Computer Physics Communications 48 (1988) 25–38
North-Holland, Amsterdam

FLIP: A LOW-DISSIPATION, PARTICLE-IN-CELL METHOD FOR FLUID FLOW

J.U. BRACKBILL, D.B. KOTHE and H.M. RUPPEL
Los Alamos National Laboratory, Los Alamos, NM 87545, USA

The FLIP (Fluid-Implicit-Particle) method uses fully Lagrangian particles to eliminate convective transport, the largest source of computational diffusion in calculations of fluid flow. FLIP is an adaptation to fluids of the implicit moment method for simulating plasmas, in which particles carry everything necessary to describe the fluid. Using the particle data, Lagrangian moment equations are solved on a grid. The solutions are then used to advance the particle variables from time step to time step. An adaptive grid and implicit time differencing extend the method to multiple time and space scale flows. Aspects of FLIP's properties are illustrated by modeling of a confined eddy, a Rayleigh-Taylor, an unstable subsonic stream, and a supersonic jet. The results demonstrate FLIP's instability applicability to hydrodynamic stability problems where low dissipation is crucial to correct modeling.

1. Introduction

The particle-in-cell method (PIC) was once the only method able to describe highly distorted flow in two or three dimensions, for which it received considerable attention [1]. With time, its faults became evident. PIC is noisy, and it has more numerical viscosity and heat conduction than are acceptable by today's standards [2]. Were it not for its usefulness in simulating plasmas [3], PIC would be mainly of historical interest.

The problems with PIC are clearly defined. To make PIC as accurate as competing methods, its viscosity and heat conduction must be reduced. The solutions follow one of two different strategies for reducing numerical diffusion, improved "classical" formulations [2,4] and "full-particle" PIC [5–8]. The "classical" PIC method is a partially Lagrangian description of the fluid. To each particle is attributed a mass and a position. The "full-particle" PIC is a fully Lagrangian description of the fluid. To each particle is attributed all of the properties of the fluid, including momentum and energy.

"Classical" PIC is able to resolve contact discontinuities because of its Lagrangian representation of the mass. As the particles stream from cell to cell during a calculation, they transport mass from cell to cell without diffusion. However, the mass weighted transport of momentum and energy, which is usually used, is diffusive. The assignment of grid quantities to the particles and back causes diffusion in momentum and energy.

"Full-particle" methods like FLIP [8] and PAL [7] preserve the ability of "classical" PIC to resolve contact discontinuities, but eliminate the major source of numerical diffusion by using a full Lagrangian representation of the fluid. FLIP also incorporates a variable particle size, adaptive mesh, and implicit solution algorithm, to extend its applicability to problems with multiple time and space scales.

The properties of FLIP when modeling fluid flows in two dimensions are examined here. A comparison of FLIP with "classical" PIC, a development of the correspondence between grid and particle solutions, and an analysis of the ringing instability are reviewed. Several examples, including calculations of a confined eddy, a Rayleigh–Taylor instability, a subsonic stream, a supersonic jet, and an ion beam target are given.

2. The particle-in-cell method

In PIC methods for fluid flows, the Navier–Stokes equations in Lagrangian form are solved on a grid, where they are approximated by

0010-4655/88/$03.50

finite differences and solved. The equations are mass,

$$\mathrm{d}\rho/\mathrm{d}t + \rho\nabla\cdot\boldsymbol{U} = 0 \tag{1}$$

momentum,

$$\rho\ \mathrm{d}\boldsymbol{U}/\mathrm{d}t + \nabla p - \nabla(\lambda + 2\mu)\nabla\cdot\boldsymbol{U} + \nabla\cdot\boldsymbol{T} = 0, \tag{2}$$

$$T_{ij} = \mu\left\{\partial U_i/\partial x_j + \partial U_j/\partial x_i\right\}$$

and energy,

$$\rho\ \mathrm{d}I/\mathrm{d}t + p\nabla\cdot\boldsymbol{U} + \lambda(\nabla\cdot\boldsymbol{U})^2 + \mu\boldsymbol{T}\cdot\boldsymbol{T} = 0, \tag{3}$$

where ρ, $\boldsymbol{U}$, I and p are the density, velocity, specific internal energy and pressure, and μ and λ are the shear and bulk viscosities.

In both "classical" and "full-particle" PIC, convection is modeled by particles moving through the grid. The data on the grid are regenerated from the particle data when there is relative motion between the fluid and the grid. In practice, the regeneration requires projecting the particle data on to the grid points by accumulating a weighted sum of particle contributions at each grid point. The grid density, for example, is calculated from

$$\rho_{\mathrm{c}} = \sum_p \frac{m_p S_{\mathrm{n}}(\boldsymbol{x}_p - \boldsymbol{x}_{\mathrm{c}})}{V_{\mathrm{c}}}, \tag{4}$$

The particle mass, m_{p}, is shared among the cells, c, with centroids, $\boldsymbol{x}_{\mathrm{c}}$, and volumes, V_{c}, according to the value of an assignment function, S_{n}, which depends upon the distance between the particle and the cell center. The assignment function has bounded support so that it is zero for distances greater than some value h (usually the mesh spacing). It is a monotone decreasing function of its argument, and it is normalized to one for any n,

$$\sum_c S(\boldsymbol{x}_c - \boldsymbol{x}_{\mathrm{p}}) = \sum_v S(\boldsymbol{x}_v - \boldsymbol{x}_{\mathrm{p}}) = 1, \tag{5}$$

where S represents any one of the often used nearest-grid-point, linear, or quadratic assignment functions [22], which are described in refs. [3,9,10,22].

"Classical" and "full-particle" PIC differ in whether the particle data is preserved from cycle to cycle. In "classical" PIC, the particles temporarily describe the fluid to model convection. Each cycle, all of the particle data, except mass and position, is replaced by data from the grid solution of the Lagrangian flow equations. The particle velocity in Harlow's PIC [1], for example, is given by

$$\boldsymbol{u}_{\mathrm{p}} = \sum_v \boldsymbol{U}_v S_{\mathrm{n}}(\boldsymbol{x}_v - \boldsymbol{x}_{\mathrm{p}}), \tag{6}$$

where $\boldsymbol{U}_v$ is the fluid velocity at grid vertex v, and S_{n} is the nearest-grid-point (NGP) assignment function. The particles are then displaced using an interpolated velocity from the grid,

$$\frac{\mathrm{d}\boldsymbol{x}_{\mathrm{p}}}{\mathrm{d}t} = \sum_v \boldsymbol{U}_v S_1(\boldsymbol{x}_v - \boldsymbol{x}_{\mathrm{p}}), \tag{7}$$

where S_1 is a linear assignment function. The grid data are then regenerated by projection using eq. (4) and similar equations for the other fluid variables.

In "full-particle" PIC, the particle data describe the fluid completely from cycle to cycle, not just during convection. Each cycle, the particle data are updated from the grid solution, not replaced. In FLIP, for example, the particle acceleration is interpolated from the grid data,

$$\frac{\mathrm{d}\boldsymbol{u}_{\mathrm{p}}}{\mathrm{d}t} = \sum_v \frac{\mathrm{d}\boldsymbol{U}_v}{\mathrm{d}t} S_1(\boldsymbol{x}_v - \boldsymbol{x}_{\mathrm{p}}). \tag{8}$$

This difference between "classical" and "full-particle" PIC is equivalent to the difference between a partially and a fully Lagrangian description of the fluid. The partially Lagrangian description given by the "classical" PIC particles results in significant diffusion of all non-Lagrangian variables.

3. Diffusion in "classical" PIC

In "classical" PIC, grid variables, such as energy and momentum, are transported from cell to cell by the particles as they move through the grid. To transport energy, for example, each particle is assigned a fraction of the cell energy in proportion to its mass before it is moved,

$$e_{\mathrm{p}} = m_{\mathrm{p}} \sum_c \frac{E_c^0}{M_c^0} S\left(\boldsymbol{x}_{\mathrm{p}}^0 - \boldsymbol{x}_c\right), \tag{9}$$

where the cell mass, M_c, is equal to $\rho_c V_c$, and the cell internal energy, E_c, is equal to $M_c I_c$. (The superscript 0 denotes data at the beginning of the time step.) The particles are then displaced to x_p^1, where the superscript 1 denotes data at the end of the time step, and a new cell energy is computed,

$$E_c^1 = \sum_p e_p S\left(x_p^1 - x_c\right)$$

$$= \sum_{c'} \frac{E_{c'}^0}{M_{c'}^0} \sum_p m_p S\left(x_p^1 - x_{c'}\right) S\left(x_p^0 - x_c\right). \quad (10)$$

Even when the particles are not displaced, the new cell energy will not equal the old [2]. Consider the simple example of evenly spaced, equal mass particles on an evenly spaced grid, $x_{c+1} = x_c + \Delta x$, in one dimension. Fourier transforming eq. (10) with $x_p^1 = x_p^0$ yields,

$$E_k^1 = S_k^2 E_k^0. \quad (11)$$

With the commonly used spline interpolation functions [22], for example, the Fourier transform is given by [3],

$$S_n(k) = \left[\frac{\sin(k\,\Delta x/2)}{k\,\Delta x/2}\right]^{n+1}. \quad (12)$$

(Nearest-grid-point (NGP) interpolation corresponds to n equal to 0, linear interpolation to n equal to 1, and quadratic interpolation to n equal to 2.) For k equal to zero, $S_n(k)$ is equal to one and energy is conserved. For all k greater than zero, $S_n(k)$ is positive but less than one. Consequently, interpolating energy from the grid to the particles and back is diffusive. In fact, using higher order splines actually increases the diffusion.

Harlow avoids the zeroth order error in "classical" PIC by using NGP interpolation [1]. With NGP, only those particles residing in cell c contribute to the energy in cell c, and there is no spatial averaging. However, energy is transported discontinuously as particles cross cell boundaries, contributing to noise and diffusion in the results.

Nishiguchi and Yabe [2], and Clark [4] avoid the zeroth order error in "classical" PIC in a different way. They use higher order interpolation only for the changes in momentum and energy due to transport. Their formulations require that stationary particles cause no change in the solution on the grid just as in the original PIC with NGP [1], but with linear or quadratic interpolation as well. With their methods and higher order interpolation, energy and momentum transport is a continuous function of particle displacement and the noise of the original PIC is reduced to a level comparable to ordinary finite difference methods.

In a sense, EPIC [12] is yet another way to improve "classical" PIC. In EPIC the diffusion, which can be calculated from eq. (10), is canceled by an anti-diffusion step (cf. Eastwood, these proceedings).

4. "Full particle" PIC

The success of plasma simulation, which is a "full particle" PIC, has influenced Marder (GAP) [5], Tajima et al. [6], Morse et al. (PAL) [7], and Brackbill and Ruppel (FLIP) [8] to try similar methods for fluid flows. The advantage of the "full-particle" representation is the elimination of numerical viscosity and thermal conduction. By making momentum and energy Lagrangian variables, diffusion of these variables is eliminated just as mass diffusion is eliminated in "classical" PIC by making mass a Lagrangian variable.

"Full particle" PIC obviously costs more, because for each particle a position, velocity, mass and energy must be stored. (The extra cost is not burdensome now, because increased computer memory and hardware-indirect-addressing are now available.) More importantly, since each particle has its own velocity, interpenetration across material interfaces or multistreaming can occur just as in the smoothed particle hydrodynamics (SPH) method [9].

To reduce multistreaming, Tajima et al. [6] calculate the particle displacement from the particle velocity, but introduce a Krook collison operator to diminish the relative velocity between neighboring particles. This affects the results similarly to a viscosity. To eliminate multistreaming, PAL and FLIP calculate the particle displacement, as in "classical" PIC, by interpolating from the grid velocity. This eliminates multistreaming

completely. (PAL introduces some dissipation to suppress the ringing instability [10], which FLIP does not [8].)

5. The FLIP algorithm

The FLIP (Fluid-Implicit Particle) algorithm uses Lagrangian particles to represent the fluid, and a rezonable Lagrangian grid of quadrilateral zones to calculate the interactions among the particles. Variables are assigned to the grid using the standard von Neumann, Richtmyer staggering (see Crowley, these proceedings). In its mixture of a particle representation of the fluid and a grid for the solution of implicit moment equations, it is similar to the implicit moment method in plasma simulation [23].

5.1. The Lagrangian calculation

Using the interpolation functions discussed above, the specific internal energy, vertex mass and center of mass velocity at the grid points are given by,

$$I_c = \sum_p e_p S_2(\boldsymbol{x}_p - \boldsymbol{x}_c)/\rho_c V_c, \tag{13}$$

$$M_v = \sum_p m_p S_1(\boldsymbol{x}_p - \boldsymbol{x}_v), \tag{14}$$

$$\boldsymbol{U}_v = \sum_p m_p \boldsymbol{u}_p S_1(\boldsymbol{x}_p - \boldsymbol{x}_v)/M_v. \tag{15}$$

Linear interpolation is used for vertex-centered variables, such as position, velocity and mass, to be consistent with the quadrilateral cells of the computation grid. Quadratic interpolation is used for cell-centered variables, such as density and energy, to give smoother solutions and to suppress the ringing instability, which is discussed in section 6. By evaluating eqs. (4) and (13)–(15), the data for the solution of Lagrangian equations of motion on the grid are assembled.

The solution of the Lagrangian equations of motion on the grid, eqs. (1)–(3), can be approximated in many ways. In FLIP, the equations are formulated using a finite volume method, which conserves mass, momentum and energy. With backward Euler time differencing, even angular momentum is conserved by the difference equations [11]. The details are given in ref. [8], to which some improvements have since been made. Implicit equations are formulated similarly to the method of Casulli and Greenspan [13] using ICCG (incomplete Cholesky decomposition, conjugate gradient method), with substantial improvements in speed and reliability.

5.2. Consistent particle and grid solutions

FLIP is formulated to given consistency between the solution of the Lagrangian equations of motion on the grid, and the equations of motion of the particles. Consistency is achieved, in part, because the normalization of the assignment function given by eq. (5) is dependent on the grid placement. *Furthermore, the particle equations of motion, eqs. (7) and (8), are consistent with the Lagrangian solution on the grid, because the assignment function, S, is a Lagrangian invariant,*

$$\mathrm{d}S/\mathrm{d}t = 0. \tag{16}$$

To achieve this, S is defined in terms of natural coordinates, (ξ, η) that are calculated by mapping each of FLIP's quadrilateral cells on to a unit square. With this choice, the grid defines the particle size. As the grid and particles move, the value of the assignment function at a point moving with the fluid is constant.

At each vertex, the natural coordinates assume integer values, (i, j) which are obviously constant as the mesh moves. Elsewhere, the mapping between physical and natural coordinates is given by bilinear interpolation, which can be written,

$$\boldsymbol{x}_\mathrm{p} = \sum_{i,j} \boldsymbol{x}_{i,j} S_1(\xi - i, \eta - j)$$

$$= \sum_{i,j} \boldsymbol{x}_{i,j} S_1(\xi - i) S_1(\eta - j). \tag{17}$$

The natural coordinates of a particle, whose physical coordinates are $\boldsymbol{x}_\mathrm{p}$, are defined by inverting this mapping. Because S is a Lagrangian invariant, the particle's natural coordinates are constant during the Lagrangian phase, and the particle displacement, eq. (7), and acceleration, eq.

(8), are calculated by interpolating from the grid values. The correspondence between particle and grid solutions through the Lagrangian phase is perfect for linear functions of the dependent variables.

From eq. (5) and eqs. (13)–(17), one can show that the total mass, momentum, angular momentum and internal energy for the particles and the grid are equal,

$$\sum_p m_p = \sum_v M_v, \tag{18}$$

$$\sum_p m_p \boldsymbol{u}_p = \sum_v M_v \boldsymbol{U}_v, \tag{19}$$

$$\sum_p m_p (\boldsymbol{r}_p \times \boldsymbol{u}_p) = \sum_v m_v (\boldsymbol{r}_v \times \boldsymbol{U}_v), \tag{20}$$

$$\sum_p e_p = \sum_c M_c I_c. \tag{21}$$

However, the kinetic energy is not the same for the particles and the grid. If one defines the particle velocity relative to the mean velocity on the grid by

$$\boldsymbol{v}_p = \boldsymbol{u}_p - \sum_v \boldsymbol{U}_v S_{vp}, \quad S_{vp} = S(\boldsymbol{x}_v - \boldsymbol{x}_p), \tag{22}$$

one can show that the difference between particle and grid kinetic energy is given by

$$\tfrac{1}{2}\left[\sum_p m_p \boldsymbol{u}_p^2 - \sum_v M_v \boldsymbol{U}_v^2\right] = \tfrac{1}{2}\left[\sum_p \left\{ m_p \boldsymbol{v}_p^2 - \sum_{v,v'} \boldsymbol{U}_v \cdot T_{vv'}^p \boldsymbol{U}_{v'} \right\}\right], \tag{23}$$

where the transfer matrix is defined by,

$$T_{vv'}^p = m_p (S_{vp}\delta_{vv'} - S_{vp} S_{v'p}). \tag{24}$$

The transfer matrix is symmetric and diagonally dominant and, therefore, has positive eigenvalues [8]. Thus, both terms on the right side of eq. (24) are positive definite and the difference in the kinetic energy of the particles and the grid can have either sign. However, if one estimates the relative size of the two terms, the second term is the result of the averaging one does to calculate the grid velocities and is never larger. The first term will be relatively large when the velocity variation among the particles is poorly represented by a linear interpolation function. In this sense, it is a measure of how accurately the grid resolves the flow gradients. It is this term that is described as a kinetic pressure in ref. [8]. There it was noted that it is large where large changes in the flow direction or velocity occur.

The variance of the particle velocity about the mean, the first term on the right side of eq. (23), does not change because of accelerations calculated on the grid, which have no effect on the relative particle velocity,

$$\frac{\mathrm{d}\boldsymbol{v}_p}{\mathrm{d}t} = \frac{\mathrm{d}\boldsymbol{u}_p}{\mathrm{d}t} - \sum_v \frac{\mathrm{d}\boldsymbol{U}_v}{\mathrm{d}t} S_{vp} = 0. \tag{25}$$

Solving the equations of motion on the grid affects only those length scales that the grid resolves. The sub-grid scale information that the particles carry is unaffected, and can be retrieved by refining the zoning.

As is required for nonlinear stability, the FLIP formulation is dissipative. Where the change in velocity at a vertex during a time step is $\Delta \boldsymbol{U}_v$, the change in the particle kinetic energy is always different from the change in the grid kinetic energy by the amount

$$\epsilon_p = -\tfrac{1}{2}\left[\sum_{vv'} \Delta \boldsymbol{U}_v \cdot T_{vv'}^p \, \Delta \boldsymbol{U}_{v'}\right]. \tag{26}$$

This error is negative definite, particle by particle, and order Δt^2. When the loss in kinetic energy is added to the particle energy for each particle, energy is conserved exactly while maintaining positive particle internal energy.

5.3. Adaptive zoning

The final step in the computation cycle requires defining the grid for the next cycle. During this step, the particles are stationary in physical space, but their natural coordinates must be recomputed.

The grid may be completely redefined from one cycle to the next. It is only required that the grid map on to a logical rectangle. In FLIP, the number of grid points is fixed, but the grid points may be moved about as needed. Presently, one may choose to use an Eulerian, fixed grid, a Lagrangian

grid (for a Lagrangian calculation, the particles are superfluous), or an adaptive grid. The adaptive grid is generated by solving a variational problem, in which one minimizes a functional. The functional may be complex allowing one to control grid smoothness, orthogonality, spacing and orientation [24,25]; or it may be simple, giving control only over spacing, such as the functional,

$$I = \int_V \frac{\left[\nabla\xi^2 + \nabla\eta^2\right]}{w} \mathrm{d}V. \tag{27}$$

To minimize I, one must solve Euler equations. In the solution, the weight function, w, controls the mesh spacing. Where w is larger, the grid points are closer together. If w is an appropriate function of the data, the grid will adapt to provide increased resolution where it is needed by moving grid points. For example, if w is proportional to the current density in a calculation of magnetohydrodynamic flow, the grid will cluster to resolve steep gradients in the magnetic field intensity [27].

Adaptive gridding can be useful in PIC calcula-

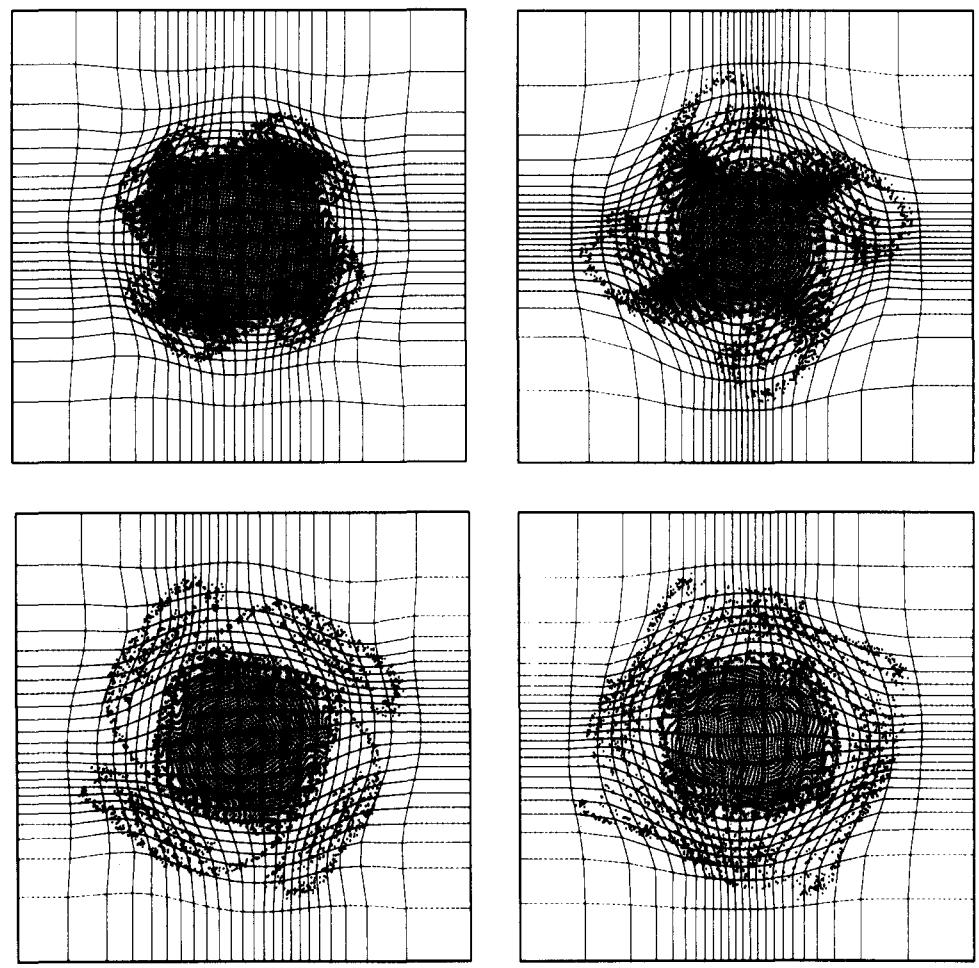


Fig. 1. The adaptive grid plot is superimposed on the particles representing a confined eddy. The eddy, which is rotating counterclockwise, advances one-half rotation period each frame. By the lower-right frame, it has rotated twice. A Kelvin–Helmholtz instability causes secondary eddies to form, which move and coalesce. In these, the higher kinetic pressure causes the grid points to cluster.

tions. For example, when w is proportional to the number of particles per zone, where the particles are sparse the zones will be larger. This extends the range of densities that can be represented by making the particle size larger in regions of low density. When w is proportional to the kinetic pressure, eq. (23), the resolution is automatically increased where the grid does not resolve the flow gradients. The results of this prescription are shown in fig. 1, where the particles in a confined, low-speed eddy are shown superimposed on the grid at intervals of one-half eddy rotation. The calculation is performed on a 24×24 zone grid with thirty-six particles per zone. This eddy problem, which is described in more detail in ref. [11], is subject to a Kelvin–Helmholtz instability. The instability causes secondary eddies to form.

The resolution required to resolve the flow changes in time, and, consequently, the grid changes from frame to frame. The grid is initially clustered in the boundary layer between the eddy and the background fluid. Later, the grid is clustered in the secondary eddies, where the gradients in the velocity become steep and the kinetic pressure becomes relatively large. (The kinetic pressure is less than one per cent of the fluid pressure.) The corresponding results on an Eulerian grid with the same number of zones are different, but the results with the same number of particles on a mesh with four times as many cells (48×48 zones) differ only in detail from those in fig. 1. The instability growth rate and resulting large scale structures are similar. The principal difference is that the cost of the 48×48 zone calculation is twice that for the adaptively zoned calculation. (The time step for the calculation is the same, as allowed by the implicit formulation, but the 48×48 zone calculation costs twice as much with four times as many zones, since the grid solutions require half the total computation time.) In three dimensions, where an adaptive grid would require a tenth as many zones as a fixed grid calculation, the savings would be even greater.

There are constraints on adaptivity in PIC calculations. In low-speed-flow problems, the number of particles per zone must be sufficient to give smoothly varying pressures and densities, even in the smallest zones. As a result, one must use more particles per zone, and the adaptive mesh cannot give the large gains in accuracy observed with finite difference methods [24].

Adaptive gridding has been useful in resolving material interfaces and in responding to changes in scale that may occur when shocks develop. Examples of adaptively zoned FLIP calculations are given in refs. [8,14].

6. The ringing instability

With four to sixteen particles per cell in a typical PIC calculation, there are many more degrees of freedom for the particles than there are for the grid. Thus, the solution of the particle equations of motion is underdetermined by the solution of finite difference equations on the grid. The many different particle solutions which look the same when projected on to the grid are called aliases. The aliases interact nonlinearly to drive the well-known finite grid instability in plasma simulation [3] and the ringing instability in PIC fluid calculations [10].

While the assignment function for the vertex centered quantities, such as position and velocity, is determined by the shape of the computation cells, the choice of assignment function for the cell-centered quantities, such as density and energy, is unconstrained. Thus, one may use higher order assignment functions to reduce the growth rate for the ringing instability [10]. Quadratic interpolation is currently used in FLIP for cell-centered variables. There is a disadvantage, however, because the ability to resolve gradients is diminished using higher order interpolation [9].

Dispersion analysis of the ringing instability in PIC for small time steps, uniform grid spacing, Δx, and uniform flow, U_0, yields the dispersion relation,

$$1 - \sum_{q=-\infty}^{\infty} \frac{S_1(k_q) S_2(k_q) k_q^2 a^2}{\left[\omega - k_q U_0\right]^2} = 0, \tag{28}$$

where a is the sound speed, ω the frequency, k the wave number, and k_q the alias wave number,

$$k_q = k + q(2\pi/\Delta x). \tag{29}$$

The imaginary frequency, labelled gamma in fig. 2, corresponds to exponential growth for Mach numbers, M equal to U_0/a, less than 0.4. The minimum growth time is about ten oscillation periods (With nearest grid point interpolation replacing S_1 and S_2 in eq. (28), the maximum unstable M is one, and the minimum growth time little more than one oscillation period.)

When the total energy is a constant of the motion, the ringing instability is bounded nonlinearly in PIC fluid calculations to relatively low amplitudes. The only evidence of the instability is a disordering of the ordinarily regular patterns of particles, with a consequent spatial modulation of the density or energy on the grid. In fig. 3, the particles representing the dense fluid in a calculation of the Rayleigh–Taylor instability [14] are plotted. With M equal to 0.001, the particles clump into patterns that are characteristic of the ringing instability. With quadratic assignment and implicit differencing, the effect of the instability is usually limited to unimportant effects like the one shown. Including the effect of a very large time step in the dispersion analysis has not been done yet for FLIP, but large time steps are observed to be very effective in suppressing the instability if other time step constraints allow them.

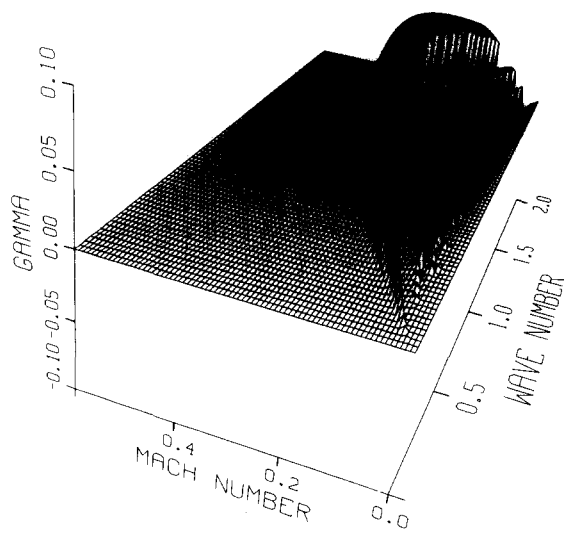


Fig. 2. The ringing instability in FLIP occurs for flow Mach numbers less than 0.4. A normalized growth rate of 0.1 corresponds to a growth time of 10 oscillations at the normal frequency.

7. Test problems

With a new method like FLIP, confidence is accumulated over time by applying it to a wide range of problems. From the results, one begins to understand its properties, its strengths and its weaknesses, in ways that complement analysis. FLIP has been applied to many problems, some of which are: (The reference number before the slash is a reference for FLIP results, the reference after the slash for a description of the experiment, or other computed results.)

1. Interaction of a shock and a thin foil [8]
2. High speed flow over a step [8/15]
3. Interaction of a shock in air and a helium bubble [10/16]
4. Plane shock (Sod's problem) [14/17]
5. Noh's problem [14/18]
6. Rayleigh–Taylor instability [14/19]
7. Imploding sphere and ellipse (unpubl.)
8. Confined eddy [11/Monaghan, these proceedings]
9. Astrophysical jet [unpubl./20]

One can make several observations about the performance of FLIP from the results of these tests. FlIP gives the correct jumps in mass, momentum and energy across the shock, but the jumps are several cells thick and not as sharp as can be achieved with the piecewise parabolic method, for example [15]. Because FLIP does not have numerical diffusion, it is important to choose the correct artificial viscosity. The grid Reynolds number, $\mathrm{Re}_g = |u|\Delta x/\lambda$, should be equal to one in the shock front to give the minimum shock thickness without overshoot.

The real strength of FLIP is its ability to model hydrodynamically unstable interfaces. Both the conservation of angular momentum and the absence of dissipation appear to be essential components of this capability. An illustration of the way FLIP treats shear layers is given by a model problem.

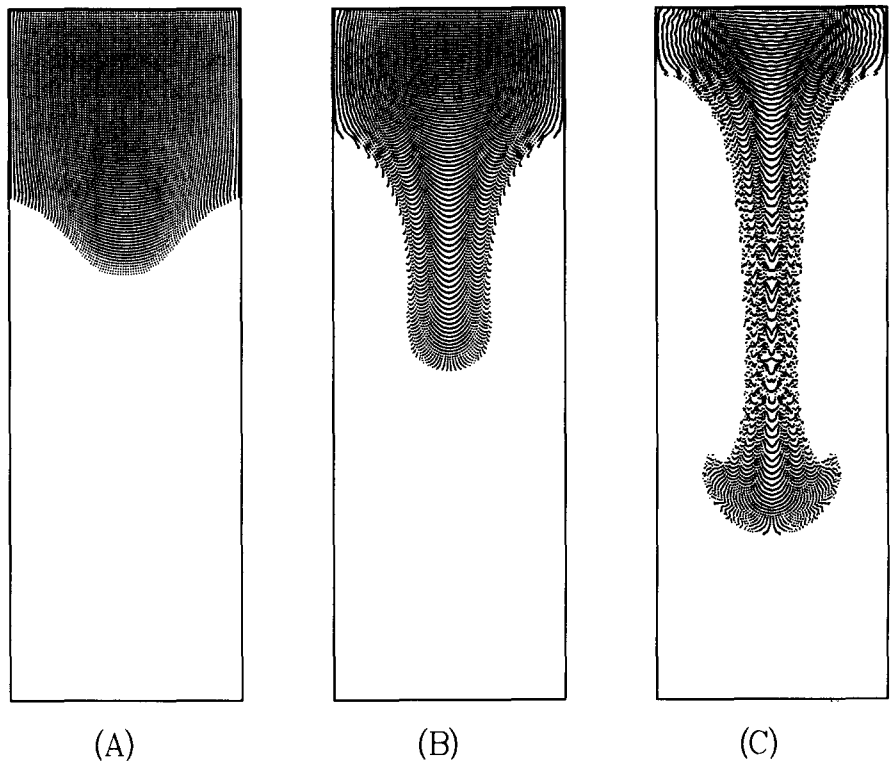


Fig. 3. Particles are plotted depicting the dense fluid in a Rayleigh–Taylor instability. In (c), the effect of the ringing instability is apparent.

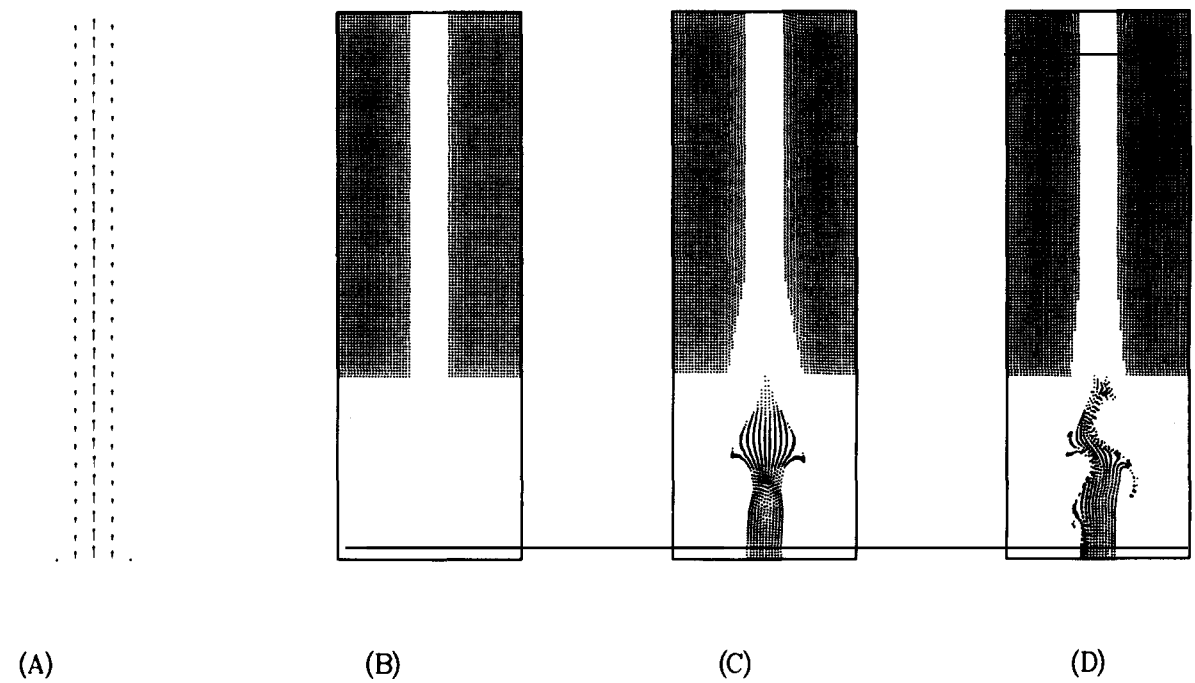
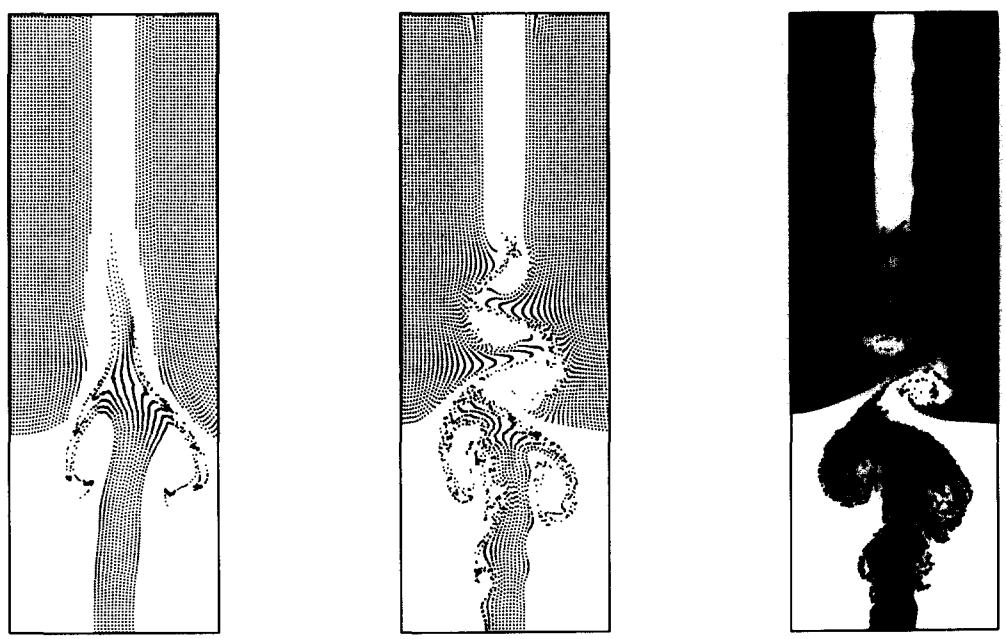


Fig. 4. A subsonic stream is modeled. The velocity is depicted in (a), the initial particle placement in the background fluid in (b), the same particles plus injected particles at $t = 5\ s/a$ on a 10×30 zone mesh in (c), and on a 20×60 zone mesh in (d).

7.1. Subsonic stream

As shown in fig. 4a, a stream of fluid enters an aperture at the bottom of the domain, flows between two flat plates, and exits at the top. The stream and the background fluid through which it flows have the same density and pressure. The distance between the plates is five times the stream thickness, s, and the length of the stream is 15 s. The Mach number for the flow is $1/\sqrt{10}$, and the artificial viscosity is zero. The particles in the upper portion of the domain at the initial time are shown in fig. 4b. The same particles, as well as those that have entered the bottom meanwhile, are shown at $t = 5\ s/a$ in fig. 4c. The dynamics of these particles is calculated on a mesh with 10×30 zones, $\Delta x = \Delta y = s/2$, and 36 particles per zone. With this resolution, there are just two zones across the stream. Because the velocity varies linearly in space, the boundary layer includes the entire stream and one zone on either side as can be seen in the relative displacement of the particles, which reflects the velocity profile. (Because the inflow velocity is constant across the aperture at the bottom boundary and varying linearly across the stream just inside the boundary, the beam is perturbed.) When the mesh spacing is halved, fig. 4d, the boundary layer thickness is halved as well, and the increased gradients in the velocity drive the stream unstable.

The subsequent development of the instability is displayed in fig. 5. With 36 particles per cell and a 10×30 mesh, the stream is gently meandering at $t = 10\ s/a$ in fig. 5a. At the same time with a 20×60 mesh and 9 particles per cell so that the total number of particles is the same, fig. 5b, the stream has become turbulent. Evidently, the instability can manifest itself only when the gradients of the flow are resolved by the mesh. Doubling the grid spacing again (40×120 zones) while keeping the number of particles per zone constant at 9 yields the results shown in fig. 5c. The larger



(A) (B) (C)

Fig. 5. Background and injected particles as in fig. 4 are plotted at $t = 10\ s/a$ for a 10×30 zone mesh, (a), a 20×60 zone mesh, (b), and a 40×120 zone mesh, (c). (a) and (b) have the same number of particles. (b) and (c) have 9 particles per cell.

features of the flow are similar to those in fig. 5b, but more rotation and mixing has occurred. Because there is no viscosity there is no physical length scale, and the results on the scale of the grid will not converge as the mesh is refined. On longer scales, the results should converge.

7.2. Supersonic jet

The unstable stream is a gentle breeze compared with the supersonic jet shown in fig. 6. (The jet shown is one of many studied by Winkler and collaborators [20] in modeling astrophysical jets.) The jet is injected through an aperture in the bottom boundary and allowed to flow freely through the right and top boundaries. The left boundary is an axis of symmetry. The domain has dimensions $6R$ by $16R$, where R is the radius of the jet, and is resolved by a 30×80 zone mesh, with 5 zones in the jet.

The jet is in pressure equilibrium with the background gas, but with only one percent of the background density. The jet speed is 30 times the ambient sound speed, a.

In fig. 6, the injected particles are plotted at intervals of $2R/a$. The interface is very unstable, and produces eddies on all scales resolved by the mesh, as is confirmed by velocity vector plots, not shown. The size of the eddies increases with distance from the head of the jet, as does the width of the turbulent flow region. The flow of the jet is unsteady. This is especially evident in figs. 6c and d, where disruptions in the jet itself can be seen

The large density discontinuity between the jet and the background gas presents some computational difficulties. For example, it is difficult to have neither too little nor too much viscosity in mixed cells. To deal with this problem, the layer of mixed cells is treated as an internal boundary. In mixed cells, where particles of two or more "colors" reside, the particle contributions are given only to surrounding cells that are mixed, or contain particles of the same color. This procedure yields, typically, a band of mixed cells only one cell thick separating two materials.

Fig. 6. Particles in a supersonic jet are plotted at intervals of $2R/a$. The particles are injected at the left-bottom. The left boundary is an axis of symmetry.

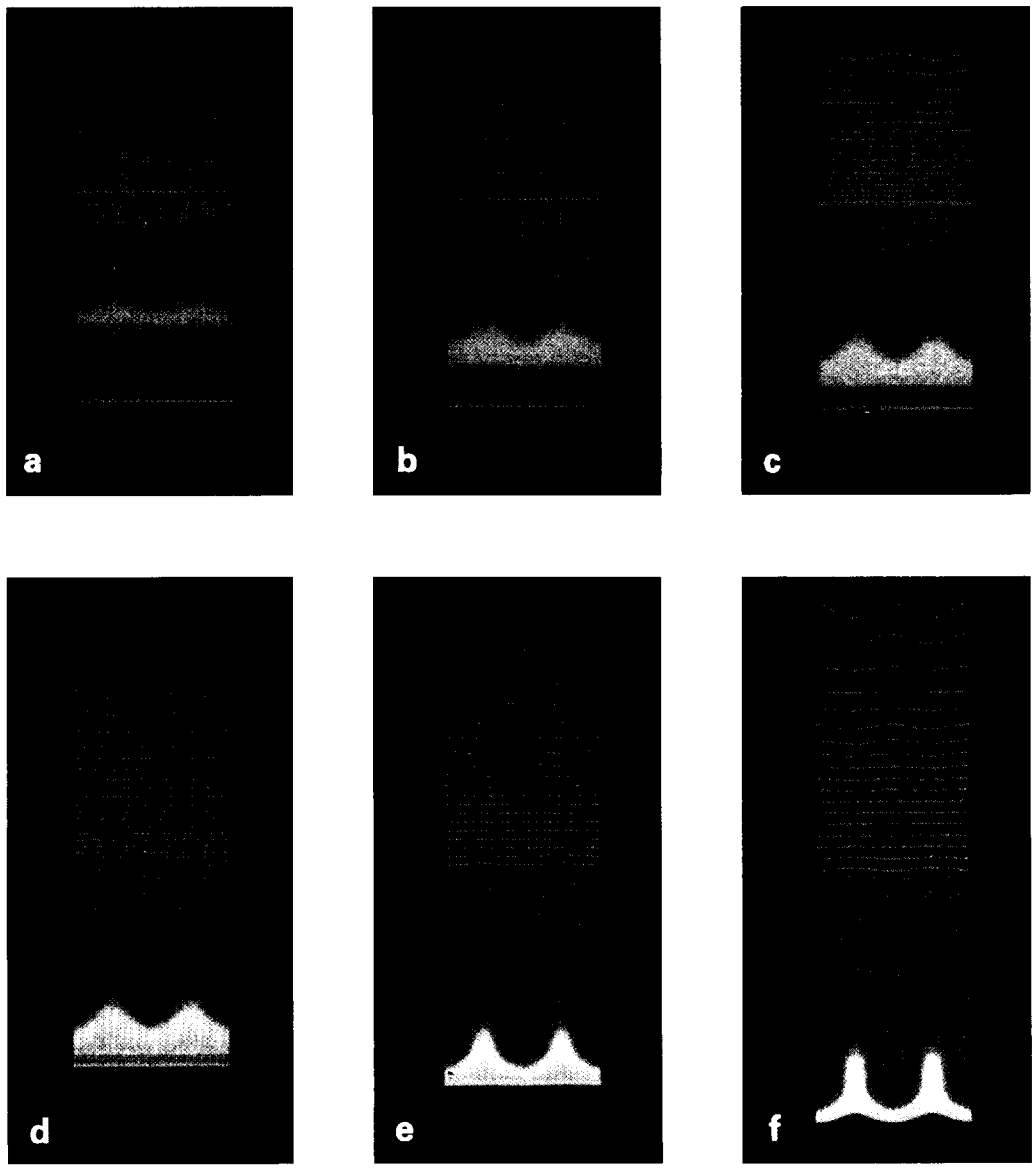


Fig. 7. Results from a calculation of the response of a target to a heavy ion beam are plotted. Green, blue and red particles represent lead, lithium and deuterium, respectively. The target is shown at 10, 17, 20, 22, 26 and 29 ns in frames (a) through (f).

7.3. Ion-beam driven hydrodynamics

FLIP can be applied to more complex problems as well as to the simple fluid flow problems described above. For example, one of us (DBK) is presently using FLIP to study ion-beam-driven targets [14]. In these, bismuth ions that have been accelerated to 10 GeV deposit 4.25 MJ of energy in 30 ns deep within a target. There, the ion energy is converted to hydrodynamic motion that compresses and heats a portion of the target material to the densities and temperatures required for fusion.

To model such targets, which are very far from thermodynamic equilibrium, the physical model in FLIP must be extended beyond the Navier–Stokes equations. Some extensions require cell-based calculations, and are simple to add. For example, to model material properties realistically one uses complex equations of state like SESAME [21]. One also must calculate the equilibration of temperatures between electrons and ions, and the ionization state of the material. Other extensions require calculating gradients, like ray-tracing to model ion beam propagation, and energy transport equations to model electron thermal conduction and convection. Luckily, the underlying grid-based data structure in FLIP resembles that of many complex physics codes, making it easy to borrow and steal the needed physical models.

The results of a calculation of the dynamics of a planar, HIBALL (Heavy Ion Beam And Liquid Lithium) target are shown in fig. 7. The colors label the target materials, with green for lead, blue for lithium, and red for deuterium. The ions, which enter the problem from above, are focused at infinity but deposit their energy in the lithium. The fluence of ions varies sinusoidally by five percent across the target, with several wavelengths included in the domain of the calculation. Where the intensity is highest, the lithium is pushed aside. There, the beam penetrates more deeply and causes a highly corrugated shock to form. The highly distorted flow that results is easily modeled by the particles.

8. Current research

Space does not permit more than a brief discussion of the adaptive zoning capability of FLIP. Adaptive zoning extends the dynamic range of FLIP, not only in spatial length scales but also in density variation. It also improves the resolution of shocks and other discontinuities. Current research emphasizes understanding the strategies one should use with FLIP to adapt the grid.

FLIP is presently being extended to multi-material flow with slip interfaces and material properties, such as strength, and to magnetohydrodynamics. Many of the ideas in FLIP are being used in developing a new, adaptively zoned, plasma simulation code in collaboration with D.W. Forslund. A version of FLIP for three-dimensional flows is an attractive, and apparently feasible, project for the future.

References

[1] F.H. Harlow, Meth. Comput. Phys. 3 (1963) 319.
[2] A. Nishiguchi and T. Yabe, J. Comput. Phys. 52 (1983) 390.
[3] C.K. Birdsall and A.B. Langdon, Plasma Physics via Computer Simulation (McGraw-Hill, New York, 1985).
[4] R.A. Clark, Los Alamos National Laboratory Report No. LA-UR 79-1947, 1979 (unpublished).
[5] B.M. Marder, Math. of Comput. 29 (1975) 473.
[6] T. Tajima, J.N. Lebouef and J.M. Dawson, J. Comput. Phys. 38 (1980) 237.
[7] R.L. McCrory, R.L. Morse and K.A. Taggart, Nucl. Sci. Eng. 64 (1977) 163.
[8] J.U. Brackbill and H.M. Ruppel, J. Comput. Phys. 65 (1986) 314.
[9] J.J. Monaghan, Comput. Phys. Rep. 3 (1985) 71.
[10] J.U. Brackbill, The Ringing Instability in Particle-in-Cell Calculations of Low-Speed Flow, J. Comput. Phys. (to be publ.).
[11] J.U. Brackbill, Comput. Phys. Commun. 47 (1987) 1.
[12] J.W. Eastwood, Comput. Phys. Commun. 44 (1987) 73.
[13] V. Casulli and D. Greenspan, Intern. J. Numer. Meth. Fluids 4 (1984) 1001.
[14] D.B. Kothe, Hydrodynamic Stability and Symmetry of Ion-Beam-Driven Planar Targets, Purdue University thesis (1987) unpubl.
[15] P. Woodward and P. Colella, J. Comput. Phys. 54 (1984) 115.
[16] J.-F. Haas and B. Sturtevant, Interaction of Weak Shock Waves with Cylindrical and Spherical Gas Inhomogeneities, Graduate Aeronautical Laboratory, California Institute of Technology, 1986 (unpubl.).

[17] G.A. Sod, J. Comput. Phys. 27 (1978) 1.
[18] W.M. Noh, J. Comput. Phys. 72 (1987) 78.
[19] B.J. Daly, Phys. Fluids 10 (1967) 297.
W.P. Crowley, An Empirical Theory for Large Amplitude Rayleigh–Taylor Instability, UCRL-72650, Lawrence Livermore National Laboratory (1970).
[20] M.L. Norman, L. Smarr, K-H.A. Winkler and M.D. Smith, Astron. Astrophys. 113 (1982) 285.
[21] K.F. Holian, T-4 Handbook of Material Properties Data Bases, LA-1060-MS, Los Alamos National Laboratory, Los Alamos, NM (1984).
[22] R.W. Hockney and J.W. Eastwood, Computer Simulation Using Particles (McGraw-Hill, New York, 1981).
[23] R.J. Mason, in: Multiple Time Scales, eds. J.U. Brackbill and B.I. Cohen (Academic Press, Orlando, 1985) pp. 233–270.
J.U. Brackbill, ibid. pp. 271–310.
[24] J.U. Brackbill and J.S. Saltzman, J. Comput. Phys. 46 (1982) 342.
[25] A.E. Giannakopoulos and A.J. Engel, Directional Control in Grid Generation, J. Comput. Phys., to be published.
[26] J.U. Brackbill, Solution Adaptive Grids for Time-Dependent Problems in Two and Three Dimensions, J. Comput. Phys., submitted.
[27] R.D. Milroy and J.U. Brackbill, Phys. Fluids 25 (1982) 775.