JOURNAL OF COMPUTATIONAL PHYSICS 65, 314–343 (1986)

FLIP: A Method for Adaptively Zoned, Particle-in-Cell Calculations of Fluid Flows in Two Dimensions*

J. U. BRACKBILL† AND H. M. RUPPEL

Los Alamos National Laboratory, Los Alamos, New Mexico 87545

Received January 3, 1985; revised October 23, 1985

A method is presented for calculating fluid flow in two dimensions using a full particle-in-cell representation on an adaptively zoned grid. The method has many interesting properties, among them an almost total absence of numerical dissipation and the ability to represent large variations in the data. The method is described using a standard formalism and its properties are illustrated by supersonic flow over a step and the interaction of a shock with a thin foil.

INTRODUCTION

The particle-in-cell (PIC) representation has been used successfully to model many aspects of highly distorted flow in two dimensions. [1–3]. In PIC, the fluid is represented by Lagrangean mass points, called particles, moving through a computation grid. Other properties of the fluid are assigned temporarily to the particles in proportion to their mass to convect information from one cell of the grid to another. Because of the Lagrangean description of the mass distribution, PIC codes are notably successful in tracking contact discontinuities and in modeling highly distorted flow. This capability has been exploited in a wide variety of applications, including the modeling of hypervelocity impact, free surface motion, multimaterial and chemically reacting flow, and magnetohydrodynamics [3].

Here, our goal is to make PIC even more useful by extending it to an adaptively zoned grid [4]. With such a grid, which can resolve fine detail in a large system, we can use the information carried by the particles more effectively in the solution of the dynamical equations, especially when large changes in scale occur or singular boundary layers form.

In extending PIC, we must ask first with which PIC formulation we should begin. Since the development of the original PIC, many attempts have been made to increase the accuracy of PIC [5–7]. As is well known, the accuracy of PIC is

 This work was performed under the auspices of the U.S. Department of Energy.

† Present address: Division of Applied Mathematics, Brown University, Providence, R.I. 02912.

0021-9991/86 $3.00

compromised by a large computational diffusion caused by transferring the momentum from the grid to the particles and back in modeling convection through the grid.

In increasing the accuracy of PIC, researchers have followed one of two approaches. In work typical of the first approach, Nishiguchi and Yabe [5] have developed more accurate momentum convection algorithms for PIC. As in the original PIC, they transfer momentum information to the particles from the grid before the particles are displaced, and then back afterwards. To increase accuracy, they have restricted the transfers from the grid to the particles and back to changes in momentum due to particle displacements, rather than to the momentum itself. Thus, the diffusion introduced varies as the particle displacement, rather than as the number of cycles in the calculation.

The second approach is suggested by the use of PIC in plasma simulation [8] and has been followed by a number of researchers [6, 7]. In plasma simulation, each numerical particle represents a large number of physical particles in a plasma. In addition to position and mass, each numerical particle is assigned a charge and a velocity from which a distribution function can be constructed. The grid carries no permanent information and acts only as a computational convenience in organizing the particle data while calculating the interaction between particles.

The success of PIC in plasma simulation suggests assigning more information to each particle in fluid flow calculations for greater accuracy. McCrory *et al.* [6] and LeBoeuf *et al.* [7], for example, use full particle models in which they assign internal energy and momentum to each particle. In their methods, reductions in numerical diffusion are achieved, but difficulties with multistreaming and numerical noise are observed and must be dealt with. In spite of these difficulties, we follow these examples in developing FLIP (fluid-implicit-particle), an algorithm for fluid flow on an adaptive grid.

First, by assigning momentum to each particle as described in Refs. [6, 7], we eliminate a major source of computational diffusion. Since this introduces the possibility of particle multistreaming, in which two particles at the same point in space may have different velocities, we distinguish between the velocity with which a particle moves through the mesh and the momentum it carries, similar to Harlow's original PIC method [1] and to the full particle model of McCrory *et al.* [6]. Thus, we introduce the effect of collisions and eliminate multistreaming without introducing dissipation.

Second, we show that using bilinear interpolation and quadrilateral zones to describe the particle and grid kinematics allows us to divide the computation cycle into Lagrangean and convection phases. During the Lagrangean phase, the particles play no role except to provide data for the calculation on the grid. When, as in other PIC methods, we solve the equations of motion on the grid we may choose from among many solution algorithms including implicit methods, which are useful in problems with voids. We are also free to extend the method to more complicated models, such as magnetohydrodynamics. (Our examples are calculated using the artificial viscosity method.)

Third, we observe that the information carried by the particles is enough to characterize the fluid flow, reducing the grid to a computational convenience as in a plasma simulation. Thus, we may use an adaptive grid with arbitrary displacements from one time step to the next without loss of information.

In outline, we describe a particle-in-cell method for an adaptive mesh, FLIP, and present the results of a shock-on-step calculation and of a shock interacting with a thin shell. We intend to provide enough information so that the rather simple code corresponding to the method can be written. Thus, we include not only a description of the FLIP representation, the transfer of information to and from the grid, and its kinematic properties, but also an artificial viscosity method for the solution of the fluid equations on a Lagrangean mesh.

1. The Particle-in-Cell Method

a. *The Particle Kinematics*

In the particle-in-cell method, FLIP, the fluid is represented by Lagrangean fluid elements, called particles, moving through a grid. The information carried by each particle specifies the mass, momentum, internal energy, and constitutive properties of the fluid in the vicinity of the particle. The information transferred from the particle to the grid by projection permits the solution of the dynamical equations on a relatively small number of highly ordered points. As in other methods of this type,

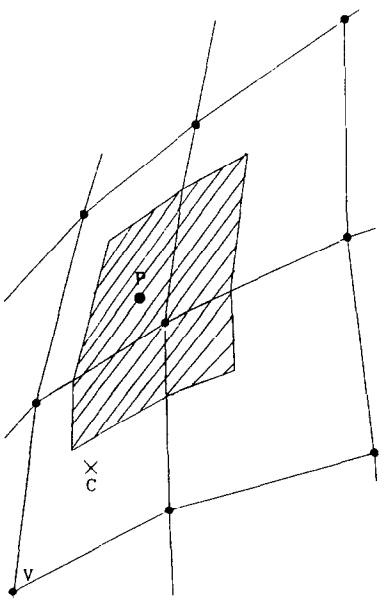

FIG. 1. A grid of nonrectilinear, quadrilateral zones is illustrated. Vertices, cell-centers, and particles are labeled v, c, and p, and the shaded area indicates generalized area weighting.

the cooperative use of the particles and grid results in an efficient algorithm for highly distorted flows with little numerical diffusion.

Tesselating the domain, we construct a grid of convex quadrilaterals as shown in Fig. 1 with vertices at positions $\mathbf{x}_v$. (The arbitrary shape of the quadrilateral zones distinguishes FLIP from previous PIC algorithms for nonuniform grids [9].) For convenience, we define natural coordinates (ξ, η) by mapping each quadrilateral onto a unit square in the space of the natural coordinates. At each vertex, the natural coordinates assume integer values (i, j). Elsewhere, the mapping between physical and natural coordinates is defined by bilinear interpolation, which can be written

$$\mathbf{x} = [\xi'\{(1-\eta')\,\mathbf{x}_{i+1,j} + \eta'\mathbf{x}_{i+1,j+1}\} + (1-\xi')\{(1-\eta')\,\mathbf{x}_{ij} + \eta'\mathbf{x}_{ij+1}\}], \tag{1}$$

where $\xi' = \xi - i$, $\eta' = \eta - j$. When $0 \leqslant \xi'$, $\eta' \leqslant 1$, x is within the quadrilateral cell (i, j).

More generally, the interpolation can be written in the form

$$\mathbf{x} = \sum_v \mathbf{x}_v s(\xi - i, \eta - j), \tag{2}$$

where s is a positive, continuous function with range $[0, 1]$ and compact support. The function is normalized in the usual way,

$$\iint_D d\xi\, d\eta\, s = 1; \qquad \sum_{i,j} s(\xi - i, \eta - j) = 1.$$

There is a natural correspondence between bilinear interpolation and quadrilateral zones. Consider displacing the vertices of the grid and constructing new quadrilaterals with the same connectivity as the original mesh. Further, consider points on the original and displaced meshes with the same natural coordinates. It follows that the displacements of these points are given by

$$\mathbf{x}' - \mathbf{x} = \sum_v (\mathbf{x}'_v - \mathbf{x}_v)\, s_v,$$

since $s_v = s(\xi - i, \eta - j)$ is constant. On the straight lines joining the vertices, Eq. (1) is a linear function of either ξ or η. Therefore, points lying on the zone boundaries in the original mesh lie on corresponding zone boundaries in the displaced mesh.

We denote the interpolation function in physical coordinates corresponding to s by

$$S(\mathbf{x}(\xi, \eta)) = s(\xi - i, \eta - j)$$

from which it follows that $\int dV\, S = J$, where $J \equiv \partial(x, y)/\partial(\xi, \eta)$. Since we choose to restrict s to functions whose support is independent of (ξ, η) and depends only on $|\xi - i|$, $|\eta - j|$, the support of S is then a function of $\mathbf{x}$.

The fluid is represented by a distribution of finite-sized particles whose properties are projected onto a grid by interpolation. With each particle, there is associated a label p, a position $\mathbf{x}_p$, a velocity $\mathbf{u}_p$, a mass m_p, an energy e_p, and a color c_p. Because of our choice of interpolation functions, the size of the particle for the purpose of projection is determined by the local grid spacing. The physical length scale is given by the interparticle spacing, not by the grid spacing.

In our examples, we will use either nearest-grid-point (NGP) interpolation or bilinear interpolation to project the particle data onto the grid. NGP is defined by

$$\begin{aligned} &s(\xi - i, \eta - j) = 1, \qquad 0 \leqslant |\xi - i|, |\eta - j| < \tfrac{1}{2}, \\ &s(\xi - i, \eta - j) = 0, \qquad \text{otherwise.} \end{aligned} \tag{3}$$

With this interpolation, all of the particle properties are projected onto the vertex nearest the particle. The bilinear interpolation, given by Eq. (1), also can be written

$$\begin{aligned} &s(\xi - i, \eta - j) = (1 - |\xi - i|)(1 - |\eta - j|), \qquad 0 \leqslant |\xi - i|, |\eta - j| \leqslant 1, \\ &s(\xi - i, \eta - j) = 0, \qquad \text{otherwise.} \end{aligned} \tag{4}$$

This interpolation is a generalization of the area-weighting used in PIC [1]. (To project onto the center of cell (i, j), the arguments of s are replaced by $(i + \frac{1}{2}, j + \frac{1}{2})$.) As illustrated in Fig. 1, the support of a particle overlaps four cells. Higher order interpolation functions, such as those discussed by Monaghan [10] or Hockney and Eastwood [19], generally overlap more than four cells and project onto a larger number of grid points.

One can calculate a density from the particle data. The mass density, for example, is given at $\mathbf{x}$ by

$$\hat{\rho}(\mathbf{x}) = \sum_p m_p \, \delta(\mathbf{x} - \mathbf{x}_p)/J(\mathbf{x}).$$

Because this definition gives a discontinuous density, a more useful (because smoother) density function is obtained by convolving the density above with an interpolation function,

$$\rho(\mathbf{x}) = \sum_p m_p S_p / J(\mathbf{x}), \tag{5}$$

where $S_p = S(\mathbf{x} - \mathbf{x}_p) = s(\xi - \xi_p, \eta - \eta_p)$. (This regularization is usually interpreted as giving the particles finite size. However, it also means that the density at a point is determined by the number of particles in the neighborhood, as defined by the grid.) Other quantities, such as the center of mass or fluid velocity and the specific internal energy, are similarly defined by sums over particles,

$$\mathbf{U}(\mathbf{x}) = \sum_p m_p \mathbf{u}_p S_p / M \tag{6}$$

and

$$I(\mathbf{x}) = \sum_{\mathrm{p}} e_{\mathrm{p}} S_{\mathrm{p}} / M, \tag{7}$$

where

$$M(\mathbf{x}) \equiv \sum_{\mathrm{p}} m_{\mathrm{p}} S_{\mathrm{p}}. \tag{8}$$

To model collisional fluids, we require that the particles at $\mathbf{x}$ move with the fluid velocity, $\mathbf{U}(\mathbf{x})$,

$$d\mathbf{x}_{\mathrm{p}}/dt = \mathbf{U}(\mathbf{x}_{\mathrm{p}}) \tag{9}$$

rather than the particle velocity, $\mathbf{u}_{\mathrm{p}}$. The fluid velocity at any $\mathbf{x}$ is calculated by interpolation from the vertex velocity, $\mathbf{U}_{\mathrm{v}}$, which is evaluated from Eq. (6),

$$\mathbf{U}(\mathbf{x}) = \sum_{\mathrm{v}} \mathbf{U}_{\mathrm{v}} s_{\mathrm{v}}, \tag{10}$$

where $s_{\mathrm{v}} = \mathrm{s}(\xi - i, \eta - j)$. Any difference in the velocity of adjacent particles is revealed only as the grid spacing approaches the interparticle spacing. In that limit, $\mathbf{U}(\mathbf{x}_{\mathrm{p}})$ will tend to $\mathbf{u}_{\mathrm{p}}$. (This prescription is the same as that given by Harlow [1] and McCrory *et al.* [6].)

Consider first the relative motion of the grid and the particles. When the grid and the particles move with the fluid velocity and the interpolation used in Eq. (10) is consistent with the choice of grid (as bilinear interpolation is consistent with quadrilateral zones), the natural coordinates of the particle are constants of the motion. To show this, differentiate Eq. (2) and substitute from Eq. (10) with $s_{\mathrm{vp}} = s(\xi_{\mathrm{p}} - i, \eta_{\mathrm{p}} - j)$,

$$\sum_{\mathrm{v}} \mathbf{x}_{\mathrm{v}} (ds_{\mathrm{vp}}/dt) = 0 = d\mathbf{x}_{\mathrm{p}}/dt - \sum_{\mathrm{v}} \mathbf{U}_{\mathrm{v}} s_{\mathrm{vp}}.$$

Since the right-hand side is zero (cf. Eq. (10)), s_{vp} is constant and, therefore, the natural coordinates of the particle are constant also.

Consider next the acceleration of the particles. Because the fluid is collisional, the acceleration also should be a function of $\mathbf{x}$ only,

$$d\mathbf{u}_{\mathrm{p}}/dt = d\mathbf{U}(\mathbf{x}_{\mathrm{p}})/dt.$$

Differentiating Eq. (10) and noting that $ds_{vp}/dt = 0$, we find,

$$d\mathbf{u}_{\mathrm{p}}/dt = \sum_{\mathrm{v}} d\mathbf{U}_{\mathrm{v}}/dt \, s_{\mathrm{vp}}. \tag{11}$$

Since for numerical stability the particles move during the time interval t to $t + \Delta t$ with a velocity calculated at some intermediate time, $t + \theta \Delta t$, $\frac{1}{2} \leqslant \theta \leqslant 1$, the motion

of the particle involves the acceleration. We calculate the motion by combining Eqs. (10) and (11),

$$d\mathbf{x}_p/dt = \sum_v (\mathbf{U}_v + (d\mathbf{U}_v/dt)\,\theta\Delta t)\, s_{vp}.$$

Only by interpolating the velocity and the acceleration in the same way are we able to solve the equations of motion on the grid without further reference to the particles after we have projected their data onto the grid.

One advantage of this procedure is clear if we consider the alternative described by McCrory *et al.* [6]. In their method, the acceleration at a vertex is assigned to the particles by NGP interpolation, Eq. (3),

$$d\mathbf{u}_p/dt = d\mathbf{U}_v/dt,$$

while the particle displacement is calculated by bilinear interpolation. As a result, their method requires two references to the particles to solve the equations of motion rather than one.

The particle equations of motion, Eqs. (10) and (11), give the correct motion of a single particle through the grid. That is, the particle motion is exactly as though there were no grid at all. The particle motion reduces to $d\mathbf{x}_p/dt = \mathbf{u}_p$, and the particle experiences no self force.

Consider a single particle with velocity $\mathbf{u}_p$. Its motion is given by Eq. (10), which, substituting from Eqs. (6) and (8), yields

$$d\mathbf{x}_p/dt = \sum_v m_p \mathbf{u}_p s_{vp}/m_p s_{vp} = \mathbf{u}_p,$$

where $s_{vp} = s(\xi_p - i, \eta_p - j)$. Thus, the particle moves as though the grid were not there.

The drift of many particles exhibits a similar independence of the grid as shown in Fig. 2, where the motion of a cold ring of particles with initial velocity $\mathbf{U}_0$ is depicted. As time progresses, the ring moves through an obviously distorted mesh without apparent relative motion between the particles. (This simple example is also a sensitive test of the amount of dissipation in the formulation. When we require that energy as well as momentum be conserved as described in Section e below, the ring heats and expands as it moves through the mesh. The expansion, which is quite apparent, is due to the conversion of less than 0.5% of the initial kinetic energy to internal energy. This small dissipation can be reduced by using a smaller time step and a value of $\theta < 1$.)

The absence of a self force, one exerted by a particle on itself, requires only that the equations of motion on the grid conserve momentum. Consider once more a single particle on the grid. Since only those vertices where $s_{vp} \neq 0$ will experience

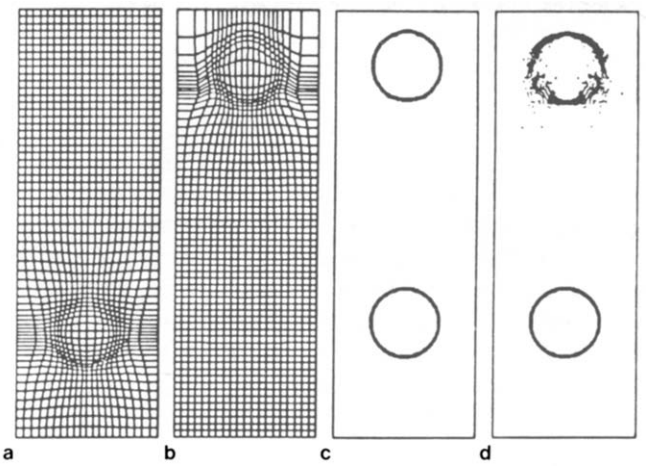

FIG. 2. A ring of initially cold particles moves through an adaptive grid. In (a) and (b), the grids corresponding to 0.1 and 0.7 transit times are displayed. In (c), the particles are plotted at both times without energy conservation, and in (d), with energy conservation imposed. The expansion in (d) is caused by the conversion of 0.4% of the inital kinetic energy into heat.

accelerations, momentum conservation requires only that the acceleration of these vertices satisfy the equation

$$\sum_{v} M_v \, d\mathbf{U}_v/dt = 0.$$

When this equation is satisfied (as it is for a symmetric interpolation function), the force on the single particle must be zero, for

$$m_p \, d\mathbf{u}_p/dt = m_p \sum_{v} d\mathbf{U}_v/dt \, s_{vp} = \sum_{v} M_v \, d\mathbf{U}_v/dt = 0.$$

LeBoeuf *et al.* [7] approach the modeling of a collisional fluid differently. In their algorithm, the particles move with the particle velocity

$$d\mathbf{x}_p/dt = \mathbf{u}_p, \tag{10'}$$

rather than the fluid velocity, $\mathbf{U}(\mathbf{x}_p)$. To reduce multistreaming, a Krook-type drag term is added to the momentum equation to reduce the difference between the particle velocity and the local fluid velocity,

$$d\mathbf{u}_p/dt = d\mathbf{U}(\mathbf{x}_p)/dt + \nu(\mathbf{U}(\mathbf{x}_p) - \mathbf{u}_p), \tag{11'}$$

at a rate determined by the value of the parameter ν. When ν is very large, the difference between particle and fluid velocity will decay rapidly, and their method becomes indistinguishable from that of McCrory *et al.* [6].

b. *The Dynamical Equations*

In FLIP, solutions to initial boundary value problems are generated by marching finite difference approximations to the equations of motion in time. At each time step, the solution is advanced in two phases. In the Lagrangean phase, during which the grid moves with the local fluid velocity, the particles and the grid do not move relative to each other and finite difference equations approximating the continuum flow equations on the grid are solved. Since there are many fewer grid points than particles, this is a very efficient substitution. In the convective transport phase, which we will describe in Section f, the particles are relocated as the grid moves and the fluid information on the grid is regenerated from the particle data.

In deriving the finite difference equations on the grid, we must make them consistent with the kinematic description given above, with the particle representation, and with the conservation of mass, momentum, and energy. Of course, these constraints are conflicting, and we must make certain compromises. In our formulation, the continuity and internal energy equations are consistent with the particle description; the momentum equation is formulated to conserve energy.

In the Lagrangean frame, the mass, momentum, and energy equations are written

$$d\rho/dt + \rho(\nabla \cdot \mathbf{U}) = 0,$$

$$\rho(d\mathbf{U}/dt) + \nabla P - \nabla(\lambda + 2\mu)\nabla \cdot \mathbf{U} + \nabla \times \mu(\nabla \times \mathbf{U}) = 0,$$

and

$$\rho(dI/dt) + P(\nabla \cdot \mathbf{U}) - (\lambda + 2\mu)(\nabla \cdot \mathbf{U})^2 - \mu(\nabla \times \mathbf{U})^2 = 0, \tag{12}$$

where ρ, I, P, and $\mathbf{U}$ are the density, internal energy, pressure, and velocity of the fluid. For constant λ and μ, the viscous terms in the momentum equation reduce to standard form [11]. However, only those viscous terms in the internal energy equation are retained that are explicitly dissipative when the artificial viscosity coefficients are positive.

The continuity and internal energy equations are easily written consistent with the particle representation. The continuity equation is automatically satisfied by the PIC representation described above. Differentiating the definition of the density with respect to time, and noting that $ds_p/dt = 0$, we find

$$d\rho/dt = \sum_p m_p s_p \, d(J^{-1})/dt = -\rho(\nabla \cdot \mathbf{U}),$$

where the divergence and curl of the velocity are given by differentiating Eq. (10),

$$\nabla \cdot \mathbf{U} = \sum_v \mathbf{U}_v \cdot \nabla S_v, \tag{13}$$

where $S_v = S(\mathbf{x} - \mathbf{x}_v)$, and

$$\nabla \times \mathbf{U} = \sum_v \nabla S_v \times \mathbf{U}. \tag{14}$$

The internal energy equation can be written by substituting Eqs. (13) and (14) directly into Eq. (12),

$$\rho(dI/dt) = -\sum_{\nu} \{(P+Q)\nabla S_\nu \cdot \mathbf{U}_\nu - \boldsymbol{\psi} \cdot (\nabla S_\nu \times \mathbf{U}_\nu)\}, \tag{15}$$

where Q is the viscous pressure,

$$Q \equiv -(\lambda + 2\mu) \sum_{\nu} \nabla S_\nu \cdot \mathbf{U}_\nu,$$

and $\boldsymbol{\psi}$ is the viscous shearing stress,

$$\boldsymbol{\psi} \equiv \mu \sum_{\nu} \nabla S_\nu \times \mathbf{U}_\nu.$$

Because we require energy conservation, the momentum equation does not follow directly from the definition of the grid variables. Instead, we begin by calculating the change in kinetic energy, which is given by

$$\rho \mathbf{U} \cdot (d\mathbf{U}/dt) = \rho \sum_{\nu,\nu'} \mathbf{U}_\nu \cdot S_\nu S_{\nu'} (d\mathbf{U}_{\nu'}/dt),$$

where we have substituted from Eq. (10). However, we simplify the solution of the numerical equations by replacing the definition above by

$$\rho \mathbf{U} \cdot (d\mathbf{U}/dt) = \rho \sum_{\nu} S_\nu \mathbf{U}_\nu \cdot (d\mathbf{U}_\nu/dt),$$

which is linear in the interpolation function. There is no evidence that this substitution causes any instabilities, but it does introduce an error in the particle energy, which we discuss below.

The resulting change in the total energy is given by

$$0 = \sum_{\nu} \mathbf{U}_\nu \cdot \{\rho S_\nu (d\mathbf{U}_\nu/dt) - \nabla S_\nu (P+Q) - \nabla S_\nu \times \boldsymbol{\psi}\}.$$

We note that the vertex velocities are independent so that the term inside the brackets must vanish. Since this term is the momentum equation, we are guaranteed conservation of energy if we use

$$\rho S_\nu (d\mathbf{U}_\nu/dt) = \nabla S_\nu (P+Q) + \nabla S_\nu \times \boldsymbol{\psi} \tag{16}$$

to advance the velocities. This equation is easy to difference, for it requires only that we differentiate the interpolation function and it avoids the task of constructing difference equations that satisfy the product rule. This technique originated with Goad [12], who identified it with the principle of virtual work in mechanics. The relationship between Goad's method and the finite element method is discussed by Lascaux [18].

c. *The Difference Equations*

A finite difference approximation to the momentum equation on the grid of quadrilateral cells (in which each cell is labeled c with volume V_c, $J(\mathbf{x}_c) = V_c$, and cell center $\mathbf{x}_c = \sum_v \mathbf{x}_v/4$), is derived by integrating Eq. (16) over the domain

$$M_v(d\mathbf{U}_v/dt) = \int_D dV\{\nabla S_v(P+Q) + \nabla S_v \times \boldsymbol{\psi}\},$$

where the substitution of the vertex mass follows from Eqs. (5) and (8).

To evaluate the integral, we make certain simplifying assumptions. First, we assume P is constant within each cell and equal to

$$P_c = P(\rho(\mathbf{x}_c), I(\mathbf{x}_c)).$$

Second, we replace ∇S_v within each cell by its average value over that cell, $\mathbf{d}_{vc} = \int dV \nabla S_v/V_c$, which we call a geometric coefficient. With these approximations, the dilatation and circulation of the fluid within a cell are computed from linear combinations of the vertex velocities,

$$(\nabla \cdot \mathbf{U})_c = \sum_v \mathbf{d}_{vc} \cdot \mathbf{U}_v$$

and

$$(\nabla \times \mathbf{U})_c = \sum_v \mathbf{d}_{vc} \times \mathbf{U}_v.$$

After integration, there results a sum over the cell indices within the support of S, and the momentum equation is then written

$$0 = M_v(d\mathbf{U}_v/dt) - \left[\sum_c \{P_c + Q_c\}\, \mathbf{d}_{vc} + \mathbf{d}_{vc} \times \boldsymbol{\psi}_c\right] V_c.$$

The number of terms in the summation is determined by the support of S; the further S extends, the larger the number of terms.

The geometric coefficients for bilinear interpolation are easily derived by differentiating Eq. (1) above. With bilinear interpolation, there are just four terms in the summation corresponding to the four cells sharing vertex v. In each cell, there are eight coefficients in Cartesian or twelve in cylindrical geometry.

The use of averaging over the cells to derive difference equations, while simple and direct, is not completely satisfactory in all applications. Although specifying the divergence and curl determines the velocity to within a constant, the difference approximations to them do not since the approximations and the average velocity give only three relations among four variables. Consequently, one has the freedom to construct velocity fields that satisfy the three relations, but correspond to distortions of a kind that lead to the bowtie instability often seen in Lagrangean

calculations. Typically, velocities like these develop as a result of cumulative error rather than instability, and can be suppressed by adding additional dissipation to the momentum equation [13]. In our calculations, the degeneracy is not a problem because the mesh is regenerated externally [4].

d. *Advancing the Equations of Motion in Time*

The equations of motion are marched in time, with time step Δt. Each cycle, the grid variables, ρ, P, $\mathbf{U}$, I, Q, and ψ, are initialized from the particle data. The new time level values of these variables can be calculated from the equations

$$0 = \rho_c^1 \left\{ 1 + \sum_v \mathbf{d}_{vc} \cdot \mathbf{U}_v^\theta \, \Delta t \right\} - \rho_c^0, \tag{17}$$

$$0 = M_v^0 \{ \mathbf{U}_v^1 - \mathbf{U}_v^0 \} - \sum_c \{ \mathbf{d}_{vc} (\tilde{P}_c + Q_c) + \mathbf{d}_{vc} \times \boldsymbol{\psi}_c^0 \} \, V_c \, \Delta t, \tag{18}$$

and

$$0 = I_c^1 - I_c^0 + \left[\sum_v \mathbf{U}_v^\theta \cdot \{ \mathbf{d}_{vc} (\tilde{P}_c + Q_c) + \mathbf{d}_{vc} \times \boldsymbol{\psi}_c^0 \} \, V_c \, \Delta t \right.$$
$$\left. + \sum_v M_{vc} (\theta - \tfrac{1}{2}) (\mathbf{U}_v^1 - \mathbf{U}_v^0)^2 \right] \Big/ M_v, \tag{19}$$

where M_{vc} is that portion of the vertex mass contributed by particles in cell c, and $\mathbf{U}_v^\theta \equiv \theta \mathbf{U}_v^1 + (1-\theta)\, \mathbf{U}_v^0$, with $\frac{1}{2} < \theta < 1$. When $\tilde{P}_c = P_c^0$ the equations are explicit and the time step must satisfy the Courant condition. When $\tilde{P}_c = P_c^\theta$, the equations are semi-implicit and are usually solved assuming $dP/d\rho = \text{const.}$ so that $\mathbf{U}_v^1$ may be eliminated between the continuity and momentum equations to give a single, second-order equation for the pressure. The implicit form of the equations is very useful when there are voids in the domain. When there are voids, and with any interpolation other than NGP, the vertex mass may go to zero more rapidly than the pressure, resulting in very large values of the temperature and the sound speed. When such singularities in the sound speed occur, the implicit equations assume the appropriate form for incompressible flow and the computed acceleration remains in scale automatically [17].

When $\theta > \frac{1}{2}$, the equations are dissipative; the dissipation is quadratic in the acceleration and proportional to $(\theta - \frac{1}{2})$. The last term, which adds the dissipated energy back into the internal energy to give overall energy conservation, is derived by projecting $(\mathbf{U}^1 + \mathbf{U}^0)/2 = \mathbf{U}^\theta - (\theta - \frac{1}{2})(\mathbf{U}^1 - \mathbf{U}^0)$ onto the momentum equation to calculate the change in the kinetic energy.

We remark that we have assumed in deriving these equations that the contribution of the "kinetic" pressure, which is equal to

$$\boldsymbol{\Pi} = \tfrac{1}{2} \sum_p m_p (\mathbf{u}_p - \mathbf{U}(\mathbf{x}_p))(\mathbf{u}_p - \mathbf{U}(\mathbf{x}_p)) \, S_p,$$

is small and can be neglected. There is evidence from numerical experiments that the kinetic pressure is small except in shear layers or other strongly accelerated flow, where it is observed to be 10% of the total pressure.

Boundary conditions are applied to the equations in the usual way. For example, no-slip or free-slip conditions are applied directly to $\mathbf{U}^\theta$ by setting $\mathbf{U}^\theta = 0$ or $\mathbf{U}^\theta \cdot \hat{n} = 0$, respectively.

e. *The Particle Dynamics*

To transfer the results of the solution of the equations of motion from the grid to the particles, we interpolate the accelerations onto the particles,

$$\mathbf{u}_p^1 = \mathbf{u}_p^0 + \sum_v (\mathbf{U}_v^1 - \mathbf{U}_v^0)\, S_{pv}, \tag{20}$$

and advance the particle positions,

$$\mathbf{x}_p^1 = \mathbf{x}_p^0 + \mathbf{U}^\theta(\mathbf{x}_p)\, \Delta t. \tag{21}$$

Note that the boundary conditions do not appear explicitly in these equations. Rather, they are felt through changes in the fluid velocity at the boundary points. For example, at a no-slip wall, $\mathbf{U}^0$ and $\mathbf{U}^1$ may both differ from zero but $\mathbf{U}^\theta$ will satisfy the boundary conditions. As a particle approaches the boundary, its velocity will approach zero as well.

To advance the particle energy and conserve energy, we must normalize changes in the energy of each particle so that the sum of the fractional parts projected onto each particle sum to the whole of the change in the energy on the grid. Within the constraint of having to conserve energy, we are free to normalize in various ways. We choose to normalize changes in the internal energy due to PdV work by the cell internal energy and those due to viscosity by the cell mass. We reason that PdV work ought to change the particle energy according to its contribution to the pressure, but that viscous heating should heat hot and cold particles equally. The change in particle energy is given by

$$\begin{aligned} e_p^1 = e_p^0 \Bigg\{ & 1 - \sum_c \tilde{P}_c (\nabla \cdot \mathbf{U}^\theta)_c s_{pc}\, \Delta t / \rho_c I_c^0 \Bigg\} \\ & - m_p \Bigg\{ \sum_c \left(Q_c (\nabla \cdot \mathbf{U}^\theta) - \boldsymbol{\psi}_c^0 \cdot (\nabla \times \mathbf{U}^\theta) \right) s_{pc}\, \Delta t / \rho_c \Bigg\} \\ & + m_p \sum_v (\mathbf{U}_v^1 - \mathbf{U}_v^0)^2\, s_{vp}(\theta - \tfrac{1}{2}) \\ & + m_p \Bigg\{ \sum_v \left((\mathbf{U}_v^1)^2 - (\mathbf{U}_v^0)^2 \right) s_{vp} - \left((\mathbf{u}_p^1)^2 - (\mathbf{u}_p^0)^2 \right) \Bigg\} \Big/ 2. \end{aligned} \tag{22}$$

The first term in the particle energy equation is the PdV work, the second the viscous heating, the third a correction for the dissipation introduced by the

backwards Euler differencing, and the fourth for an interpolation error described below.

When the accelerations are transferred to the particles, the change in the kinetic energy of the particles is not equal to the change in the kinetic energy calculated on the grid. The grid energy is linear in the interpolation function while the particle energy is quadratic.

If the error is absorbed by the internal energy, total energy is conserved. The result is generally satisfactory because the error is small and the correction positive definite.

Consider the difference between the kinetic energy of the grid and that of the particles. This difference, or error in the kinetic energy transfer, is

$$\varepsilon = -\tfrac{1}{2}\left\{\sum_{\nu} M_{\nu c}((\mathbf{U}_\nu^1)^2 - (\mathbf{U}_\nu^0)^2) - \sum_{p} m_p((\mathbf{u}_p^1)^2 - (\mathbf{u}_p^0)^2)\right\}.$$

This equation can be written in the quadratic form

$$\varepsilon = -\tfrac{1}{2}\sum_{p,\nu,\nu'} \Delta\mathbf{U}_\nu \cdot T_{\nu\nu'}^p \cdot \Delta\mathbf{U}_{\nu'}, \tag{23}$$

where

$$\Delta\mathbf{U}_\nu \equiv \mathbf{U}_\nu^1 - \mathbf{U}_\nu^0$$

and the elements of the transfer matrix are given by

$$T_{\nu\nu'}^p \equiv s_{\nu p}\,\delta_{\nu\nu'} - s_{\nu p}s_{\nu' p}$$

The transfer matrix is symmetric and diagonally dominant since $\sum_{\nu'} T_{\nu\nu'}^p = 0$. Thus, by a theorem proved by Gershgorin [14], the eigenvalues of the matrix are real and positive and the error in the kinetic energy transfer for each particle can be written

$$\varepsilon_p = -\sum_{\nu} \lambda_\nu^p\, \Delta\mathbf{U}_\nu \cdot \Delta\mathbf{U}_\nu < 0,$$

where λ_ν^p are the eigenvalues of the matrix. (For nearest-grid-point interpolation, the transfer matrix and the eigenvalues are zero and the energy error is zero.) Therefore (except for nearest-grid-point interpolation), the error in the kinetic energy transfer is negative definite particle by particle, the correction in the internal energy is positive, and the overall scheme is dissipative and energy conservative.

We repeat the analysis for the algorithms which include a drag term such as those described by McCrory *et al.* [6] and LeBoeuf *et al.* [7] In their algorithms, Eq. (20) is replaced by

$$\mathbf{u}_p^1 = \mathbf{u}_p^0(1-\nu) + \sum_{\nu} (\mathbf{U}_\nu^1 - (1-\nu)\,\mathbf{U}_\nu^0)\, S_{p\nu}, \qquad 0 \leqslant \nu \leqslant 1. \tag{20'}$$

When $v=0$, Eqs. (20) and (20′) are the same. When $v=1$, Eq. (20′) reduces to,

$$\mathbf{u}_p^1 = \sum_v \mathbf{U}_v^1 S_{pv}.$$

We note that the Krook-type term in Eq. (20′) is consistent with momentum conservation,

$$\sum_{v,p} m_p(\mathbf{U}_v^0 S_{pv} - \mathbf{u}_p^0) = 0,$$

but not with energy conservation. We can demonstrate this by repeating the steps leading to Eq. (23) with the result

$$\varepsilon = -\tfrac{1}{2} \sum_{p,v,v'} \Delta\mathbf{U}_v \cdot T_{vv'}^p (\Delta\mathbf{U}_{v'} + 2v\mathbf{U}_v^0). \tag{23′}$$

The error in Eq. (23′) is no longer in quadratic form. Thus, except for NGP interpolation for which the error is zero, the sign of the error is indeterminate. The error is also $O(\Delta t)$ rather than $O(\Delta t^2)$ as in Eq. (23), and therefore the total error in a given calculation will not decrease with the time step. For these reasons, it appears that algorithms using drag terms to reduce multistreaming cannot be extended successfully to bilinear interpolation.

f. *Convection*

At the end of the Lagrangean phase of the calculation we regenerate the grid, either to restore it to its original configuration or to adapt to the solution of the dynamical equations. The particles remain stationary, but we must relocate them on the displaced grid.

The relocation problem may be stated quite simply. Our information about the location of the particle relative to the grid is inaccurate as soon as the grid is moved, because the information is local. Thus, we must develop an algorithm to relocate a point $\mathbf{x}_p$ within a cell on a mesh of quadrilaterals. When the mesh is rectilinear, as it is in most PIC methods, the problem is so easily solved as to be no problem at all. When the mesh is nonrectilinear, some sort of search is necessary because the geometry changes from cell to cell.

Where the natural coordinates are defined by Eq. (1), $\mathbf{x}_p$ is in a cell with index (i, j) iff

$$i \leqslant \xi \leqslant i+1, \qquad j \leqslant \eta \leqslant j+1. \tag{24}$$

When $\mathbf{x}_p$ is in another cell, other values of (ξ, η) are obtained from the transformation. Usually, these are approximately correct for a neighboring cell so that a systematic search with intermediate solutions giving the next indices to test will succeed in a number of steps approximately equal to $((\xi-\xi^0)^2+(\eta-\eta^0)^2)^{1/2}$, where (ξ^0, η^0) is the initial guess.

In a normal fluid flow calculation, the relative motion between the grid and the particles is limited. Time step constraints imposed by the algorithm for the dynamical equations limit the particle displacement to one cell width per time step and the regeneration of the grid from one time step to the next will usually result in a new grid that is very much like the old one. Under these conditions, we may expect many particles to remain in their original cell each time step, and most of the rest to move only one cell away. Therefore, the natural coordinates from the previous cycle are a good starting point for a search, and the algorithm should succeed after one iteration for almost all the particles. This makes the search very efficient. Of course, the algorithm must cope with exceptional cases as well because we do not want to limit the time step by particle motion if it should be useful to move the mesh more aggressively.

Even when it is known that $\mathbf{x}_p$ is not in cell (i, j), Eq. (1) can be solved for (ξ', η') as follows: Define $\xi'' = \xi' - \frac{1}{2}$, $\eta'' = \eta' - \frac{1}{2}$, and rewrite Eq. (1) as a Taylor series expansion about the cell center,

$$\mathbf{x}_p - \mathbf{x}_c = \mathbf{x}_\xi \xi'' + \mathbf{x}_\eta \eta'' + \mathbf{x}_{\xi\eta} \xi'' \eta'',$$

where $\mathbf{x}_\xi = \partial \mathbf{x}/\partial \xi$, $\mathbf{x}_\eta = \partial \mathbf{x}/\partial \eta$, $\mathbf{x}_{\xi\eta} = \partial^2 \mathbf{x}/\partial \xi\, \partial \eta$, and $\mathbf{x}_c = \sum_v \mathbf{x}_v/4$. The derivatives are easily evaluated by differentiating Eq. (1). Solving the component equations for ξ'' yields the quadratic equation

$$\xi'' = (-b + \sqrt{(b^2 - 4ac)})/2a, \tag{25}$$

where

$$a = x_\xi y_{\xi\eta} - y_\xi x_{\xi\eta},$$
$$b = x_\xi y_\eta - x_\eta y_\xi + x_{\xi\eta}(y_p - y_c) - y_{\xi\eta}(x_p - x_c),$$
$$c = x_\eta(y_p - y_c) - y_\eta(x_p - x_c).$$

In choosing the sign, we assume that the cells are only moderately skew, so that $4ac > b^2$, and convex, so that $b > 0$.

When the particles change cells, new values of (i, j) can be calculated from the natural coordinates. Substituting these into Eq. (1) will not satisfy the equations, but the indices can be corrected so that a recalculation of the natural coordinates will reproduce $\mathbf{x}_p$ and complete the search.

In exceptional cases, as when the discriminant is negative so that the transform is complex, or when b is negative so that the extrapolated mapping is no longer convex, the transform misleads the search. For example, when the grid is very skew, the values of the natural coordinates can become complex even though all the cells are convex. When the particles are displaced from their original cell, extrapolating the mapping beyond a cell is not an accurate approximation when the geometry changes radically from one cell to the next. In such cases, a geometric test sup-

plements the coordinate transformation and directs the search, which then shifts one cell at a time.

It is convenient to store the natural coordinates of the particles, since they are used often each computation cycle in the interpolation between grid and particles. The physical coordinates are easily calculated from Eq. (1).

We have also found a linked list data structure to be useful [19]. We update the linked lists as particles move from cell to cell, and generate short cell-ordered lists when vectorization is sufficiently rewarding to warrant the cost. For example, there is a significant number of computations to be done in the solution of the dynamical equations, all of which can be vectorized on the Cray computer. For this portion of the calculation, short lists pay for the cost of creating and maintaining them.

g. *Review of the Computation Cycle*

In this section, we review the computational algorithm by listing the steps in a computation cycle in sequence. Beginning with the projection of particle attributes onto the grid, the sequence of steps is:

i. We project the particle data onto the grid by interpolation,

$$\rho_c = \sum_p m_p s_{pc}/V_c, \tag{5}$$

$$M_v \equiv \sum_p m_p s_{pv}, \tag{8}$$

$$\mathbf{U}_v = \sum_p m_p \mathbf{u}_p s_{pv}/M_v, \tag{6}$$

and

$$I_c = \sum_p e_p s_{pc}/\rho_c V_c. \tag{7}$$

ii. We solve the Lagrangean equations of motion, first Eq. (18) for $\mathbf{U}_v^\theta$,

$$0 = M_v^0\{\mathbf{U}_v^\theta - \mathbf{U}_v^0\} - \sum_c \{\mathbf{d}_{vc}(\tilde{P}_c + Q_c) + \mathbf{d}_{vc} \times \boldsymbol{\psi}_c^0\}\, V_c \theta\, \Delta t; \tag{18}$$

second Eq. (17) for ρ_c^1, (but only if the equations are implicit),

$$0 = \rho_c^1 \left\{1 + \sum_v \mathbf{d}_{vc} \cdot \mathbf{U}_v^\theta\, \Delta t\right\} - \rho_c^0; \tag{17}$$

and finally we advance $\mathbf{U}_v$ fully,

$$\mathbf{U}_v^1 = (\mathbf{U}_v^\theta - (1-\theta)\,\mathbf{U}_v^0)/\theta.$$

It is not necessary to solve Eq. (19) for I_c^1.

iii. We solve the particle equations of motion

$$\mathbf{u}_p^1 = \mathbf{u}_p^0 + \sum_v (\mathbf{U}_v^1 - \mathbf{U}_v^0)\, S_{pv}, \tag{20}$$

$$\mathbf{x}_p^1 = \mathbf{x}_p^0 + \mathbf{U}^\theta(\mathbf{x}_p)\, \Delta t, \tag{21}$$

$$\begin{aligned} e_p^1 = e_p^0 \left\{ 1 - \sum_c \tilde{P}_c (\nabla \cdot \mathbf{U}^\theta)_c\, s_{pc}\, \Delta t / \rho_c I_c^0 \right\} \\ - m_p \left\{ \sum_c (Q_c (\nabla \cdot \mathbf{U}^\theta) - \boldsymbol{\psi}_c^0 \cdot (\nabla \times \mathbf{U}^\theta))\, s_{pc}\, \Delta t / \rho_c \right\} \\ + m_p \sum_v (\mathbf{U}_v^1 - \mathbf{U}_v^0)^2\, s_{vp}(\theta - \tfrac{1}{2}) \\ + m_p \left\{ \sum_v ((\mathbf{U}_v^1)^2 - (\mathbf{U}_v^0)^2)\, s_{vp} - ((\mathbf{u}_p^1)^2 - (\mathbf{u}_p^0)^2) \right\} / 2. \end{aligned} \tag{22}$$

iv. We regenerate the grid and solve for new particle natural coordinates

$$\xi'' = (-b + \sqrt{(b^2 - 4ac)})/2a, \tag{25}$$

where

$$\begin{aligned} a &= x_\xi y_{\xi\eta} - y_\xi x_{\xi\eta}, \\ b &= x_\xi y_\eta - x_\eta y_\xi + x_{\xi\eta}(y_p - y_c) - y_{\xi\eta}(x_p - x_c), \\ c &= x_\eta (y_p - y_c) - y_\eta (x_p - x_c). \end{aligned}$$

v. If linked lists are used, they are updated at this time.

2. Numerical Examples

The properties of the adaptive PIC algorithm are illustrated in two numerical examples: flow over a step and the interaction of a shock wave with a thin foil. Using these examples, we examine the accuracy of PIC compared with finite difference methods. We also illustrate the properties of alternative PIC formulations, and explain the use of adaptive zoning in problems with shocks or contact discontinuities.

a. *Flow over a Step*

We consider confined flow over a step on the domain shown in Fig. 3. A similar problem has been studied by Woodward and Colella [15]. In the step flow problem, Mach 3 flow enters a duct with smooth, plane-parallel walls a distance L apart. In the duct, there is placed a step with a rough surface $0.6L$ in from the

orifice. The height of the step is $0.2L$. The domain of the calculation extends $3L$ from orifice to outflow, where continuative boundary conditions are imposed.

i. *Accuracy: Comparison with Finite Difference Methods.* We evaluate the accuracy of FLIP by comparing results for supersonic flow with particles to those obtained using finite difference methods. The results of many different methods are available from previous studies [15, 16], and comparisons allow us to place PIC accurately within the hierarchy of methods available.

We will first attempt to classify FLIP according to some standard scheme. Woodward and Colella [15] classify the various methods for treating problems with shocks into three categories. The categories, listed in order of increasing accuracy, are artificial viscosity, linear hybridization, and Godunov's method. According to our previous description, the PIC method is an artificial viscosity method, because artificial viscosity is used in the Lagrangean phase of the calculation. However, in two-phase Lagrangean–Eulerian methods, [17] the way convection is treated is important. Woodward's BBC code, which also uses artificial viscosity in the Lagrangean phase, obtains improvements in accuracy by using a linear hybrid of first and second order schemes to calculate convection, and by comparison BBC is nearer in accuracy to linear hybrid schemes than to more standard, single phase artificial viscosity methods [15]. Thus the way convection is calculated should determine the classification of the method.

We compare FLIP results with a code developed by Saltzman [16]. His code uses flux-corrected transport to calculate convection in the Eulerian phase of a Lagrangean–Eulerian code. Since the Lagrangean phase of his code uses the same

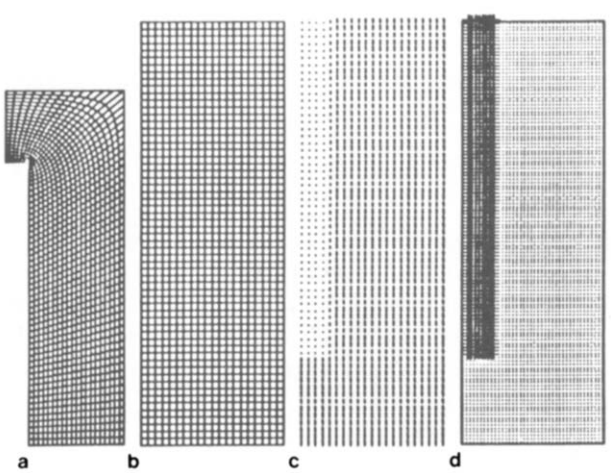

FIG. 3. In (a), the body-fitted grid with 20×60 zones used in the finite difference calculations is plotted. Mach 3 flow enters at the top and exits at the bottom. In (b), (c), and (d) are plotted the uniform grid used in FLIP, the velocity vectors corresponding to the initial flow (which enters from the bottom), and the particles (with asterisks denoting the fixed particles which define the step).

difference equations as used in FLIP, the major differences in the results are due to our using particles to represent the fluid.

There is also a minor difference in the meshes used in the two calculations, as shown in Fig. 3. The mesh for the FLIP calculation has a rectangular boundary that includes the step. The mesh for Saltzman's calculations uses body-fitted coordinates that exclude the step. Because of the small zones at the corner of the step in the body-fitted mesh, Saltzman's calculations require five times as many time steps as does FLIP, even though the stability conditions are similar.

Initially, the flow velocity is constant everywhere. However, as time progresses a bow shock and a sequence of reflected shocks form within the duct. What makes the problem difficult to do is that each reflected shock is weaker than the last, and numerical dissipation or lack of resolution will eventually dominate. Because of the no-slip boundary conditions on the step in our calculations, a boundary layer forms there and partially obstructs the flow after several transit times.

The development of the flow in time as calculated with Saltzman's code is illustrated in Fig. 4, where a sequence of pressure contour plots are shown at intervals of one transit time, $L/|u|$. These results will be the standard for comparison. The mesh for this calculation is divided into 20×60 zones.

In comparing the PIC results with those shown in Fig. 4, we first test the effect of the number of particles per cell. The results are illustrated in Fig. 5, where the pressure contours after three transit times are shown for calculations with 4, 9, 16, and 25 particles per cell. With 4 particles per cell, there is so much "noise" due to the granularity of the representation that the shock structure is difficult to identify. With 9–25 particles per cell, the results are more nearly comparable to the finite difference results. At least, all of the reflected shocks can be identified, and they are in

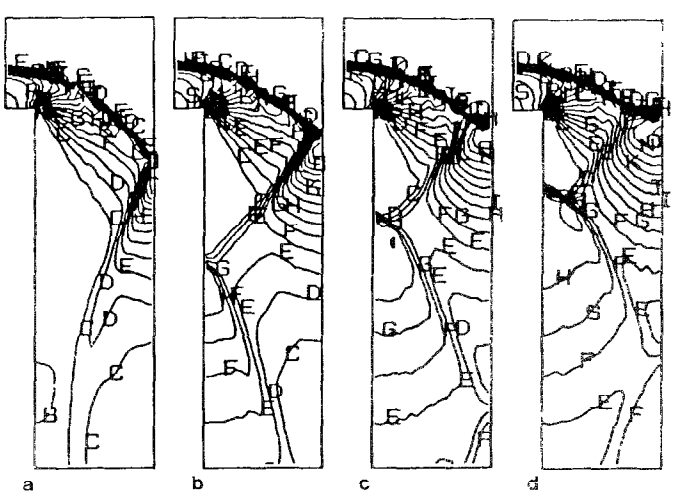

FIG. 4. The results of the finite difference calculation for Mach 3 flow over a step are illustrated by pressure contours at $t = 1$, 2, 3, and 4 transit times in (a–d).

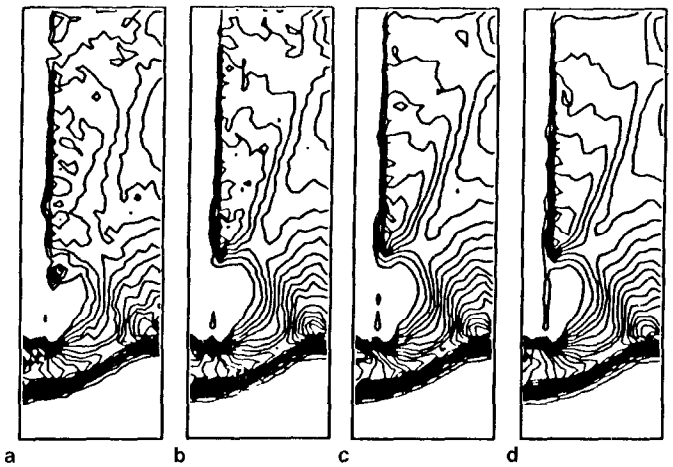

FIG. 5. Pressure contours from FLIP calculations of Mach 3 flow over a step are plotted at $t = 3$ (transit times) with 4, 9, 16, and 25 particles per cell in (a), (b), (c), and (d), respectively.

the correct locations. With as few as 9 particles per cell the major features of the flow appear to be resolved. In Fig. 6, a sequence of pressure contours is shown for the calculation with 25 particles per cell. With 16–25 particles per cell, the particle contribution to the error is comparable to that from the grid, and further increases in the number of particles result in only small, additional increases in accuracy.

From these calculations, we conclude that the particle code can do similar problems as can be done with a finite difference code. The accuracy is less than with finite difference methods. The decrease in accuracy (or, equivalently, the increase in

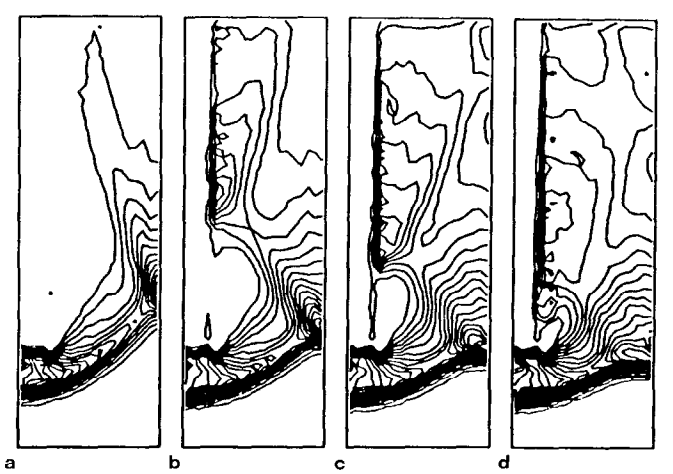

FIG. 6. Pressure contours from a FLIP calculation with 25 particles per cell are plotted at times corresponding to those in Fig. 4.

effort required for comparable accuracy) means that when efficiency is an issue, FLIP should be reserved for problems that are very difficult to do otherwise, such as the shock–foil interaction problem discussed below.

ii. *Accuracy: Comparison with Alternative* PIC *Formulations.* As we have discussed earlier, there are several differences between our PIC formulation and that of McCrory *et al.* [6] and LeBoeuf *et al.* [7]. One difference is their use of a drag term to reduce multistreaming, and the other is the use of NGP interpolation by LeBoeuf *et al.* [7] to calculate the density, internal energy, and velocity on the grid. We use computational examples to test the effect of these differences in two separate experiments.

First, let us consider NGP interpolation. When NGP interpolation rather than bilinear interpolation is used, the thermodynamic variables change discontinuously when a particle moves from one cell to another. This is similar to the original PIC, where mass weighted fractions of the donor cell energy and momentum are carried by particles to the acceptor cell. One might expect large fluctuations to develop driven by fluctuations in the pressure. However, Harlow argues that cell crossings themselves are dissipative and introduce an "effective viscosity" which increases as the number of particles moving from one cell to another in a time step [1]. Conversely, particle "ringing" in stagnated flow, which is often observed in PIC calculations, is due to the absence of dissipation.

If this argument is applicable, we ought to see much less "ringing" in FLIP at all flow speeds even though cell crossings in FLIP are not dissipative because bilinear interpolation reduces the fluctuations that drive ringing. In Figs. 7b and d there is very little evidence of ringing with bilinear interpolation at Mach 3 or at Mach 1.

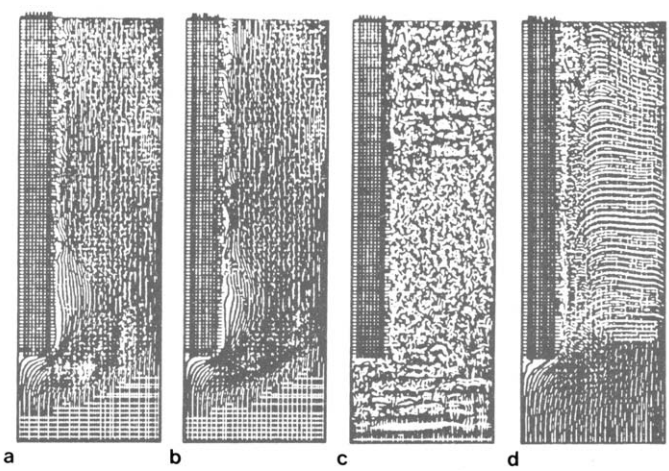

FIG. 7. Particles are plotted at $t = 3$ transit times for Mach 3 and Mach 1 flow with nearest-grid-point interpolation in (a) and (c), and for Mach 3 and Mach 1 flow with bilinear interpolation in (b) and (d). Note the particle ringing in (c).

There is little evidence of any deviation from an orderly particle distribution, except in the boundary layer where one should see it. On the other hand, if we replace the bilinear interpolation defining the density and energy by NGP interpolation, the results, also shown in Fig. 7, indicate a strong particle ringing at Mach 1 (Fig. 7c) but not at Mach 3 (Fig. 7a). As time progresses, the "ringing" increases in amplitude, but not exponentially.

From these and other results not shown, we conclude that bilinear interpolation is better to use for flows where the thermodynamic pressure is significant compared with the kinetic pressure. In our example, NGP gives results with some ringing for transonic flows, and even stronger ringing for low speed flows. Even bilinear interpolation gives results with ringing as the Mach number is decreased further. However, with implicit differencing in time and bilinear interpolation the ringing is suppressed at all Mach numbers.

Next, we replace Eq. (20) by Eq. (20′) and repeat the Mach 3 flow with 16 particles per cell and $\nu = 0.1$. (This increases the similarity of FLIP to the method described by McCrory *et al.* [6].) Because we are requiring energy conservation, a larger value of ν results in an instability when the correction to the particle internal energy is negative. The results of the calculations are shown in Fig. 8 after three transit times. Compared with the contours in Fig. 6, the ones in Fig. 8 do not show clearly the second reflected shock, much less the third. We conclude that a Krook-type drag term when used with bilinear interpolation to reduce multistreaming will cause a significant loss of accuracy.

As we noted earlier, when NGP interpolation is used Krook-type drag is not dissipative. However, the "ringing" we observe with NGP is a more apparent problem

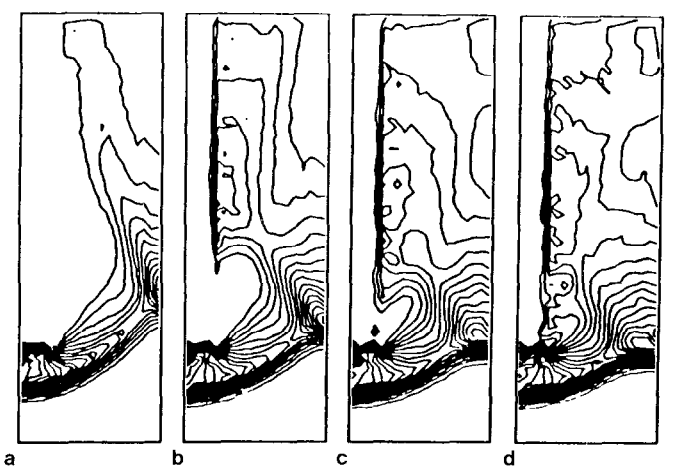

FIG. 8. The pressure contours for a calculation of Mach 3 flow over a step with a Krook-type drag term to reduce multistreaming are plotted. Compared with the results shown in Fig. 6 without drag, the solution with additional dissipation has much less structure.

than multistreaming. It seems also to be present in the results of LeBoeuf *et al.* [7] in their calculation of the Kelvin–Helmholtz instability. In their particle plots, there are striations and gaps in the particle distribution.

b. *Adaptive Zoning*

We now consider the use of adaptive zoning with particles. By comparing the results in a calculation of flow over a step with those for shock interaction with a thin foil, we show that adaptive PIC is most useful in those problems for which the particle representation is most applicable. There, its use enables one to employ very crude meshes, yet still capture the essential features of the flow.

First, we show the results of an adaptively zoned calculation of flow over a step in Figs. 9 and 10 with 20×60 zones. The adaptive zoning is generated by minimizing the functional

$$F = \iint d\xi\, d\eta [\{\nabla\xi^2 + \nabla\eta^2\} + \{wJ^2\}] \tag{26}$$

as described in Ref. [8]. In the calculation shown, the weight function w is given by

$$w = d^2 N_c^2 \{\nabla P/P_0\}^2. \tag{27}$$

where N_c is the number of particles in cell c, d is the distance from the corner of the step, and P_0 is the inflow pressure. Because the gradients in the pressure are less meaningful in cells where there are fewer particles, the weight function is made to scale with N_c. The factor d prevents very small cells from forming in the stagnation region and also amplifies the gradients near the outflow boundary where the shocks are weaker.

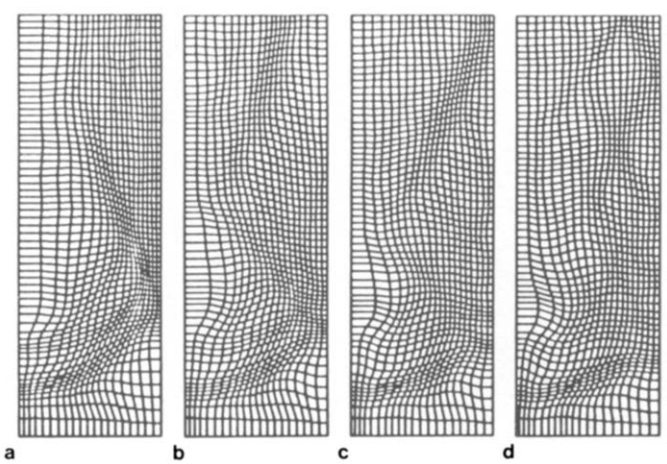

FIG 9. An adapted grid for Mach 3 flow over a step is plotted. The grid is adapted to resolve gradients in the pressure.

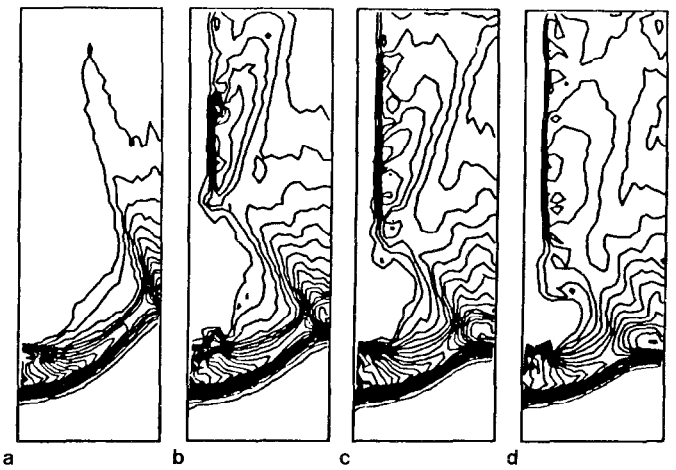

FIG. 10. The pressure contours for a calculation on an adaptive grid are plotted at times corresponding to those in Fig. 4.

The refinements in the grid spacing, shown in Fig. 9, are correlated with the pressure gradients, which can be deduced from the pressure contours shown in Fig. 10. The results of the adaptively zoned calculation are somewhat improved over those in Fig. 6, although the pressures have been smoothed so that fluctuations do not drive the adaptive grid. The improvement obtained with adaptive zoning is not as dramatic as that reported by Saltzman [16]. To obtain that kind of improvement, one must make the volume weighting term more dominant in

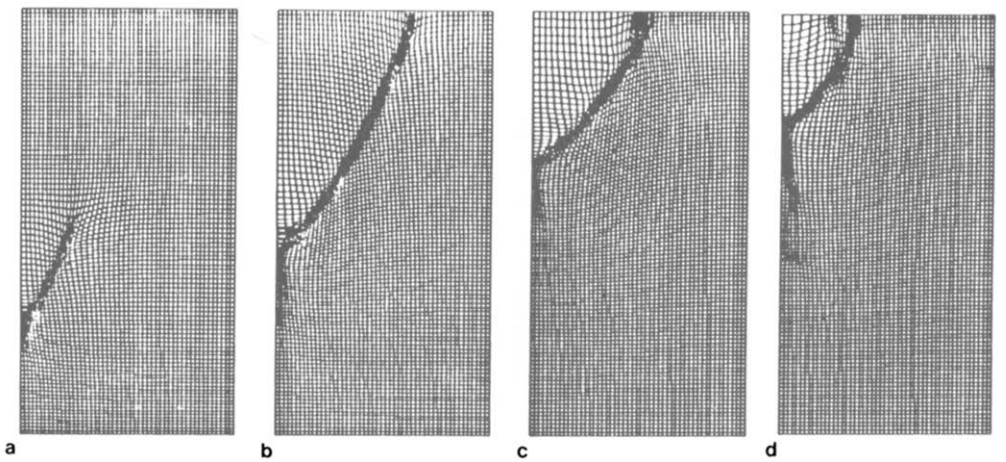

FIG. 11. The 50×100 grid for a calculation of the interaction of a shock with a thin foil is plotted at $t = 10$, 20, 30, and 35 problem time units in (a–d). The right, top, and left boundaries are free slip, rigid walls, and the bottom boundary is a prescribed inflow boundary. The passage of the shock along the foil causes a progressive refinement of the zones.

Eq. (26). However, with a particle code, the smallest zone must contain at least one particle and this places an upper limit on the adaptivity of the mesh. For example, with 9 particles per average cell, the linear dimensions of the smallest zone can be no smaller than one-third the average value.

We next consider an obliquely incident, plane shock which is driven into a thin foil. Initially, the foil density is 200 and the fill gas density is 0.1 in problem units. The shock is amplified by reflection from a rigid wall, and interacts a second time with the foil and compresses it.

A very fine resolution, adaptive grid with 50×100 zones is shown in Fig. 11 at $t = 10$, 20, 30, and 35 units of problem time. The results obtained with this grid at the cost of an hour of computing time will be our reference for comparison. The left, top, and right boundaries are rigid walls; on the bottom boundary, fluid is injected at supersonic speed. Initially, the domain is filled with cold gas at low density.

The inflow of cold gas with density 0.4 in problem units at the bottom boundary drives a plane shock upward. It begins to interact with the foil, first crushing the thin end of the wedge and driving it upward. As the shock begins to interact, the mesh adapts to the pressure gradients as prescribed by Eq. (27). Since there are many more particles in the foil than in the fill gas, as shown in Fig. 13, the zones tend to be more concentrated in the foil than in the fill gas for equal values of the pressure gradient scale length. In the grids depicted in Figs. 11a and b, the passage of the shock is remarked by the grid; its response to the changing pressure gradients is localized. In Fig. 11d, the zoning reflects the increase in the complexity of the solution in the interior of the foil.

In Fig. 12, density contours trace the history of the foil. The passage of the shock over the foil compresses and accelerates the foil upward beginning at the apex of

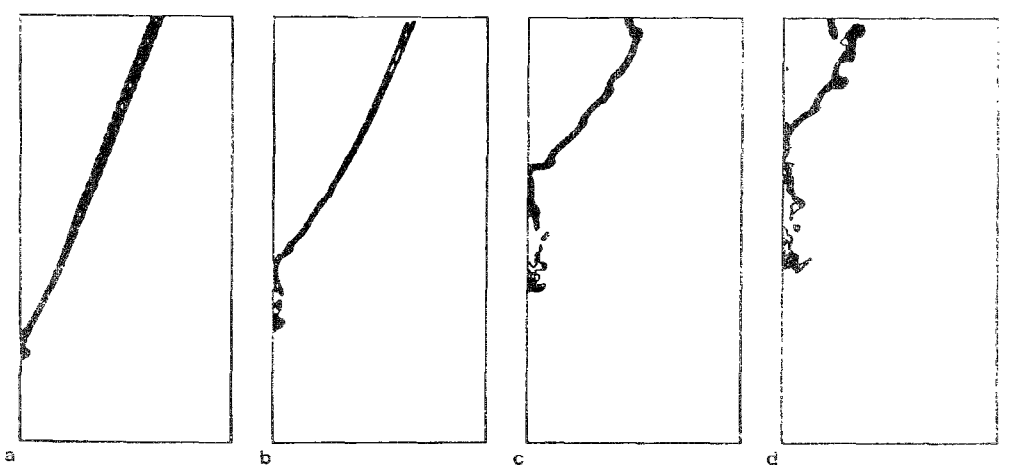

FIG. 12. The density contours at times corresponding to those in Fig. 11 are plotted. Note the rippling of the foil in (c) and the folding at the top, with separation occurring at the fold in (d).

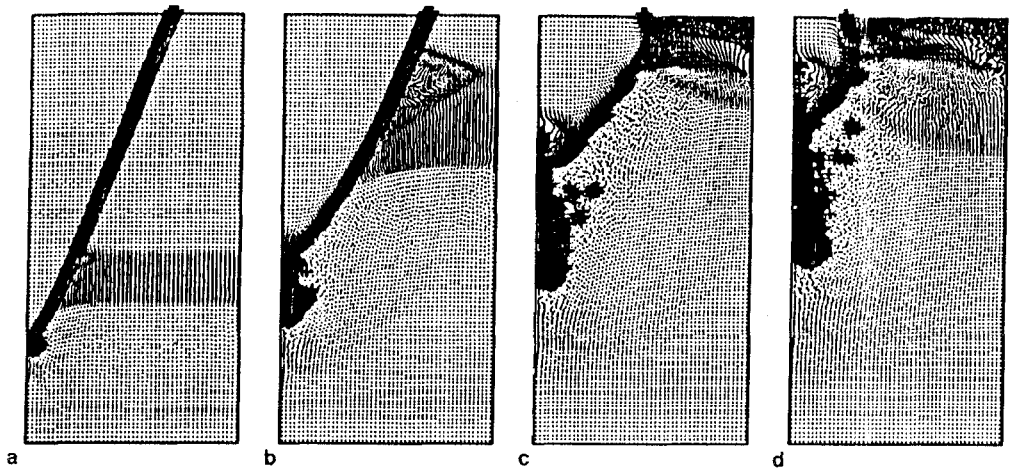

FIG. 13. The particles for the shock–foil problem are plotted at times corresponding to those in Fig. 11. Many details of the flow which are not evident in the density plot are visible in this representation. Note the mixing of foil material from the lower apex with the gas flow, the jetting of foil material into the cavity along the left boundary in (c) and (d), the large amplitude deformation of the foil surface in (d), and the multiple shocks due to reflections from the foil and the wall in (b)–(d).

the foil. Note the narrowing of the high density region as the grid is refined, which occurs because of the decrease in the effective size of the particles. In Fig. 12c at $t = 30$, a number of new features in the flow are apparent. The apex of the foil has been pulled away from the left boundary. Above the apex, the foil is rippled at regular intervals, and, near the top boundary, the foil is folded under. In the grid at the corresponding time, shown in Fig. 11c, one can see concentrations of the zones at the ripples, indicating relatively stronger pressure gradients at these points. In Fig. 12d, the ripples have become nodules, and the section of the foil above the fold has separated.

The rippling of the foil is the most significant feature of the calculations, as is evident from the particle plots in Fig. 13. In the plots, the foil is represented by asterisks, and the fill gas by dots. The rippling can be explained by an apparent Kelvin–Helmholtz instability driven by shear flow at the surface of the foil. The subsequent amplification of the ripples is due, evidently, to a Rayleigh–Taylor instability driven by the reflected, secondary shock.

With adaptive zoning, some of the more important features of this calculation can be reproduced by calculations on a much cruder grid even when they are lost on a Eulerian grid with the same number of zones. On a Eulerian grid with 15×30 zones, the instability disappears as shown in Fig. 14. With an adaptive grid using the same number of grid points, the instability is evident as shown by the particle plots in Fig. 15. The adaptive grid for the calculation, shown in Fig. 16, is successful in correctly representing the unstable foil dynamics, because the pressure gradients driving the instability are resolved. This is a very encouraging result,

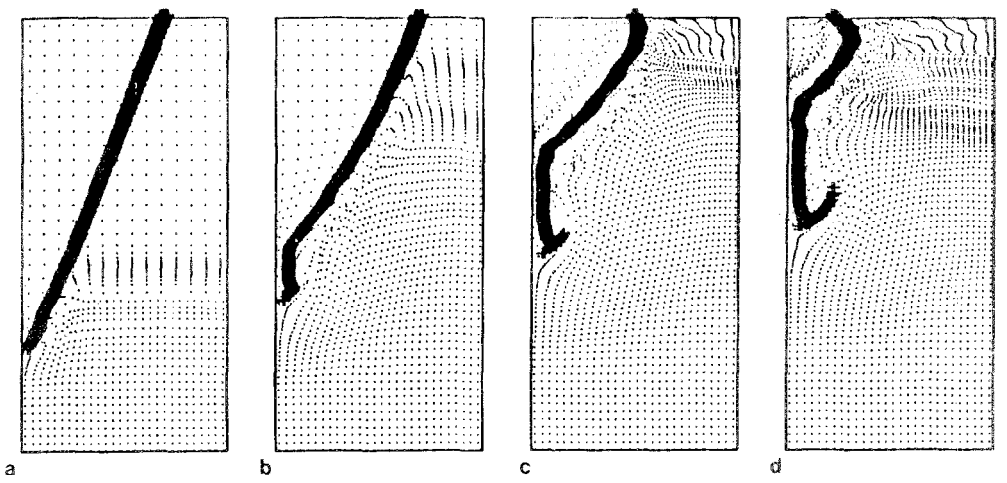

FIG. 14. In the particle plot for the shock–foil calculation on a Eulerian grid with 15 × 30 zones, no evidence of a Rayleigh–Taylor instability is visible. The time sequence is the same in Figs. 11–16.

because the adaptive calculation with the smaller number of zones required only 2–3 min of computing time. With this kind of speed, FLIP calculations can be used as an inexpensive diagnostic tool.

Of course, there is more complexity in the flow than is reproduced by the crudely zoned calculation. Some of this complexity is indicated by the depiction of the flow on the fine grid by the particle plots in Fig. 13. Consider first the shock in the fill gas. In Fig. 13a, the densely packed particles lie below the shock and above the

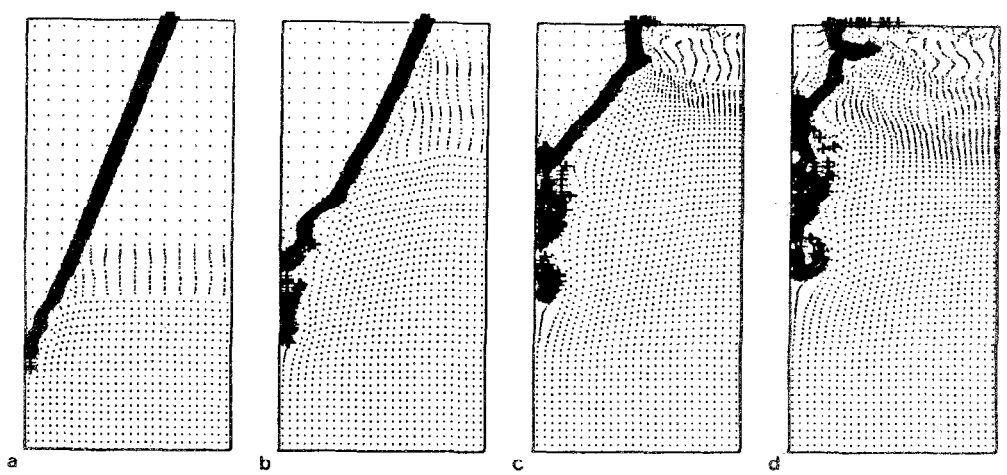

FIG. 15. In the particle plot for the shock–foil calculation on an adaptive grid with 15 × 30 zones, large deformations of the foil by a Rayleigh–Taylor instability and jetting of the foil material as in Fig. 13 are both visible. However, details of the shocks in the fill gas are not visible.

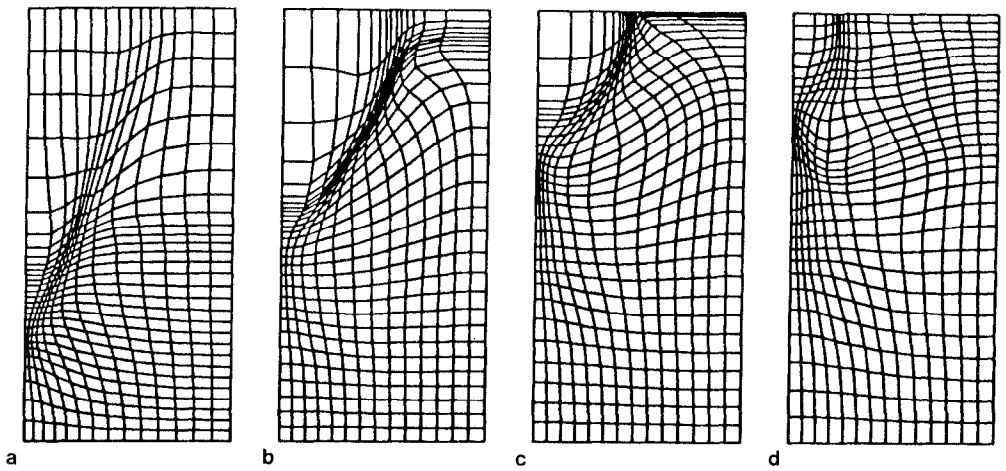

FIG. 16. The 15×30 zone adaptive grid resolves the pressure gradients due to the interaction of the fill gas with the foil.

contact discontinuity between the initial fill and the injected gas. In Fig. 13b, a second shock transition is visible at the boundary between flow parallel to the foil and flow upward. In Fig. 13c, a reflected shock forms at the top boundary, and a shock also forms above the apex of the foil in the gas trapped between the foil and the left wall. The reflected shock is clearly responsible for folding the foil. Also evident in the particle plot is a jet of foil material moving upward from the apex of the foil into the trapped gas which also can be seen in Fig. 15.

3. CONCLUSIONS

Using the FLIP method, one is able to use adaptive zoning with the particle-in-cell representation to model complex geometries with many contact discontinuities. Since particles are easy to initialize and the adaptive zoning is automatic once the criteria for adapting the mesh have been chosen, FLIP is also easy to use.

It is apparent that FLIP is less accurate than finite difference methods. It is also somewhat more expensive. With 9 particles per cell, which seems to be about the minimum number one should use, a third of the computation time each cycle is spent pushing particles, or about 20 μs per particle per cycle on a Cray. On the other hand, FLIP is more accurate than alternative formulations using lower order interpolation or drag terms in the momentum equation.

However, the real value of FLIP is apparent in the shock–foil calculation where finite difference methods are not appropriate. The combination of the particle representation and the adaptive mesh gives the necessary special capabilities of particles and the accuracy of adaptive zoning. When interfaces and discontinuities

requiring a Lagrangean description are to be modeled, the particle representation is flexible and powerful. When there are small, embedded features of the flow to be resolved in the initial data or captured as they develop in the solution, the adaptive mesh gives the needed resolution automatically and efficiently in computer time and storage.

We note again that the separation of the computation cycle into Lagrangean and Eulerian phases is extremely useful. The separation allows us to choose from among many standard finite difference methods for the dynamical equations, including the one we have used in our examples which extends PIC to flow at all speeds using a time-implicit formulation. We also can extend the formulation to more complicated phenomena such as magnetohydrodynamic flow.

Some properties of FLIP are poorly understood and further analysis is needed. For example, the source of "ringing" is incompletely understood, although the combination of bilinear interpolation and implicit differencing in time seems to suppress it.

ACKNOWLEDGMENTS

The authors are grateful to C. Cranfill, D. Forslund, F. Harlow, and J. Saltzman for suggestions, criticism, and encouragement, and to G. Fraley for his collaboration on the shock-foil interaction problem.

REFERENCES

1. F. H. HARLOW, Los Alamos National Laboratory Report No. LA-2301, 1959 (unpublished).
2. F. H. HARLOW, in *Proceedings of the Symposium on Appl. Math. XV, Experimental Arithmetic, High Speed Computations and Mathematics*, 1963 (unpublished).
3. A. A. AMSDEN, Los Alamos National Laboratory Report No. LA-3466, 1966 (unpublished).
4. J. U. BRACKBILL AND J. S. SALTZMAN, *J. Comput. Phys.* **46** 342 (1982).
5. S. NISHIGUCHI AND T. YABE, *J Comput. Phys.* **52** 390 (1983).
6. R. L. MCCRORY, R. L. MORSE, AND K. A. TAGGART, *Nucl. Sci. Eng.* **64** 163 (1977).
7. J. N. LEBOEUF, T. TAJIMA, AND J. M. DAWSON, *J. Comput. Phys* **31**, 379 (1979).
8. J. DAWSON, *Rev. Mod. Phys.* **55** 403 (1983).
9. A. NISHIGUCHI AND T. YABE, *J. Comput. Phys.* **47**, 297 (1982).
10. J. J. MONAGHAN, *SIAM J. Sci. Stat. Comput.* **3**, 422 (1982).
11. G. H. A. COLE, *Fluid Dynamics* (Methuen, New York, 1962), Ch. 5.
12. W. B. GOAD, Los Alamos National Laboratory Report No. LAMS-2365, 1960 (unpublished).
13. R. K.-C. CHAN, *J. Comput. Phys.* **17**, 311 (1975).
14. E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods* (Wiley, New York, 1966), p. 135.
15. P. WOODWARD AND P. COLELLA, *J. Comput. Phys.* **54** 115 (1984).
16. JEFFREY SALTZMAN, Courant Mathematics and Computing Laboratory Report DOE/ER/03077-174, New York University, (unpublished).
17. C. W. HIRT, J. L. COOK, AND T. D. BUTLER, *J. Comput. Phys.* **5** 103 (1970).
18. P. LASCAUX, Center Étude Limeil, Villeneuve-St-Georges, France (unpublished).
19. R. W. HOCKNEY AND J. W. EASTWOOD, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).