MATHEMATICS OF COMPUTATION, VOLUME 29, NUMBER 130
APRIL 1975, PAGES 434–446

GAP—A PIC-Type Fluid Code

By B. M. Marder

Abstract. GAP, a PIC-type fluid code for computing compressible flows, is described and demonstrated. While retaining some features of PIC, it is felt that the GAP approach is conceptually and operationally simpler.

I. Introduction. Particle-in-cell type codes have proven to be useful in simulating compressible fluid behavior, especially where free surfaces, complicated boundary conditions, or multiple species flow are encountered. In this paper, we present a variation of the PIC [1] method. It is felt that this method is conceptually and operationally simpler than PIC and, in some respects, it is more versatile.

II. The GAP Code. We shall refer to the code described in this paper as GAP (grid and particles). In GAP, as in PIC, a number of particles which represent the fluid move on a fixed background grid. These particles serve as markers of fluid position and carriers of fluid properties. For simplicity, only the one-dimensional case will be described; the two-dimensional extension is straightforward, although computationally more ambitious. It is our purpose, in this paper, only to present the basic GAP method. We will, therefore, consider only one-dimensional, one-material flow. Heat conduction and viscosity will be touched on briefly.

The system we wish to treat is described by the set of equations [2]

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0, \quad \rho\frac{\partial v}{\partial t} + \rho v\frac{\partial v}{\partial x} + \frac{\partial p}{\partial x} = 0, \tag{1}$$
$$de = Tds - pd\tau, \quad p = p(e, \rho), \quad T = T(e, \rho), \quad s = s(e, \rho),$$

where:

ρ is fluid density	τ is specific volume
v is velocity	T is temperature
e is specific internal energy	s is specific entropy

and

p is pressure.

We include the possibility of discontinuous solutions.

The particle and grid quantities used in GAP are listed below.

Received October 23, 1973.
AMS (MOS) subject classifications (1970). Primary 76–04; Secondary 65P05.

Particle Quantities		*Grid Quantities*	
M	mass	ρ	density
X	position	p	pressure
V	velocity	e	specific internal energy
E	internal energy	f	force
U	volume		

Notice that in addition to mass, position, and velocity, each particle has two independent thermodynamic variables. In this sense, each particle carries all the local fluid information. The grid serves only to store spatial averages.

In describing the GAP algorithm, we let an n subscript denote quantities associated with the nth particle while (L) refers to the Lth grid point. A "k" superscript denotes the location of the variable in time. That is, $\rho^k(L)$ is the density at time $K\Delta T$ and location $L\Delta X$. $V_n^{k+½}$ is the velocity of the nth particle at time $(k + ½)\Delta T$.

The GAP algorithm will now be described. On a fixed background grid which represents the domain of the problem, a number of particles are positioned so as to yield the desired initial conditions. By "positioning particles", we mean that values of X_n^0 are specified. Initial velocities, $V_n^{-½}$ are also supplied in a manner consistent with the desired velocity distribution. The mass, M_n, and initial volume, U_n^0, are assigned in accordance with the desired density. There is some freedom in the method of loading the particles, and some criterion such as equal masses or uniform spacing must be applied. Finally, the particle internal energy, E_n^0, is given to yield the initial pressure.

Before we begin to "push particles", we must elaborate on the relationship between grid and particle quantities. We will need to define a value of grid quantities at particle positions. That is, we will want the values of the force or the density at the particle location. We do this by a linear interpolation. If the particle position X_n lies between the L and $L + 1$ grid point, the force felt by that particle is simply

$$f_n = \delta f(L + 1) + \delta' f(L),$$

where $\delta = [(L + 1)\Delta X - X_n]/\Delta X$ and $\delta' = [X_n - L\Delta X]/\Delta X$. Notice the notation: although $f(L)$ is a grid quantity, f_n is the interpolated value of this quantity at X_n.

To obtain grid quantities, such as density, from the particles, we do essentially the inverse. Again, if X_n lies between $L\Delta X$ and $(L + 1)\Delta X$, the mass M_n contributes to both grid points. That is, to the L grid point the particle contributes $M_n\delta'$ while giving $M_n\delta$ to the $L + 1$ grid. When the sum is performed over all the particles, each grid point has a mass which is the "area weighted" total of the masses of all particles within a distance ΔX of it. All particles lying between $(L - 1)\Delta X$ and $(L + 1)\Delta X$ contribute to the mass at grid point L. When this mass is divided by the effective volume of a grid point, ΔX in the interior, $\Delta X/2$ on the boundary, the density $\rho(L)$ is obtained. A similar normalized sum over internal energy, E_n, gives the grid specific internal energy, $e(L)$.

The GAP algorithm can now be described:

1. Assume M_n, X_n^k, $V_n^{k-1/2}$, and U_n^k are known for the particles and $p^k(L)$ is known on the grid.

2. Advance V_n^k and X_n^k for each particle by

$$\frac{M_n}{U_n^k} \frac{V_n^{k+1/2} - V_n^{k-1/2}}{\Delta T} = f_n. \tag{2}$$

The force felt by each particle is obtained from the grid force defined by $f(L + 1/2) = (1/\Delta X)[p(L + 1) - p(L)]$. M_n/U_n^k is the particle's density at time step k. The relationship

$$(X_n^{k+1} - X_n^k)/\Delta t = V_n^{k+1/2} \tag{3}$$

gives the particle's new position.

3. From the new particle positions, the density, $\rho^{k+1}(L)$, is computed and, from it, new particle volumes are found by

$$U_n^{k+1} = M_n/\rho_n^{k+1}, \tag{4}$$

where ρ_n^{k+1} denotes density linearly interpolated to the particle's position, X_n^{k+1}.

4. The pdU part of the particle's internal energy is updated by

$$E_n^{k+1} - E_n^k = -p_n^k(U_n^{k+1} - U_n^k). \tag{5}$$

Alternatively, since each particle carries its thermodynamic variables, the functional form of p, $p[1/2(E_n^{k+1} + E_n^k), 1/2(U_n^{k+1} + U_n^k)]$, can be used to yield an implicit equation for E_n^{k+1}.

5. Finally, in a manner similar to the density, the grid internal energy is computed and is used with the density to update the grid pressure.

Notice that, since thermodynamic information is carried by the particles, we have a very simple form for the energy equation, (5). When the implicit form of this equation can be solved for E_n^{k+1}, second-order accuracy in time is achieved for the system (2) through (5).

To complete the numerical solution to Eqs. (1), we must include the entropy contribution to the internal energy. This is accomplished by introducing a numerical viscosity in the form of a particle drag. This viscosity serves a dual purpose. Since velocity and specific internal energy are single-valued functions of position, the quantities carried by the particles in the area should not differ much from the ensemble average. That is, large local fluctuations in particle quantities cannot be tolerated. The numerical viscosity used here enables particles to "communicate" and thus damps out local anomalies should they occur.

A simple example will help clarify these concepts. Consider the plane piston problem in which a piston is driven into a cold ($p = e = 0$) gas at rest. The analytic solution predicts a shock moving into the gas at a speed greater than that of the piston. If we use an ideal gas approximation in which $p = (\gamma - 1)e\rho$ with $\gamma = 5/3$, the density behind the shock increases by a factor of 4. Consider now what happens if we attempt to simulate this problem using the method as it has been described. As the piston encounters particles, it reflects them with twice the piston velocity. These particles stream through the undisturbed gas, but the pressure and internal energy remain zero since

$$de = -pdU \quad \text{and} \quad p = (\gamma - 1)e\rho.$$

Thus, the density will only double and the pressure will remain zero. Of course, the addition of explicit dissipation, such as viscosity or heat conduction, will generate internal energy and produce shock-like behavior, but the particle streaming will not be effectively checked.

What the numerical viscosity does is "smooth" the flow in such a manner that particle quantities will not differ too much from some average. This is done in such a way as to conserve total momentum and energy. To motivate the final form of the smoothing, we first look at some alternatives. One way to smooth would be on a cell basis. That is, we periodically compute for each cell a mean velocity such that, if each particle in that cell were given that velocity, the momentum contained in the cell would not be altered. Such a velocity is given by $\bar{V}(L) = \Sigma M_n V_n / \Sigma M_n$, where the sum is over all particles lying in the cell labeled "L". Although the momentum in the cell is unchanged by this process, the kinetic energy will be altered. If the cell contained only two particles with opposite momenta, the $\bar{V}$ would be zero, but $\frac{1}{2}M_1 V_1^2 + \frac{1}{2}M_2 V_2^2$ in kinetic energy would be lost. To conserve total energy, the change in a particle's kinetic energy is given directly to its internal energy: $\Delta E_n = -\frac{1}{2}M_n \Delta(V_n^2)$. Finally, the internal energy can be similarly smoothed over the cell so that each particle in the cell has the same specific internal energy, $E_n = M_n \Sigma E_l / \Sigma M_l$. Such a process would indeed smooth the flow and generate internal energy at the expense of kinetic. That is, it would act as an effective viscosity on a particle basis. The method described is much too crude, however, since it is discontinuous and time-step dependent. Let us consider a second approach which does essentially the same thing but does it on an area weighted basis. The method will be described, using two particles for simplicity, as illustrated in Fig. 1. The area weighted average velocity and internal energy are given symbolically by

$$\bar{V}(L) = \sum M_n V_n \delta_n \Big/ \sum M_n \delta_n,$$

where δ_n represents either δ' or δ for the nth particle. The sum is performed over all particles lying in the two cells between $(L - 1)$ and $(L + 1)$. The particle velocities are now reassigned from the $\bar{V}(L)$ by linearly interpolating from the grid.

$$\begin{aligned} V_1 &= \delta_1' \bar{V}(L-1) + \delta_1 \bar{V}(L), \quad E_1 = E_1 - \tfrac{1}{2}M_1 \Delta(V_1^2), \\ V_2 &= \delta_2' \bar{V}(L) + \delta_2 \bar{V}(L+1), \quad E_2 = E_2 - \tfrac{1}{2}M_2 \Delta(V_2^2). \end{aligned} \tag{6}$$

Again, the energy and momentum given to the grid are exactly matched by those returned to the particles, so these quantities are conserved.

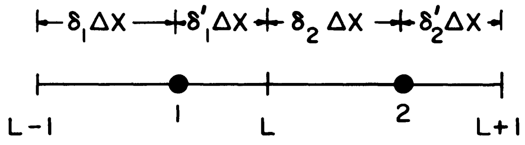

FIGURE 1. *Grid and two particles*

By comparing the mean values of velocity at three adjacent grid points before and after the smoothing operation, one can show that, for uniformly spaced particles, the dissipation is given by

$$\frac{\partial \bar{V}}{\partial t} = \frac{(\Delta X)^2}{6\Delta t}\frac{\partial^2 \bar{V}}{\partial X^2} + O(\Delta t) + O\left(\frac{\Delta X^4}{\Delta t}\right) + O\left(\frac{1}{N^2}\right), \tag{7}$$

where N is the number of particles per cell.

Thus, while this method of smoothing is continuous and superior to the first method, the amount of numerical diffusion introduced by smoothing in this manner becomes prohibitively large as Δt decreases. This can be overcome by doing essentially the smoothing described here but on a fractional basis. That is, if V_n is the initial particle velocity and $\bar{V}_n$ is the interpolated velocity obtained from $\bar{V}(L)$ which the particle would have if smoothed, the desired velocity, $\tilde{V}_n$, lies somewhere between:

$$\tilde{V}_n = V_n + F(\bar{V}_n - V_n), \qquad 0 \leqslant F \leqslant 1. \tag{8}$$

If $F = 0$, no smoothing occurs; if $F = 1$, total smoothing is performed. The dissipation resulting from this approach has the form

$$\frac{\partial V}{\partial t} = \frac{\Delta X^2}{6\Delta t} F \frac{\partial^2 V}{\partial X^2} + O(\Delta t) + O\left(F\frac{\Delta X^4}{\Delta t}\right) + O\left(\frac{1}{N^2}\right). \tag{9}$$

Thus, if F is made proportional to Δt, the numerical diffusion will be independent of time step. This means, we smooth more often as the time step decreases, but we do less each time. From the definition of $\bar{V}_n$, we see that momentum is still conserved:

$$\sum_n M_n \tilde{V}_n = \sum_n M_n V_n + F\left(\sum M_n \bar{V}_n - \sum M_n V_n\right) = \sum_n M_n V_n. \tag{10}$$

A final requirement must be met before we arrive at the final form of the smoothing operation. It is desirable to be able to apply the smoothing selectively in space. We generally want a small amount of smoothing to prevent local fluctuations from developing, but we need to provide additional smoothing in shocks and perhaps none at all in rarefactions. The smoothing parameter F should be sensitive to local conditions. To achieve this, we make it a quantity carried with the particles and denote it by F_n. To insure momentum conservation with a particle F_n, we modify our definition of $\bar{V}_n$, the mean velocity toward which the particle velocities tend. Assuming F_n has been determined for each particle, we redefine

$$\bar{V}(L) = \sum M_n F_n V_n \delta_n \Big/ \sum M_n F_n \delta_n. \tag{11}$$

If we now let $\bar{V}_n$ be interpolated from this new $\bar{V}(L)$, the new particle velocity $\tilde{V}_n$ is given by

$$\tilde{V}_n = V_n + F_n(\bar{V}_n - V_n), \tag{12}$$

and we again achieve momentum conservation:

$$\sum M_n\tilde{V}_n = \sum M_n V_n + \sum M_n F_n \bar{V}_n - \sum M_n F_n V_n = \sum M_n V_n. \tag{13}$$

Using this method, the dissipation becomes

$$\frac{\partial V}{\partial t} = \frac{\Delta X^2}{6\Delta t}\frac{\partial}{\partial X}\left(\bar{F}\frac{\partial v}{\partial X}\right) \tag{14}$$

plus lower-order terms. $\bar{F}$ is an average taken over the F_n's in a cell. The derivation of (14) is given in the appendix.

The smoothing is performed along with the rest of the algorithm and requires no additional passes through the particle table. There is no stability criterion associated with this artificial viscosity, it has a purely stabilizing effect.

The inclusion of the smoothing operation into the basic algorithm modifies (1), the system of equations being solved. Mass conservation is still rigorously maintained, but velocity and energy smoothing introduce viscosity and heat conduction. Thus, from the form of Eq. (14), we find that the physical system being modeled is given by

$$\begin{gathered}
\frac{\partial \rho}{\partial t} + \frac{\partial(\rho V)}{\partial X} = 0, \\
\rho\frac{\partial V}{\partial t} + \rho V\frac{\partial V}{\partial X} + \frac{\partial P}{\partial X} - \frac{\partial}{\partial X}\left(\sigma_1\frac{\partial V}{\partial X}\right) = 0, \\
de = Tds - pd\tau, \\
\rho T\left(\frac{\partial S}{\partial t} + V\frac{\partial S}{\partial X}\right) = \sigma_2\left(\frac{\partial V}{\partial X}\right)^2 + \frac{\partial}{\partial X}\left(\sigma_1\frac{\partial e}{\partial X}\right), \\
T = T(e, \rho),
\end{gathered} \tag{15}$$

where σ_1 and σ_2 are related to the $\bar{F}$ of Eq. (14). The system is solved by the GAP algorithm to first order in ΔT, second order in ΔX, and second order in $1/N$.

We see from (15) that viscosity and heat conduction can be incorporated into the GAP algorithm through the smoothing operation. The coefficients σ_1 and σ_2 are obtained from the particle F_n's and can be given as functions of the particle thermodynamic variables, E_n and U_n. If σ_1 and σ_2 are not equal, different F_n's must be defined for the energy smoothing and velocity smoothing. To treat shocks, however, the F_n is chosen to yield a shock thickness of 2 or 3 ΔX rather than from physical considerations.

We conclude the presentation of the GAP algorithm with a discussion of the method of determining F_n in the absence of viscosity or heat conduction. We have seen that F_n must satisfy three requirements:

1. $0 \leqslant F_n \leqslant 1$,
2. F_n proportional to Δt,
3. F_n very small in smooth isentropic regions but sufficiently large in shock regions to give sharp shocks.

Determining a form of F_n is not unlike determining a form for a von Neumann [3] type artificial viscosity; both, a functional form and a coefficient, are required. A form that works well is

$$F_n = \begin{cases} 1 \\ \alpha \Delta X \Delta t \dfrac{d}{dt} \dfrac{1}{U_n} = F_n^* \\ 0 \end{cases} \text{if} \begin{cases} F_n^* \geqslant 1, \\ 0 \leqslant F_n^* \leqslant 1, \\ F_n^* \leqslant 0. \end{cases} \tag{16}$$

The coefficient "α" is chosen such that $F_n \approx 1$ in strongly shocked regions. Notice that in this definition of F_n only particle quantities enter. The F_n given by (16) is only a suggestion, other forms may be equally serviceable. The form of F_n used in the following problems is (16). For the piston problem, an additional term $c_s \Delta t / \Delta X$ was added.

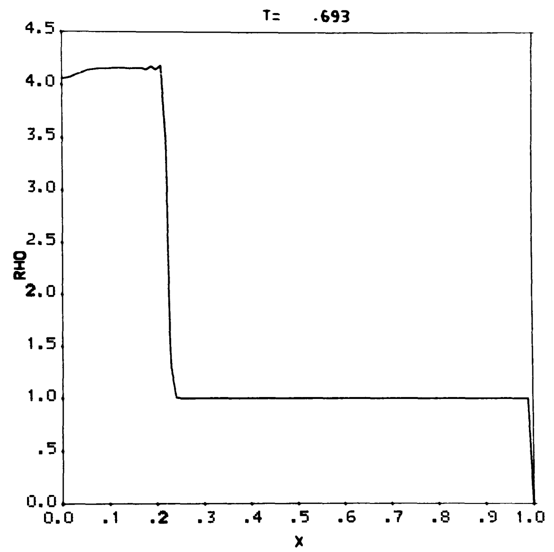

FIGURE 2. *Infinite strength shock.* $c_s \Delta T / \Delta x = .67$

Figures 2 and 3 show the density profile of an infinite strength shock obtained using the GAP code. In Fig. 2, $c_s \Delta t / \Delta x = .67$, while, in Fig. 3, $c_s \Delta t / \Delta x = .0067$, where c_s is the sound velocity behind the shock. In both runs, γ was taken to be 5/3. The density takes on nearly its predicted value of 4 behind the shock. The pressure and shock speed also assume correct values. Five particles per cell were used with 100 cells. The shock thickness is under $3\Delta x$. Although the time step in Fig. 3 is one-

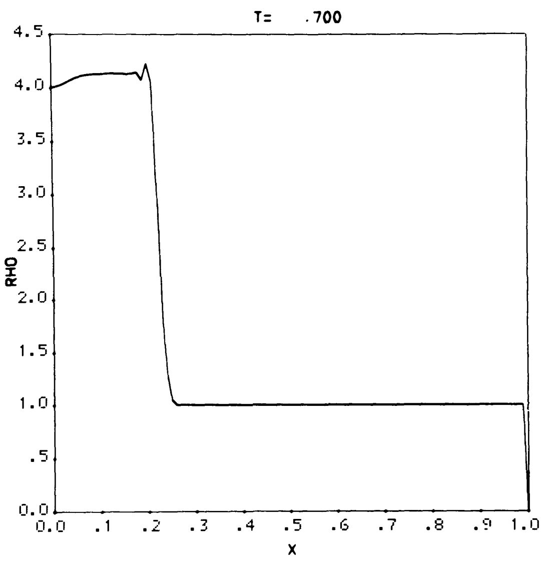

FIGURE 3. *Infinite strength shock.* $c_s\Delta T/\Delta x = .0067$

hundredth that of the one used in Fig. 2, the shock thickness is virtually unchanged and no oscillations are induced behind the shock.

Figures 4 and 5 show the density and sound speed squared, $\gamma p/\rho$, obtained in a a rarefaction of gas into a vacuum. $c_s\Delta t/\Delta x = .7$ where c_s is the sound speed of the undisturbed gas. Notice that the sound speed is well behaved at the gas-vacuum interface, indicating that the code did not generate unwanted entropy in the expansion. This problem was initiated with 90 particles per cell occupying a third of the 100 available cells.

A final one-dimensional test was performed on the diaphragm problem. Two regions of gas at equal temperature but different densities are separated by a diaphragm which is removed at $t = 0$. We took a density ratio of two, with 20 and 10 particles per cell. Figure 6 shows the subsequent density profile with a rarefaction, contact discontinuity, and shock. The numerical solution differs by less than 1% of the analytic solution in all regions except in the slight jitter by the contact discontinuity.

The three test problems presented here, while quite simple, are not at all trivial. As would be expected of any useful fluid code, GAP accurately handles shocks, rarefactions, and contact discontinuities.

In the original PIC code, only the position and mass are retained as particle quantities. All other information is stored as grid variables, and the particles transport this information between cells. As the particles move from one cell to another they carry a portion of the cell's mass, momentum, and energy with them to their new cell. The momentum and energy equations are solved on the grid as partial differential equations. In GAP, because each particle carries all its thermodynamic properties, the energy equation can be solved as an ordinary difference equation. Since no partial

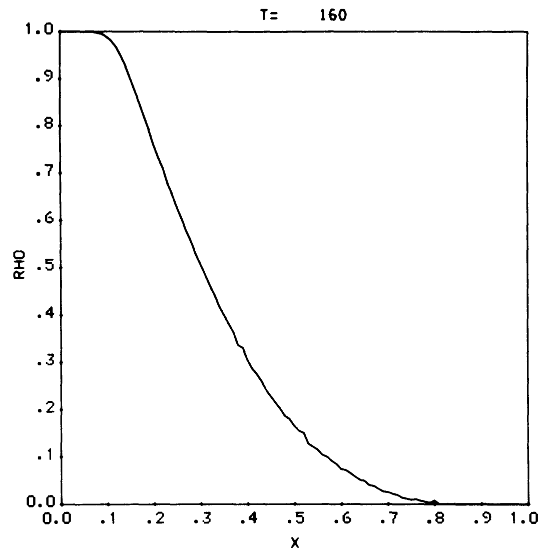

FIGURE 4. *Density in rarefaction*

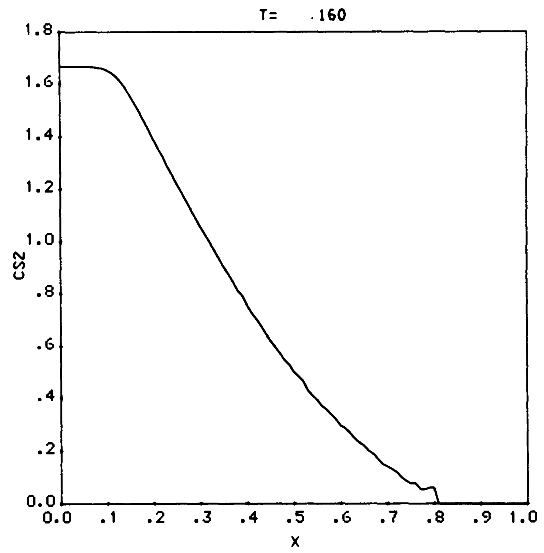

FIGURE 5. *Sound speed squared in rarefaction*

differential equations are explicitly solved and since the smoothing operation directly influences each particle even when it does not cross cell boundaries, the stability properties of GAP tend to be quite good.

Extending the code to handle two-dimensional flows is straightforward in a plane geometry but requires some care in cylindrical coordinates. In that case, the particles may be thought of as hoops about the axis of symmetry, therefore, to achieve a uniform initial density, either the number of particles must increase with radius or the

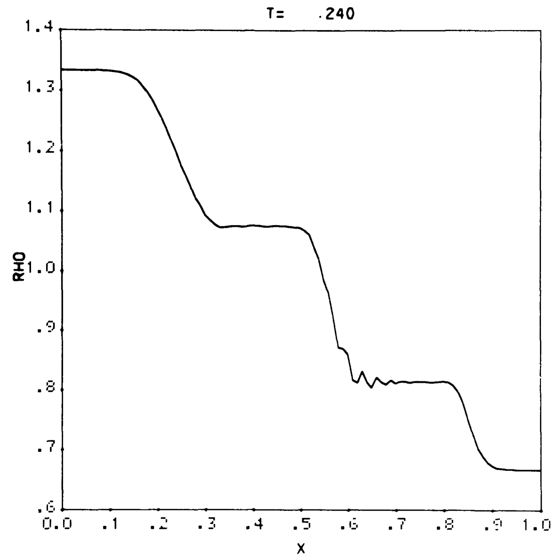

FIGURE 6. *Diaphragm problem*

mass of the particles must be proportional to the initial radius with a uniform loading. The latter method gives better resolution near the axis but is more complicated.

III. An Application. Two-dimensional versions of this code are being used to study a problem in magnetohydrodynamics: the simulation of the plasma focus [4] experiment. In this device, a plasma is magnetically driven down and around a cylindrical electrode producing very high temperatures and densities at the center. Figures 7 through 9 depict the time history of such a device as run with the GAP code on a CDC 7600. The cylindrical electrode has a radius of about 2.4 cm and is 10 cm long. The first 3 cm are surrounded by an insulating ring across which the plasma first breaks down. A current is driven along the plasma-vacuum interface between the inner electrode and outer wall. This creates a magnetic field in the (white) vacuum region which interacts with the current to produce a $J \times B$ force on the plasma. The darker regions in the plasma represent compressed gas and are separated from the undisturbed plasma by a shock front. There were 25 particles per cell on a 30×40 grid and the computing was 2 minutes. An attempt was made in this simulation to describe the experiment with sufficient accuracy to influence future design considerations. To this end, the external circuitry as well as the plasma-produced inductance were used to compute currents and voltages.

Since the experiment was being performed in conjunction with the numerical calculations, it was possible to check the code's reliability by how accurately it predicted the experimental behavior. In this respect, it was quite good. The gross hydrodynamics of the experiment were very accurately reproduced. That is, the time at which the shock hits the outer electrode, the time when it reaches the end of the inner electrode, and the time of peak collapse were correct to within a tenth of a microsecond.

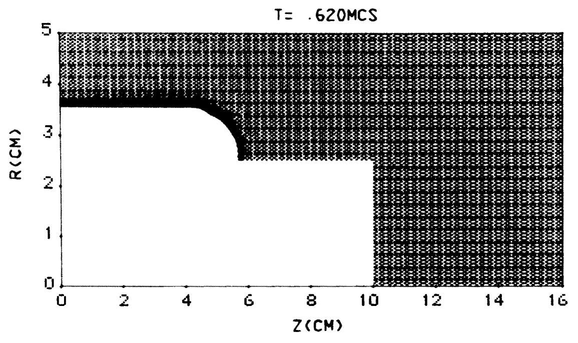

FIGURE 7. *Simulation of plasma focus using GAP*

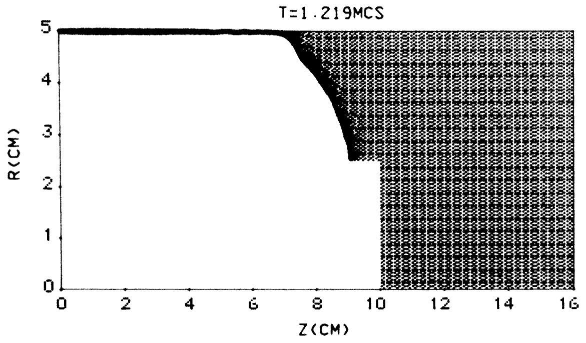

FIGURE 8. *Simulation of plasma focus using GAP*

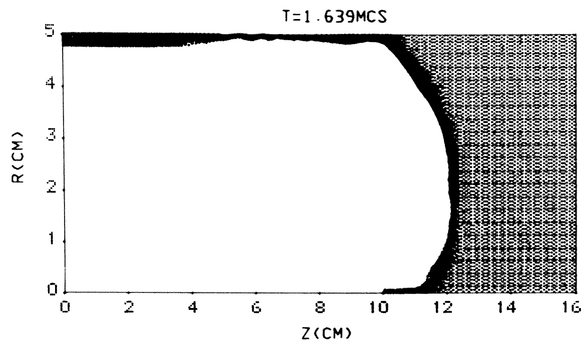

FIGURE 9. *Simulation of plasma focus using GAP*

A word about boundary conditions might be appropriate here. On solid surfaces or surfaces of reflection, one has zero normal velocities. A particle which passes through such a surface can be elastically, or inelastically, reflected. That is, its position, velocity, and internal energy are suitably altered. In addition, the normal component of the grid velocity used in smoothing is set to zero. Particle-in-cell codes such as GAP and PIC are well suited to handle such boundary conditions.

IV. Some Concluding Remarks. In this section, I would like to review some of the differences between GAP and the earlier PIC code. One of the fundamental differences is the manner in which the two codes treat thermodynamic variables. The separation of the entropy generation and numerical smoothing from the rest of the GAP algorithm allows one to have both temporal and spatial control over the amount of numerical diffusion introduced through the parameter F_n. Thus, except where it is required, this diffusive entropy generation can be kept small, or even eliminated. Another desirable feature of the artificial viscosity used in GAP is that it enters as a viscous particle drag—not as a partial differential equation. Thus, it does not require that any stability criterion be satisfied. Although PIC has a type of smoothing which apportions particle quantities uniformly in a cell, an additional von Neumann viscosity is required in stagnation regions. The form of the GAP smoothing also allows the particles to better resolve quantities on a subgrid level,

One obvious disadvantage to GAP is that since the particles retain more of an individual identity, more memory is required. With large-core modern computers and extended memory devices, unavailable when PIC was developed, this may not be too serious. GAP requires neither more particles per cell nor more cells than PIC.

Finally, since the logic in GAP is quite simple, with only ordinary differential equations being solved, the programming tends also to be simple and straightforward.

Acknowledgments. I wish to thank Colonel George Cudahy for his helpful suggestions. I appreciate also the discussions with Dr. Francis Harlow and his coworkers which helped me understand the excellent earlier work in this field.

Appendix. Equation (14) is derived in this section. Referring to Fig. 1, assume each cell contains N evenly spaced particles. The particles in the cell $(L - 1, L)$ will be subscripted with "n" while for those in $(L, L + 1)$ we will use "m". Thus, ΣV_m means the sum of the velocities of particles in the second cell. Assume that all particles have the same mass.

Before any smoothing is performed, we can define an average velocity at grid location "L":

$$V^*(L) = \sum (V_n\delta_n + V_m\delta'_m)\Big/\sum(\delta_n + \delta'_m).$$

Ignoring the contributions of particles lying before $(L - 1)$ or past $(L + 1)$, we obtain the mean velocities defined by Eq. (11)

$$\bar{V}(L-1) = \sum V_n f_n \delta_n \Big/ \sum f_n \delta'_n, \qquad \bar{V}(L+1) = \sum V_m f_m \delta_m \Big/ \sum f_m \delta_m,$$

$$\bar{V}(L) = \sum (V_n f_n \delta_n + V_m f_m \delta'_m) \Big/ \sum (f_n \delta_n + f_m \delta'_m).$$

The new particle velocities defined by (12) are

$$\tilde{V}_n = V_n + F_n\{\bar{V}(L-1)\delta'_n + \bar{V}(L)\delta_n - V_n\},$$

$$\tilde{V}_m = V_m + F_m\{\bar{V}(L)\delta'_m + \bar{V}(L+1)\delta_m - V_m\}.$$

Using these, the new average velocity at "L" is

$$V^{**}(L) = \sum(\tilde{V}_n\delta_n + \tilde{V}_m\delta'_m)\Big/\sum(\delta_n + \delta'_m).$$

We then have

$$\begin{aligned} V^{**}(L) - V^{*}(L) = \Big\{ & \bar{V}(L-1)\sum F_n\delta_n\delta'_n + \bar{V}(L)\sum F_n\delta_n^2 \\ & + \sum(1 - F_n)V_n\delta_n + \bar{V}(L)\sum F_m\delta_m'^2 \\ & + \bar{V}(L+1)\sum F_m\delta_m\delta'_m \\ & + \sum(1 - F_m)V_m\delta'_m - \sum V_n\delta_n - \sum V_m\delta'_m\Big\}\Big/\sum(\delta_n + \delta'_m). \end{aligned}$$

The assumption of uniform loading implies

$$\delta_n = \delta_m = (2n-1)/2N, \qquad 1 \leqslant n \leqslant N, \qquad \delta'_n = 1 - \delta_n.$$

We also assume that F_n is constant in a cell so that $F_n = \bar{F}(L - ½)$ and $F_m = \bar{F}(L + ½)$. Using the relations

$$\sum\delta_n = N/2 \quad \text{and} \quad \sum\delta_n^2 = N/3 - 1/12N,$$

we find

$$\begin{aligned} V^{**}(2) - V^{*}(2) = & \frac{1}{6}\bar{V}(L+1)\bar{F}(L+½) + \frac{1}{6}\bar{V}(L-1)\bar{F}(L-½) \\ & - \frac{1}{6}\bar{V}(L)(\bar{F}(L+½) + \bar{F}(L-½)) + O\left(\frac{1}{N^2}\right), \end{aligned}$$

which can be written

$$\frac{V^{**}(L) - V^{*}(L)}{\Delta t}$$
$$= \frac{1}{6}\frac{\Delta X^2}{\Delta t}\left\{\bar{F}(L+½)\frac{\bar{V}(L+1) - \bar{V}(L)}{\Delta X} - \bar{F}(L-½)\frac{\bar{V}(L) - \bar{V}(L-1)}{\Delta X}\right\}\Big/\Delta X + O\left(\frac{1}{N^2}\right).$$

Finally, we arrive at Eq. (14),

$$\frac{\partial V}{\partial t} = \frac{1}{6}\frac{\Delta X^2}{\Delta t}\frac{\partial}{\partial X}\left(\bar{F}\frac{\partial V}{\partial X}\right) + O(\Delta t) + O\left(\bar{F}\frac{\Delta X^4}{\Delta t}\right) + O\left(\frac{1}{N^2}\right).$$

Los Alamos Scientific Laboratory
University of California
P. O. Box 1663
Los Alamos, New Mexico 87544

1. F. H. HARLOW, "The particle-in-cell computing method for fluid dynamics," *Methods in Computational Physics*, vol. 3, Academic Press, New York, 1964.

2. R. COURANT & K. O. FRIEDRICHS, *Supersonic Flow and Shock Waves*, Interscience, New York, 1948. MR 10, 637.

3. J. VON NEUMANN & R. D. RICHTMYER, "A method for the numerical calculation of hydrodynamical shocks," *J. Appl. Phys.*, v. 21, 1950, pp. 232–237. MR 12, 289.

4. J. W. MATHER & P. J. BOTTOMS, "Characteristics of the dense plasma focus discharge," *Phys. Fluids*, v. 11, 1968, pp. 611–618.