{i-1,j} = \text{prescribed outflow}$$

$$v_{i-1,j} = -v_{i,j}$$

$$v_{i-1,j-1} = v_{i,j-1}$$

$$\psi_{i-1,j} = \psi_{i,j} \quad .$$

(5) Out-flow (continuative): A continuative out-flow boundary allows fluid to pass out of the systems at its own chosen rate.

$$\tilde{u}_{i-1,j} = \tilde{u}_{i,j}$$

$$u_{i-1,j} = \tilde{u}_{i-1,j} - \frac{\psi_{i,j} - \psi_{i-1,j}}{\delta x}$$

$$\psi_{i-1,j} = 0$$

$$v_{i-1,j} = v_{i,j}$$

$$v_{i-1,j-1} = v_{i,j-1} \quad .$$

These boundary conditions can not only be used to set up the exterior boundaries of the fluid flow, but also to erect obstacles in the fluid. Although these boundary conditions are described for a left wall, those for any other wall are directly analogous.

D. General Comments

A calculational cycle for the restricted SMAC algorithm is shown in Fig. 3.

The tilde pressure field is arbitrary and is usually set to zero. However, in certain cases the hydrostatic pressure can be used to improve the calculational efficiency, and is given by

$$\tilde{p} = a_x x + a_y y \quad , \tag{16}$$

where a_x and a_y are the body acceleration components in the x- and y-directions, respectively, and x and y are the coordinates of the cell center. The hydrostatic pressure cannot be used in problems

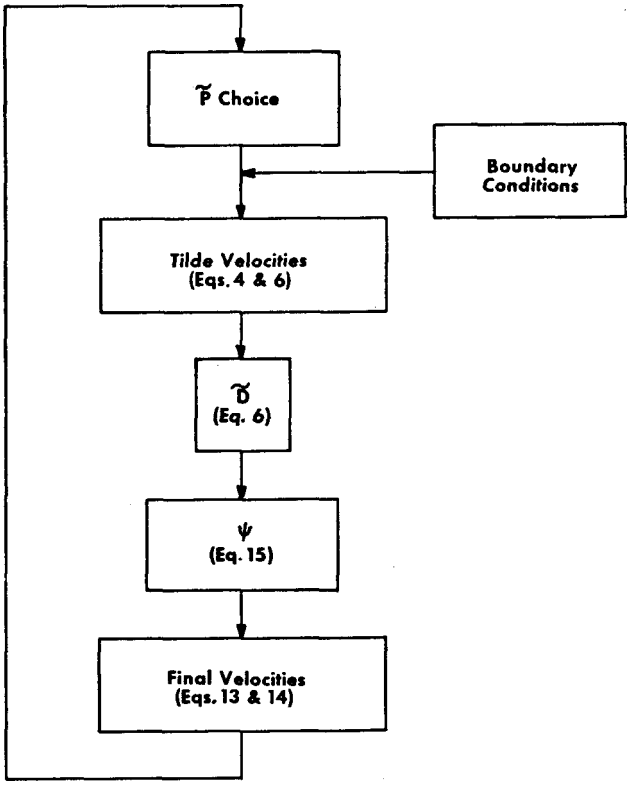


Fig. 3. The restricted SMAC calculational cycle.

involving a continuative out-flow boundary, because the requirement that D = 0 results in an incorrect velocity gradient. In this type of problem, $\tilde{p} = 0$ is used.

The time step δt and the spatial increments δx and δy must be chosen considering numerical stability and accuracy. Intuitively, δt should be such that the percentage change in the largest velocity in the mesh is small, say less than 10%. Similarly, δx and δy must be small enough so that the percentage change in velocity from cell to cell is also small. The first condition ensures that the problem is quasi-static during any time step, whereas the second condition guarantees that the mesh contains enough cells to describe adequately the fluid flow.

Similarly, the kinematic viscosity coefficient ν can cause instability if it is too small. However, because the restricted SMAC algorithm described here does not allow free surfaces, ν can be smaller than for the original SMAC method.

The potential ψ is calculated by a successive over-relaxation (SOR) technique according to Eq. (17).

$$\psi_{i,j}^{h+1} = \frac{1+\alpha}{\frac{2}{\delta x^2}+\frac{2}{\delta y^2}}\left[-\tilde{D}_{i,j} + \frac{\psi_{i+1,j}^{h+1}+\psi_{i-1,j}^{h+1}}{\delta x^2} + \frac{\psi_{i,j+1}^{h}+\psi_{i,j-1}^{h+1}}{\delta y^2}\right] - \alpha\psi_{i,j}^{h} \quad , \tag{17}$$

where h is the iteration number and α is an over-relaxation parameter lying between 0 and 1. If $\alpha = 0$, there is no over-relaxation. The starting conditions for the potentials are found from the known boundary values for ψ and from the ψ field available from the previous time step.

The restricted SMAC algorithm described here was simplified in two ways. First, every cell in the mesh is full of fluid at all times, i.e., there are no free fluid surfaces. Second, all calculations are done in rectangular coordinates, whereas the original SMAC method can use either rectangular or polar coordinates. These two simplifications greatly reduce the complexity of the algorithm, while still allowing it to solve a large class of problems.

III. HARDWARE IMPLEMENTATION

A. The Block Diagrams

Next we determine what hardware is required in each cell to implement the restricted SMAC algorithm described above. First, write down block diagrams to simulate the equations, then describe the auxiliary logic required to operate the entire mesh in a synchronous fashion.

A more convenient form of the equations for the tilde velocities is given below. A block diagram implementing Eq. (4′) is shown in Fig. 4.

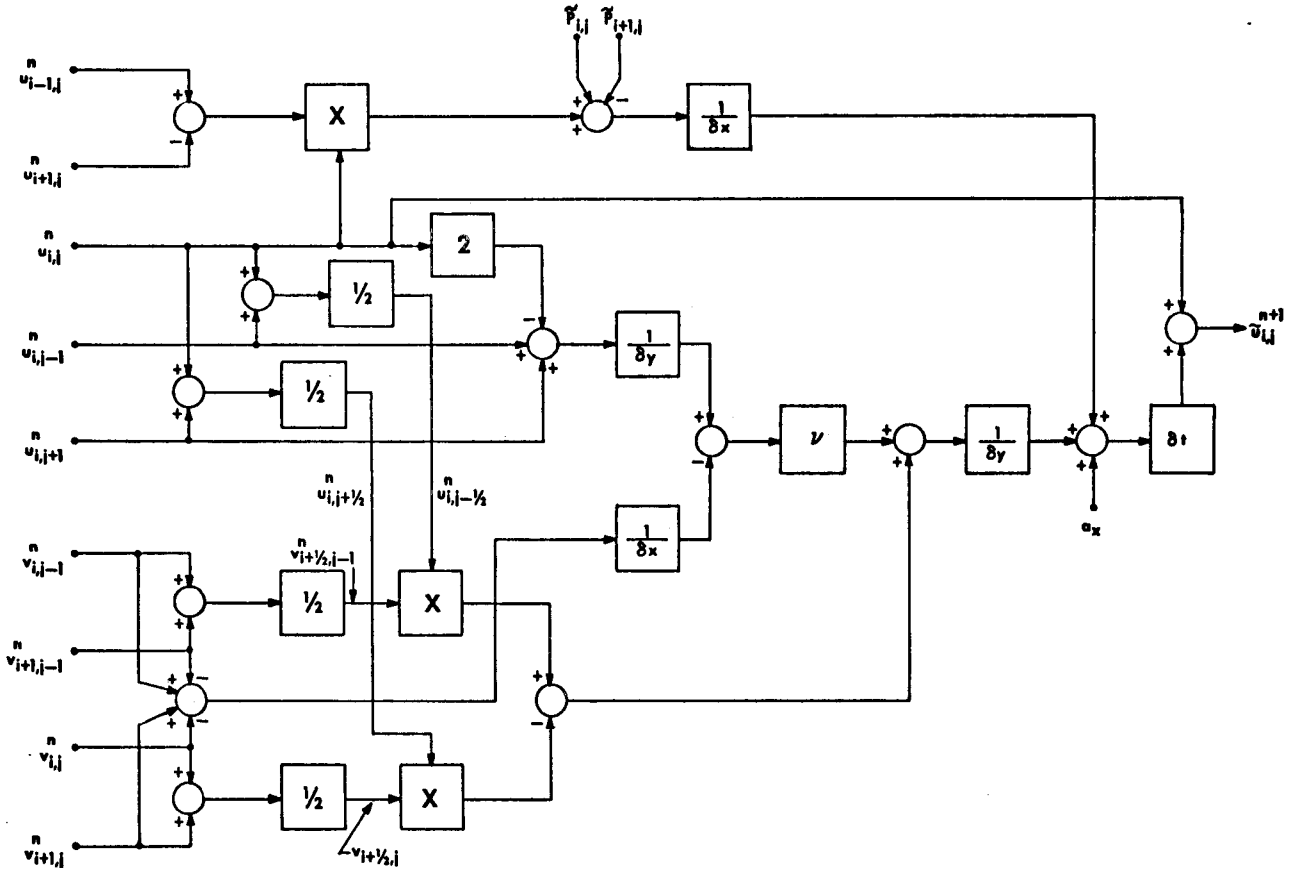


Fig. 4. A block diagram for calculating $\tilde{u}_{i,j}^{n+1}$ in one cell.

$$\tilde{u}^{n+1}_{i,j} = u^n_{i,j} + \left[u^n_{i,j}\left(u^n_{i-1,j} - u^n_{i+1,j}\right) + \tilde{p}_{i,j} - \tilde{p}_{i+1,j}\right]\frac{\delta t}{\delta x} + \left[u^n_{i,j-1/2}\, v^n_{i+1/2,j-1} - u^n_{i,j+1/2}\, v^n_{i+1/2,j}\right]\frac{\delta t}{\delta y}$$

$$+ a_x \delta t + \nu\left[\left(u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}\right)\frac{\delta t}{\delta y^2} - \left(v^n_{i+1,j} - v^n_{i+1,j-1} - v^n_{i,j} + v^n_{i,j-1}\right)\frac{\delta t}{\delta x \delta y}\right] \qquad (4')$$

and

$$v^{n+1}_{i,j} = v^n_{i,j} + \left[v^n_{i,j}\left(v^n_{i,j-1} - v^n_{i,j+1}\right) + \left(\tilde{p}_{i,j} - \tilde{p}_{i,j+1}\right)\right]\frac{\delta t}{\delta y} + \left[u^n_{i-1,j+1/2}\, v^n_{i-1/2,j} - u^n_{i,j+1/2}\, v^n_{i+1/2,j}\right]\frac{\delta t}{\delta x}$$

$$+ a_y \delta t + \nu\left[\left(v^n_{i+1,j} + v^n_{i-1,j} - 2v^n_{i,j}\right)\frac{\delta t}{\delta x^2} - \left(u^n_{i,j+1} - u^n_{i-1,j+1} - u^n_{i,j} + u^n_{i-1,j}\right)\frac{\delta t}{\delta x \delta y}\right] . \qquad (5')$$

The block diagram for implementing Eq. (5′) is functionally identical; the difference is the inputs and output. Note that the $\tilde{u}^{n+1}_{i,j}$ and $\tilde{v}^{n+1}_{i,j}$ calculations both require three multipliers, but a total of only five is required for a cell because one of the multiplications is common to both calculations.

The block diagram shown in Fig. 5 indicates how $\tilde{D}^{n+1}_{i,j}$ may be calculated from Eq. (6).

To find $\tilde{D}^{n+1}_{i,j}$ the tilde x-velocity from the left adjacent cell and the tilde y-velocity from the bottom adjacent cell are required as cell inputs.

Figure 6 shows how Eq. (17) can be implemented.

Note that the potentials from the four neighboring cells are required. A different method of potential computation is discussed below.

The last block diagram (Fig. 7) shows how the final corrected x-velocity is found. The y-velocity calculation is similar.

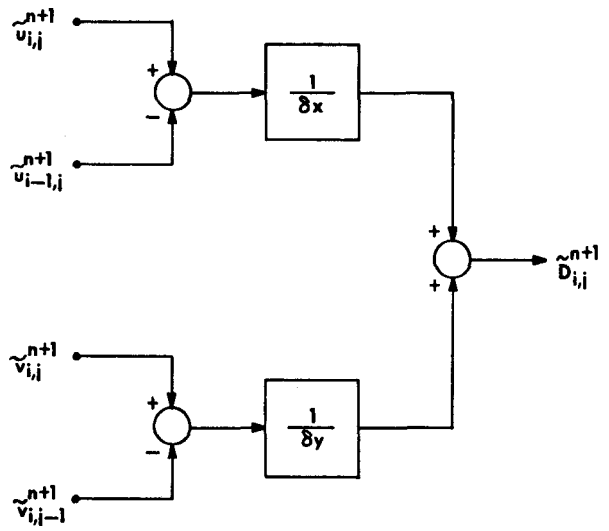


Fig. 5. A block diagram for calculating $\tilde{D}^{n+1}_{i,j}$.

B. Cell Circuitry

The multipliers are the major problem in terms of cost and complexity per cell. Both analog and digital multipliers are expensive and bulky. Further, the calculations following the multipliers are best done in an analog fashion. Therefore, it is suggested that the optimum way to implement the cell is to first convert the necessary velocity signals to synchronous random pulse sequences (SRPS) (see Appendix), perform the multiplications, convert back to analog form, then complete the remainder of the calculations using conventional analog techniques. Here, only scaling summers are required.

A simpler way to calculate the potentials is based on a resistive mesh analog to the potential equation. Consider the resistive mesh shown in Fig. 8.

Let the $\psi_{i,j}$ be represented by voltages and R_x and R_y be resistors. Writing Kierchhoff's current law equation at node i,j ,

$$\frac{\psi_{i,j} - \psi_{i-1,j}}{R_x} + \frac{\psi_{i,j} - \psi_{i+1,j}}{R_x} + \frac{\psi_{i,j} - \psi_{i,j+1}}{R_y}$$

$$+ \frac{\psi_{i,j} - \psi_{i,j-1}}{R_y} + I_D = 0 \quad . \qquad (18)$$

Solving for $\psi_{i,j}$,

$$\psi_{i,j} = \left[\frac{2}{R_x} + \frac{2}{R_y}\right]^{-1} \qquad (19)$$

$$\left[-I_D + \frac{\psi_{i-1,j} + \psi_{i+1,j}}{R_x} + \frac{\psi_{i,j+1} + \psi_{i,j-1}}{R_y}\right] .$$

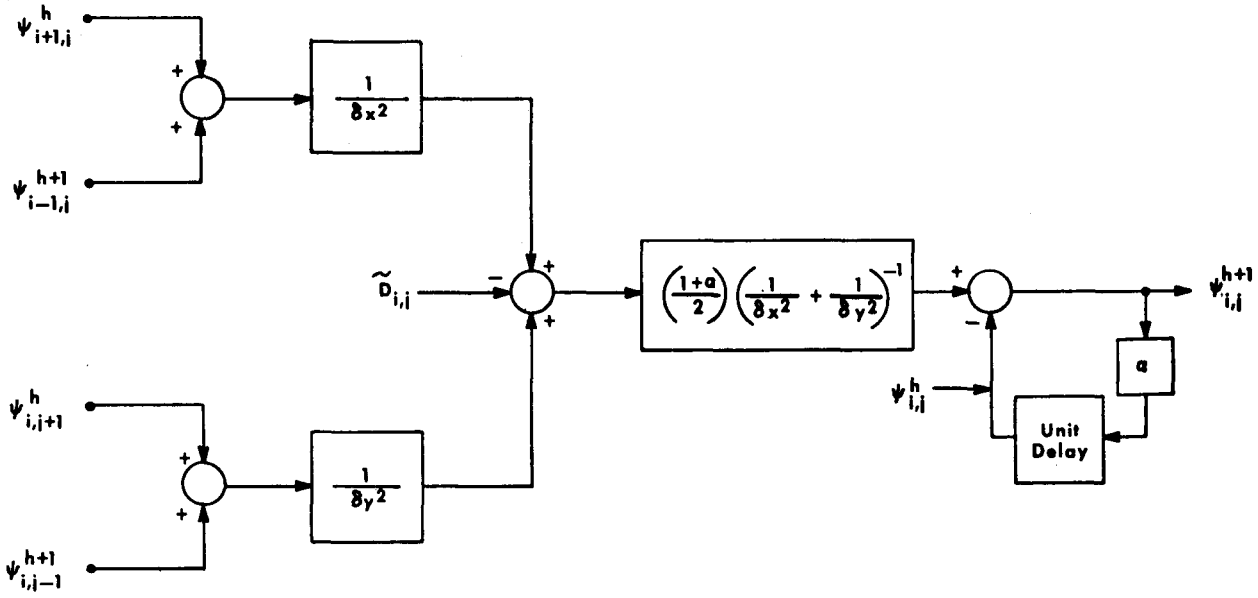


Fig. 6. A block diagram for finding $\psi_{i,j}$ by the SOR method.

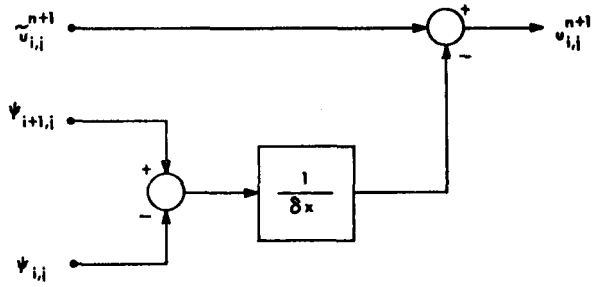


Fig. 7. A block diagram for calculating $u^{n+1}_{i,j}$.

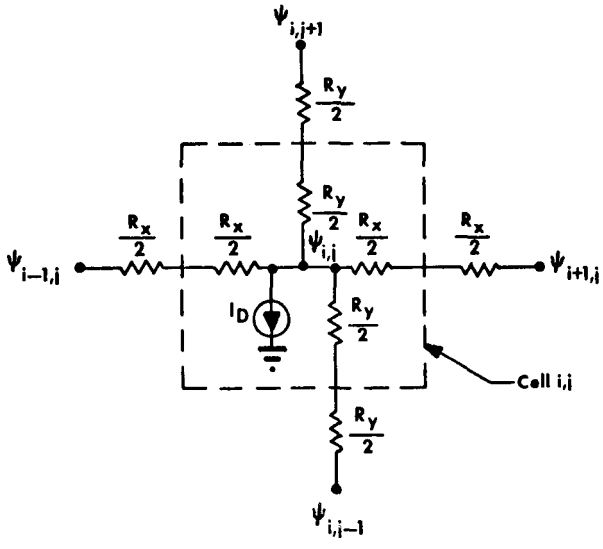


Fig. 8. Resistive mesh for finding $\psi_{i,j}$.

This is precisely the same as Eq. (17) if $\alpha = 0$ (no over-relaxation), if $R_x = \delta x^2$ and $R_y = \delta y^2$, and if $I_D = \tilde{D}_{i,j}$. Thus, a simple resistive network can replace the scaling summers that would have been required to calculate the potentials.

C. Obtaining Mesh Synchronism

Because the SMAC method is a discrete-time algorithm, the calculations must proceed synchronously throughout the mesh. However, the only quantities that need to be synchronous from cell to cell are the final velocities, thus the remaining computations can be done continuously without the need for clocking circuitry. This reduces the complexity of each cell.

The synchronous actions of the mesh can be obtained with a sample-and-hold circuit on the tentative velocities. Each tentative velocity is sampled for a short time after the previous final velocities have stabilized. If the time between samples is long enough, this time can be constant because the final velocities will always have time to stabilize.

D. Boundary Condition Handling

The only remaining problem is the hardware handling of the boundary conditions. In what follows, it is assumed that all velocities and pressures are represented by voltages. With the possible exception of the continuative out-flow boundary,

there is no need for cells lying outside the boundary. In each case, the boundary quantities lying outside the boundary are defined either by (1) quantities lying within the boundary, which can therefore be replaced where required by these inside quantities by a wire to the appropriate connection, or by (2) prescribed inflows or outflows, which can be simulated using constant voltage sources.

The continuative out-flow boundary can be handled in the same manner, except that some computation is required for the out-flow velocity (refer to the boundary conditions discussed in Sec. II.C). One way is to provide the necessary hardware in every cell so that it is always available should that cell be placed on such a boundary. The alternative is to build a special kind of cell to lie just outside a continuative out-flow boundary. This however has the disadvantage of requiring two different kinds of cells.

E. Summary

Each cell requires 10 velocity inputs (8 final and 2 tentative velocities), 3 pressures (unless these are always set to zero, which may be the case), 2 accelerations, 4 potentials, 2 spatial increments and 1 time increment (these may be fixed for all time), and the coefficient of kinematic viscosity. Each cell requires a minimum of 5 multipliers and a maximum of 32 scaling summers (this number can be cut by at least 30% by judicious rearrangement of the block diagram). In addition, two sample-and-hold circuits are required for the two tilde velocities in each cell.

Because of the cell complexity, it is difficult to imagine less than $10 per cell, and the cost could be easily three times that. At $10 per cell, a 30 by 30 cell mesh would cost $9,000, which is not unreasonable considering the cost of obtaining the SMAC algorithm capability on a general purpose digital computer.

In addition, some hardware external to the mesh is required. One need is a master clock to gate all of the sample-and-hold circuits, and another is the boundary condition voltage sources. Also, some method for reading out the final velocities must be provided. This could be done, for example, with a commutated digital voltmeter and printer. Approximately $5,000 should be allotted for this external equipment, for a total minimum cost of $14,000 for a 30 by 30 cell fluid flow simulator.

IV. LIMITATIONS AND EXTENSIONS

Two restrictions have been placed on the SMAC algorithm in implementing the simulator. First, no fluid-free surfaces are allowed, which eliminates one complex set of boundary conditions and associated computational circuitry. Second, all problems are solved in rectangular coordinates, whereas the SMAC method could use polar coordinates as well. Polar coordinates are not used because some of the terms in the polar form equations contain the coordinate r, which means that each cell is different because of its position, and circuitry to communicate this position is necessary. Neither of these limitations is fundamental and could be eliminated at the expense of cost, complexity, and ease of operation.

One desirable extension is to compressible fluids. However, this requires that account be kept of the fluid density and temperature in each cell. The SMAC method does not have this capability, thus extensive modifications in the algorithm and the hardware would be required.

Another possible extension is to multiple fluids. Again, the SMAC method with its accompanying hardware is unable to handle these kinds of problems.

Therefore, it appears that more general problems cannot be handled with this simulator without numerous changes.

V. CONCLUSIONS

A restricted form of the SMAC algorithm has been described, and a method of hardware implementation in a cell-by-cell fashion has been given. Each cell must contain a large amount of circuitry, but size and cost can probably be reduced by using integrated circuit techniques. We estimate a minimum cost of $14,000 for a 30 by 30 cell simulator, which is probably the smallest useful size. This cost may be low enough to make the simulator useful to those who have specific fluid flow problems to solve without ready access to a large digital computer.

ACKNOWLEDGMENTS

My special thanks to Professor W. J. Poppelbaum, Director of the Information Engineering Laboratory at the University of Illinois, for his initial conception of a fluid flow simulator and for his encouragement during this project.

APPENDIX

BASIC STOCHASTIC COMPUTING TECHNIQUES

The following is from the works of Afuso,[1] Poppelbaum,[2] and Poppelbaum, Afuso, and Esch.[3] Refer to their publications for a more complete discussion.

The basis for stochastic computing lies in the representation of continuous analog signals by means of a sequence of randomly occurring digital pulses whose average rate is proportional to the value of the analog signal. An equivalent statement is that the value of the analog signal determines the probability of occurrence of a pulse in the random pulse sequence (RPS). In practice, the RPS pulses are of constant width and are constrained to lie within well-defined time slots to eliminate various signal handling problems. Such a RPS is called a synchronous random pulse sequence (SRPS). Historically, such a sequence is called a stochastic sequence, and the arithmetic done with such sequences is called stochastic computing or stochastic processing. Only the stochastic multiplier is discussed here. The SRPS (Fig. A-1) is described by a voltage e(t), which consists of pulses occurring randomly in the time slots, each of height V_o and width τ, with the frequency changing randomly about an average value f, which is proportional to some analog signal. The average value of e(t) over a time $T \gg \tau$ is given by

$$V = \frac{1}{T} \int_o^T e(t)\, dt \quad ,$$

and the average duty cycle is

$$u = \frac{V}{V_o} \leq 1 \quad .$$

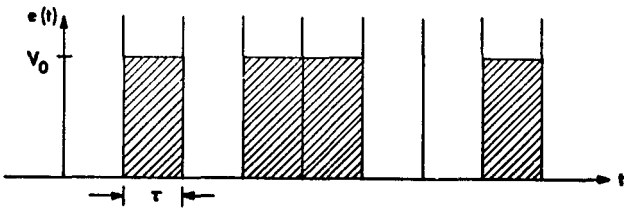


Fig. A-1. A typical SRPS.

Clearly, u is equal to the probability of finding $e(t) = V_o$ in any time slot, and the average frequency f is given by

$$f = \frac{u}{\tau} \quad .$$

Let x be the probability that a time slot contains a pulse. Then

$$x = \frac{f}{f_o} \quad ,$$

where f_o is the frequency of the time slots.

Now put two SRPSs corresponding to $x_1 = f_1/f_o$ and $x_2 = f_2/f_o$ into an AND circuit as shown in Fig. A-2.

The AND circuit has a 1 at its output only if both inputs are 1. The probability x of this event (assuming x_1 and x_2 are uncorrelated) is the product of the probability that input 1 is a 1 with the probability that input 2 is a 1, that is, $x = x_1 \cdot x_2$. Therefore, once the signals are converted to SRPS form, multiplication is a trivial operation. It is this feature that makes stochastic multiplication useful for the fluid flow simulator since numerous multiplications are required in parallel.

Some modifications are required so that negative numbers can be accommodated, but these are minor. It is also possible to do the other arithmetic operations with stochastic computing techniques.

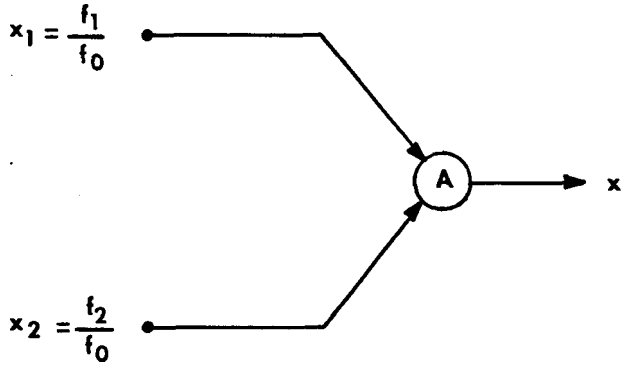


Fig. A-2. ANDing two SRPSs.

REFERENCES

1. C. Afuso, "Analog Computation With Random Pulse Sequences," Ph.D. Thesis, University of Illinois, Dept. of Computer Science Report No. 255 (1968).

2. W. J. Poppelbaum, Computer Hardware Theory, (The MacMillan Company, New York, 1972), p. 319.

3. W. J. Poppelbaum, C. Afuso, and J. W. Esch, "Stochastic Computing Elements and Systems," in Proceedings of the Fall Joint Computer Conference, Anaheim, CA, 1967.

4. A. A. Amsden and F. H. Harlow, "The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows," Los Alamos Scientific Laboratory report LA-4370 (May 1970).

5. A. J. Chorin, "Numerical Solution of the Navier-Stokes Equations," Math. Comput. 22, 745 (1968).