DOE/ET-53088-564 IFSR #564

Particle Simulation Algorithm with Short Range Forces in MHD and Fluid Flow

S. CABLE, T. TAJIMA, and K. UMEGAKI
Institute for Fusion Studies
The University of Texas at Austin
Austin, Texas 78712

July 1992

Short Communication

Particle Simulation Algorithms with Short-Range Forces in MHD and Fluid Flow

S. Cable, T. Tajima, and K. Umegaki
Institute for Fusion Studies
The University of Texas at Austin
Austin, Texas 78712

Abstract

Attempts are made to develop numerical algorithms for handling fluid flows involving liquids and liquid-gas mixtures. In these types of systems, the short-range intermolecular interactions are important enough to significantly alter behavior predicted on the basis of standard fluid mechanics and magnetohydrodynamics alone. We have constructed a particle-in-cell (PIC) code for the purpose of studying the effects of these interactions. Of the algorithms considered, the one which has been successfully implemented is based on a MHD particle code developed by Brunel *et al.*In the version presented here, short range forces are included in particle motion by, first, calculating the forces between individual particles and then, to prevent aliasing, interpolating these forces to the computational grid points, then interpolating the forces back to the particles. The code has been used to model a simple two-fluid Rayleigh-Taylor instability. Limitations to the accuracy of the code exist at short wavelengths, where the effects of the short-range forces would be expected to be most pronounced.

I. Introduction

Fluid dynamics and magnetohydrodynamics (MHD) have been studied quite often through the modern techniques of computer simulation. There have been a large number of research and review papers on such methods.

Nevertheless, very few attempts have been made to include short-range interactions in dynamical fluid or MHD computation. Phenomena arising from these interactions include liquid-like behavior such as surface tension and consequent droplet formation. The short-range forces tend to be quite intense at molecular distances but carry almost no influence beyond these distances. For example, at the interface of a liquid and gas, the surface tension arising from the short-range forces can maintain the density of the liquid at values many times greater than the density of the adjacent gas. It is this severe discrepancy of scales in density, arising from the quick fall-off in strength of short-range forces with distance, that makes stiff demands on computational models. Moreover, typical fluid simulation techniques employing a continuum model may face difficulty (or, at least, awkwardness) in treating the boundaries between gas and liquid phases when the boundary can rapidly change.

One commonly used technique involves removing the advective nonlinearities in the fluid equations by following the characteristics of moving fluid elements [1-4]. This technique is equivalent, in principle, to following a model macromolecule (or plasma particle) that posesses appropriate properties such as mass and momentum as well as perhaps some interaction properties.

It behooves us to consider a computational model which includes short-range forces to model liquid behavior, as this model perhaps allows natural tracking of freely changing boundaries of different phases. Applications may include not just the mixture of a liquid and a gas but a wider range of multi-phase fluid sytems, such as admixtures of oil and water.

II. Adiabatic Fluid Algorithm

We have constructed a PIC code for the purpose of studying the effects of short-range, intermolecular forces on fluid and MHD flow. Here we present the algorithm we have used. We also present two other algorithms which can, hopefully, be helpful in overcoming timescale problems.

The algorithm is based on an adiabatic MHD particle code developed by Brunel *et al.* [5]. In this algorithm, computational particles carry mass and momentum. Fluid quantities such as pressure, density, and fluid velocity are interpolated from the particles to a computational grid. In return, these quantities are used to construct fluid forces which are interpolated back to the particles and used to alter the particle velocities.

The algorithm works as follows:

(1) Fluid density is computed: $\rho^n(\mathbf{x}) = \sum_p S_{pv}(\mathbf{x}^n - \mathbf{x}_p^n)$ where S_{pv} is the interpolation function and the sum is over all particle positions within some predetermined distance from the cell at $\mathbf{x}$.

(2) Pressure accelerations are calculated, as well as magnetic accelerations, if we are studying MHD flow:

$$\mathbf{F}_{pr}^n = -\nabla p^n/\rho^n \ , \quad \text{where} \quad p = p_0(\rho/\rho_0)^\gamma \tag{1}$$

$$\mathbf{F}_B^n = ((\nabla \times \mathbf{B}^n) \times \mathbf{B})^n / 4\pi\rho^n \ . \tag{2}$$

(3) Fluid accelerations are interpolated to particle positions: $\mathbf{F}_p^n = \sum_v S_{pv}(\mathbf{x}^n - \mathbf{x}_p^n)(\mathbf{F}_{pr}^n + \mathbf{F}_B^n)$ where the sum is over all grid cells within a predetermined distance of the particle p.

(4) Short-range accelerations are calculated for each particle:

$$\mathbf{F}_{sp}^n = \sum_q \mathbf{F}_s(\mathbf{x}_p^n - \mathbf{x}_q^n) \ . \tag{3}$$

These accelerations are then interpolated to the grid cells and then back to the particles. In our particular code, the accelerations were calculated by a PPPM method [6]. Effectively, at each time step, particles are assigned a "box" based on thier positions. The box size is comparable to the range of the short range force. When pair-wise interactions are calculated, only particle pairs in the same boxes, or in nearest neighbor boxes, are considered as possible candidates for interactions.

(5) Particle velocities are advanced by a half time-step:

$$\mathbf{v}_p^n = \mathbf{v}_p^{n-1/2} + \frac{\Delta t}{2}(\mathbf{F}_p^n + \mathbf{F}_{sp}^n) \ . \tag{4}$$

(6) New fluid velocities are computed by interpolation: $\mathbf{v}_f^n = \sum_p S_{pv}(\mathbf{x} - \mathbf{x}_p)\mathbf{v}_p^n$.

(7) The magnetic field is advanced a half time-step using a Lax method:

$$\mathbf{B}^{n+1/2} = \langle \mathbf{B}^n \rangle + \frac{\Delta t}{2}(\nabla \times (\mathbf{v}_f^n \times \mathbf{B}^n)) \tag{5}$$

where $\langle \mathbf{B}^n \rangle_{i,j} = (\mathbf{B}_{i+1,j}^n + \mathbf{B}_{i-1,j}^n + \mathbf{B}_{i,j+1}^n + \mathbf{B}_{i,j-1}^n)/4$.

(8) Particle velocities are advanced another half time-step:

$$\mathbf{v}_p^{n+1/2} = \mathbf{v}_p^n + \frac{\Delta t}{2}(\mathbf{F}_p^n + \mathbf{F}_{sp}^n) + \nu(\mathbf{v}_f^n - \mathbf{v}_p^n) \tag{6}$$

where ν is an arbitrary constant ranging from 0 to 1. $\nu = 1$ prevents multi-streaming.

(9) Particle positions are pushed a half time-step:

$$\mathbf{x}^{n+1/2} = \mathbf{x}^n + \mathbf{v}^{n+1/2}\frac{\Delta t}{2} \ . \tag{7}$$

(10) New fluid velocities are calculated by interpolation:

$$\mathbf{v}_f^{n+1/2} = \sum_p S_{pv}(\mathbf{x} - \mathbf{x}_p)(\mathbf{v}^{n+1/2}, \mathbf{x}^{n+1/2}) \ . \tag{8}$$

(11) The magnetic field is pushed a full time-step, completing the Lax scheme:

$$\mathbf{B}^{n+1} = \mathbf{B}^n + \Delta t \left(\nabla \times (\mathbf{v}_f^{n+1/2} \times \mathbf{B}^{n+1/2})\right) \ . \tag{9}$$

(12) The algorithm cycle is completed with the final advance of the particle positions:

$$\mathbf{x}^{n+1} = \mathbf{x}^{n+1/2} + \mathbf{v}^{n+1/2}\frac{\Delta t}{2} \; . \tag{10}$$

Initially, an attempt was made to add the short range accelerations directly to the particles in step (4). This approach had to be abandoned because of problems with spatial aliasing of short wavelength fluctuations in the fluid quantities. The interpolation scheme acts as a spatial low-pass filter on the short-range force effects.

In an effort to get past the limitation on timestep imposed by the sound speed of the adiabatic fluid described above, an alternative incompressible fluid algorithm may be considered. It proceeds thusly:

(1) Fluid density, velocity, short-range forces, and magnetic field are calculated at each grid point. This is done in the same way as in the previous algorithm.

(2) Acceleration from the magnetic field and short-range forces are calcualted on the grid points:

$$\mathbf{A}_B^n = ((\nabla \times \mathbf{B}^n) \times \mathbf{B})^n / 4\pi\rho^n \; . \tag{11}$$

(3) Preliminary values of the updated fluid velocities are calculated:

$$\widetilde{\mathbf{U}}_v^n = \mathbf{U}_v^{n-\theta} + \mathbf{A}_v^n \theta \Delta t \tag{12}$$

where $\mathbf{A}_v^n = \mathbf{A}_{Bv}^n + \mathbf{A}_{sv}^n$.

(4) The velocity field is now made incompressible. This is accomplished by first solving a Poission-type equation for the velocity potential ϕ:

$$\nabla \cdot \left(\nabla \phi_c^n + \frac{(\nabla \rho^n)_v}{\rho_v^n} \phi_v^n \right) = \nabla \cdot \widetilde{\mathbf{U}}_v^n \tag{13}$$

where the subscript c indicates a quantity defined at the centers of the grid cells, as opposed to quantities defined at the grid points, which are labeled by v. The velocity potential ϕ is of physical significance in that it is proportional to the pressure divided by the density:

$$\phi_c^n = \frac{p_c^{n+\theta}}{\rho_c} \theta \Delta t \; . \tag{14}$$

$\mathbf{U}_v^n$ is now calculated by

$$\mathbf{U}_v^n = \widetilde{\mathbf{U}}_v^n - \left(\nabla\phi_c^n + \frac{(\nabla\rho^n)_v}{\rho_v^n}\phi_v^n\right) . \tag{15}$$

(5) The magnetic field is advanced to its preliminary values:

$$\mathbf{B}^* = \langle\mathbf{B}^n\rangle + \frac{\Delta t}{2}(\nabla \times (\mathbf{U}_v^n \times \mathbf{B}^n)) , \tag{16}$$

where $\langle\mathbf{B}^n\rangle_{i,j} = (\mathbf{B}_{i+1,j}^n + \mathbf{B}_{i-1,j}^n + \mathbf{B}_{i,j+1}^n + \mathbf{B}_{i,j-1}^n)/4$.

(6) The fluid velocities are advanced to the next timestep:

$$\mathbf{U}_v^{n+1-\theta} = [\mathbf{U}_v^n - (1-\theta)\mathbf{U}_v^{n-\theta}]/\theta . \tag{17}$$

(7) Updated particle velocities are interpolated from the grid cells:

$$\mathbf{U}_p^{n+1-\theta} = \mathbf{U}_p^{n-\theta} + \sum_v(\mathbf{U}_v^{n+1-\theta} - \mathbf{U}_v^{n-\theta})S_{pv} \tag{18}$$

where S_{pv} is the function of interpolation from grid point v to particle p.

(8) The magnetic field is advanced to the next time step:

$$\mathbf{B}^{n+1} = \mathbf{B}^n + \Delta t(\nabla \times (\mathbf{U}_v^{n+1-\theta} \times \mathbf{B}^*)) . \tag{19}$$

(9) The cycle is completed by pushing the particle positions a full timestep:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_v \mathbf{U}_p^{n+1-\theta} . \tag{20}$$

In our particular code, θ was set equal to 1/2. The Poisson equation in step (4) was solved by an SOR method [7].

An alternative third algorithm for introducing short-range force effects has also been considered, though it is not yet implemented ina working code. Short-range forces are not explicitly calculated. Rather, a surface tension fluid force is calculated wherever the fluid density gradient exceeded some predetermined value ϵ. In the physical equations of motion of fluid or MHD flow, the surface tension force at a sharp interface is given by

$$\mathbf{F}_T = \sigma\kappa\hat{\mathbf{n}}\delta(\mathbf{x} - \mathbf{x}_s) , \tag{21}$$

where σ is the surface tension coefficient, and κ is the local radius of curvature of the interface, defined along with the unit vector $\hat{\mathbf{n}}$ so that the force always points into the fluid from which the interface appears concave.

To adapt this force to a fluid code mesh, we make the change

$$\sigma\kappa\hat{\mathbf{n}}\delta(\mathbf{x}-\mathbf{x}_s) = \begin{cases} -\sigma\hat{\mathbf{n}}\cdot\nabla\hat{\mathbf{n}} \text{ if } |\nabla\rho| \geq \epsilon \\ \quad 0 \qquad \text{if } |\nabla\rho| < \epsilon \end{cases}, \tag{22}$$

where $\hat{\mathbf{n}} = \nabla\rho/|\nabla\rho|$ and ϵ is chosen beforehand. This method of handling surface tension was developed by Brackbill, Kothe, and Zemach [8].

The algorithm proceeds in a fashion similar to the incompressible algorithm outlined above, with two exceptions. First, as has been stated, short-range forces are not directly calculated. Second, surface tension is included in the calculation of the preliminary velocities $\widehat{\mathbf{U}}^n$. This is accomplished by an implicit inner iteration scheme in which the continuity and momentum equations are coupled:

$$\mathbf{R}^{n+\theta}_{m+1} = -\nabla\widetilde{\mathbf{U}}^{*n}_{m}\cdot\mathbf{R}^{n+\theta}_{m+1}\theta\delta t + \mathbf{R}^n$$

$$\widehat{\mathbf{R}}^{n+\theta}_{m+1} = \mathbf{Q}^{n+\theta}_{m}\cdot(\nabla\widetilde{\mathbf{U}}^{*n}_{m}\cdot\widehat{\mathbf{R}}^{n+\theta}_{m+1})\theta\delta t + \widehat{\mathbf{R}}^n$$

$$\widetilde{\mathbf{U}}^{n}_{m+1} = \left\{\mathbf{A}^n_{Bv} - \frac{\sigma\mathbf{R}^{n+\theta}_{m+1}}{[\rho]\langle\rho\rangle}\nabla\cdot\mathbf{R}^{n+\theta}_{m+1}\right\}\theta\delta t + \mathbf{U}^n$$

$$\widetilde{\mathbf{U}}^{*n+\theta}_{m+1} = \omega(\mathbf{U}^{n+\theta}_{m} - \mathbf{U}^{n+\theta}_{m+1}) + \mathbf{U}^{n+\theta}_{m-1},$$

where

$$\mathbf{R} \equiv \nabla\rho,$$

$$\widehat{\mathbf{R}} \equiv \frac{\nabla\rho}{|\nabla\rho|},$$

$$\widehat{\mathbf{Q}} \equiv \widehat{\mathbf{R}}\widehat{\mathbf{R}} - \mathbf{I}, \tag{23}$$

and $[\rho]$ is the difference in density across the interface and $\langle\rho\rangle$ is the average of the density across the interface. Upon convergence, $\mathbf{U}^n = \widetilde{\mathbf{U}}^n_{m+1}$. The last two algorithms discussed have not been successfuly implemented at this time.

III. Simulation of a Rayleigh-Taylor Instability

We implemented the first algorithm in a simulation of a two-fluid Rayleigh-Taylor unstable system. The simulation code is two-dimensional. All distances were normalized to the grid spacing Δ. All velocities were normalized to the sound speed $c_s = \gamma p_0/\bar{\rho}$ where $\bar{\rho}$ is the average mass density and p_0 is an arbitrary physical pressure. Time was normalized to $\Delta t = \Delta/c_s$. All mass was normalized to the particle mass. The computational grid size was 64×64. The timestep was $\Delta t = 0.05$.

The initial density profile was

$$\rho(y) = \frac{\rho_1 + \rho_2}{2} + \frac{\rho_2 - \rho_1}{2} \tanh\left(\frac{y - y_{\text{int}}}{y_{\text{size}}}\right) ,$$

where ρ_1 was the density at the bottom of the computational area, ρ_2 was the density at the top of the area, y_{int} was the location of the center of the interface region, and y_{size} was the approximate thickness of the interface region. In our particular simulation, $y_{\text{int}} = 32$, placing the interface half-way between the top and bottom of the compuational area, and $y_{\text{size}} = 4$. ρ_1 was three particles per cell and ρ_2 was 12.3 particles per cell. Particle positions were initialized by first placing the particles in a series of horizontal rows. The x-direction spacing between particles was kept constant. The y-direction spacing between rows was set inversely proportional to the local density. Then, in each row of particles, alternate particles were moved one half of the distance up to the next row of particles. This was done to keep the particles in the thin region outside of one another's range of interaction. The entire system was made subject to a gravitational acceleration of $g = -0.05\hat{y}$, and was stabilized by an inhomogeneous magnetic field in the z-direction (*i.e.* perpendicular to the computational area). Boundary conditions were chosen to be periodic at the x boundaries and reflective at the y boundaries.

Outside of the interface region, the fluid velocities were perturbed according to

$$v_{fx} = \sin(kx) \times \begin{cases} -0.1 \times \dfrac{\cosh(ky)}{\sinh(ky_{\text{int}})} , & y < y_{int} - y_{\text{size}}/2 \\ p(y) , & y_{\text{int}} - y_{\text{size}}/2 < y < y_{\text{int}} + y_{\text{size}}/2 \\ 0.1 \times \dfrac{\cosh(k(y_{\max} - y))}{\sinh(k(y_{\max} - y_{\text{int}})} , & y > y_{\text{int}} + y_{\text{size}}/2 . \end{cases} \tag{24}$$

and

$$v_{fy} = \cos(kx) \times \begin{cases} -0.1 \times \dfrac{\sinh(ky)}{\sinh(ky_{\text{int}})} , & y < y_{\text{int}} - y_{\text{size}}/2 \\ q(y) , & y_{\text{int}} - y_{\text{size}}/2 < y < y_{\text{int}} + y_{\text{size}}/2 \\ 0.1 \times \dfrac{\sinh(k(y_{\max} - y))}{\sinh(k(y_{\max} - y_{\text{int}}))} , & y > y_{\text{int}} + y_{\text{size}}/2 , \end{cases} \tag{25}$$

where $p(y)$ and $q(y)$ are second and third order polynomials respectively. These polynomials were chosen so that 1) the fluid flow would be initially incompressible in the interface region, as it was in the rest of the computational area, 2) v_y and $\partial_y v_y$ would be continuous at $y = y_{\text{int}} - y_{\text{size}}/2$ and at $y = y_{\text{int}} + y_{\text{size}}/2$, and 3) v_x would be continuous at $y = y_{\text{int}} - y_{\text{size}}/2$ and at $y = y_{\text{int}} + y_{\text{size}}/2$. The above velocities were assigned as fluid velocities to the computational grid points. Particle velocities were then initialized by interpolation from the grid.

For an initial test of the code, the short-range force on a particle 1 from a particle 2 was chosen to be particularly simple, namely

$$\mathbf{F}_{sr}(\mathbf{x}_1 - \mathbf{x}_2) = \begin{cases} \sin(\pi|\mathbf{x}_1 - \mathbf{x}_2|/r_{eq}) \dfrac{(\mathbf{x}_1 - \mathbf{x}_2)}{|\mathbf{x}_1 - \mathbf{x}_2|} ; & |\mathbf{x}_1 - \mathbf{x}_2| < 2r_{eq} \\ 0 & |\mathbf{x}_1 - \mathbf{x}_2| > 2r_{eq} . \end{cases} \tag{26}$$

The value of r_{eq} was chosen as $\sqrt{\rho_2}$. With this choice of a short range force, particle oscillation periods would be on the order of $1.9 \times \Delta/c_s$, so they would not be too short for the time step to handle.

The short-range forces could be expected to effect the system in a manner reminiscent of surface tension, namely, reducing the growth-rate of the instability [9]. However, our preliminary run of a 110 timestep simulation gave only ambiguous confirmation of this expectation so for. A simulation was first run with the short-range force "turned off". The growth rate of the instability was calculated from the square root of the kinetic energy, which is plotted as a function of time in Fig. 1. The kinetic energy followed a $\cosh(\gamma t)$ type of curve at early times. At the time that its growth became clearly exponential, it had reached a value of about 21.75. The growth rate was 0.033 in simulation units. This is in the ballpark with the classical value of 0.054. (It should be noted that attempts to measure the growth by the "bend" in the interface did not reveal a clear regime of exponential growth.)

A simulation with the short range force described above was also performed for 110 timesteps. When the kinetic energy began to grow in a clearly exponential manner, its value was about 16.44, about 24% less than the value of the kinetic energy without short-range forces. (See Fig. 2.) However, when exponential growth actually did begin, the growth rate was 0.074, over twice as high as the growth rate without short-range forces. By the end of the runs, the values of the two kinetic energies were comparable.

It might be thought that the added kinetic energy growth in the short-range force simulation came from particles moving toward more stable local equilibria with one another. It is true that the initial particle arrangement was not a true equilibrium. In a true equilibrium, the dense region particles would be arranged in a hexagonal configuration with nearest-neighbor particles separated approximately by the equilibrium distance of the short-range force [10]. In contrast, the inital arrangement of dense region particles was square packed, with nearest neighbors separated by $1/\sqrt{2}$ of the equilibrium distance. Of course, the particles pushing against one another would create a type of equilibrium, at least temporarily. The truly problematic particles would be the ones near the interface. In order to keep the interface density gradient small enough for the computational grid to handle, the interface particles had to separated by distances greater than the equilibrium distance but less than the cut-off distance of the short-range force. This means that the particles near the interface were strongly attracted to one another. This point needs further study.

However, it would appear that this non-equilibrium arrangement is not the source of the added growth of kinetic energy. The square root of the kinetic energy plus the change in the short-range potential energy is plotted as a function of time in Fig. 3. If the added kinetic energy growth came from particles moving toward local equilibria, the short-range potential energy would become more negative. Therefore, the kinetic energy added to the potential energy would grow more slowly than the kinetic energy in and of itself. However, this is not what happened. The kinetic energy plus the change in the potential energy grew much faster than the kinetic energy alone. In fact, this quantity had its own range of exponential growth; its growth rate was 0.65. In late times, this quantity drops off markedly, beginning at $t = 2.2$. However, this does not explain the enhanced growth at early times. It might be that the strong short-range force enhances the available free energy at the early (settling) stage of the run. Note that the simple theory does not include this effect so that our theoretical knowledge is incomplete here. We most likely need a more appropriate form of short-range forces, which will be explored in the future.

IV. Conclusions

We have developed algorithms that allow natural treatment of multiphased fluid sytems where short-range forces influence the dynamics in the fluid or MHD equations. A preliminary implementation and its application to a Rayleigh-Taylor instability are reported. This attempt is preliminary but, to our knowledge, is a new attempt to approach the above mentioned problems that have been largely left unattacked to date.

The algorithm described here amounts to a "marriage" of a fluid algorithm and a molecular dynamics algorithm. It has been managed to some degree, but is not without its expected problems. First and foremost is that, in a fluid code, short-range forces can be expected to act similarly to surface tensions. However, surface tensions have their most dramatic effects at short wavelengths [9]. It is precisely in such situations that the fluid algorithm fails. This problem can be overcome to some degree by increasing grid resolution (and particle number) at the expense of computing time. Finding a more elegant solution to the problem must be deferred to further research. The code does seem to perform well at long wavelengths, however. A second problem is related to timescale. It may be desirable to introduce short-range forces with oscillation periods much shorter than the timescale of the overall fluid motion. To avoid overly long computer runs, this problem will have to be dealt with by altering the algorithm, probably by introducing some type of implicit time stepping scheme.

Of the algorithms presented in Sec. 2, we have had success in implementing only the one described here. Writing a code based on the third (continuous surface tension) algorithm will be tried in the future. The second (incompressible) algorithm has not been successfully implemented in modeling a Rayleigh-Taylor instability. In a preliminary attempt at implementation, a code based on this algorithm lost a substantial part of its kinetic energy in a 500 timestep run. This problem arose even with short-range forces "switched off." Several minor variations on the pushing of fluid and particle quantities were made to the code, none of which returned significant improvement. Another incompressible algorithm has been developed, based on the Eulerian algorithm of Aydemir and Barnes [11]. It has exhibited similar difficulties. The element common to both codes was the SOR routine used to isolate the compressible part of the fluid velocity. Perhaps this routine is introducing artificial viscosity into the fluid flow and needs to be replaced.

This work was supported in part by the U.S. Department of Energy contract DE-FG05-80ET-53088.

References

1. F.H. Harlow, *in* "Methods in Computational Physics" (B. Alder *et al.*, Eds.), Vol. 3, p. 319, Academic Press, New York, 1964.

2. R.W. Hockney and J.W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).

3. J.U. Brackbill and J.J. Monaghan, *Particle Methods in Fluid Dynamics and Plasma Physics* (North-Holland, Amsterdam, 1988).

4. T. Tajima, *Computational Plasma Physics* (Addison-Wesley, New York, 1989), Ch. 6.

5. F. Brunel, J.N. Leboeuf, T. Tajima, and J.M. Dawson, J. Comput. Phys. **43**, 268 (1981).

6. R.W. Hockney and J.W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).

7. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes* (Cambridge University Press, 1986).

8. J.U. Brackbill, D.B. Kothe, and C. Zemach, LANL preprint, 1990.

9. S. Chandrasekhar, *Hydrodynamic and Hydromagnetic Stability* (Oxford, 1961).

10. D. Greenspan, unpublished preprint.

11. A.Y. Aydemir and D.C. Barnes, J. Comput. Phys. **53**, 100 (1984).

Figure Captions

Fig. 1. Growth of square root of kinetic energy in simulation of Rayleigh-Taylor instability. No short-range force effects included.

Fig. 2. Growth of square root of kinetic energy in simulation of Rayleigh-Taylor instability. Short-range forces have been added to equations of particle motion.

Fig. 3. Square root of kinetic energy plus short-range force potential energy.

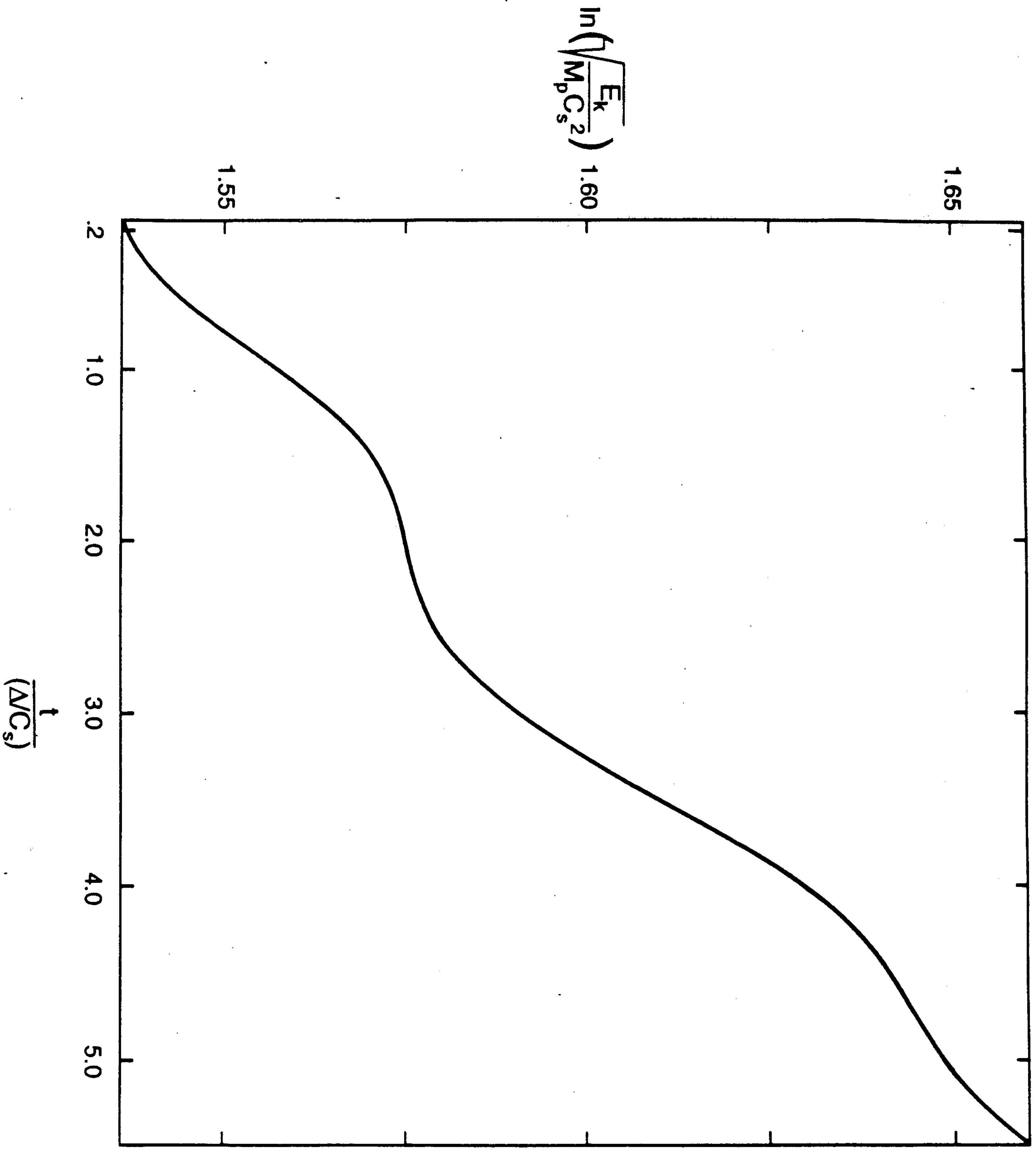
ln$\left(\sqrt{\frac{E_k}{M_p C_s^2}}\right)$
1.55
1.60
1.65
.2
1.0
2.0
3.0
4.0
5.0
$\frac{t}{(\Delta/C_s)}$

Fig. 1

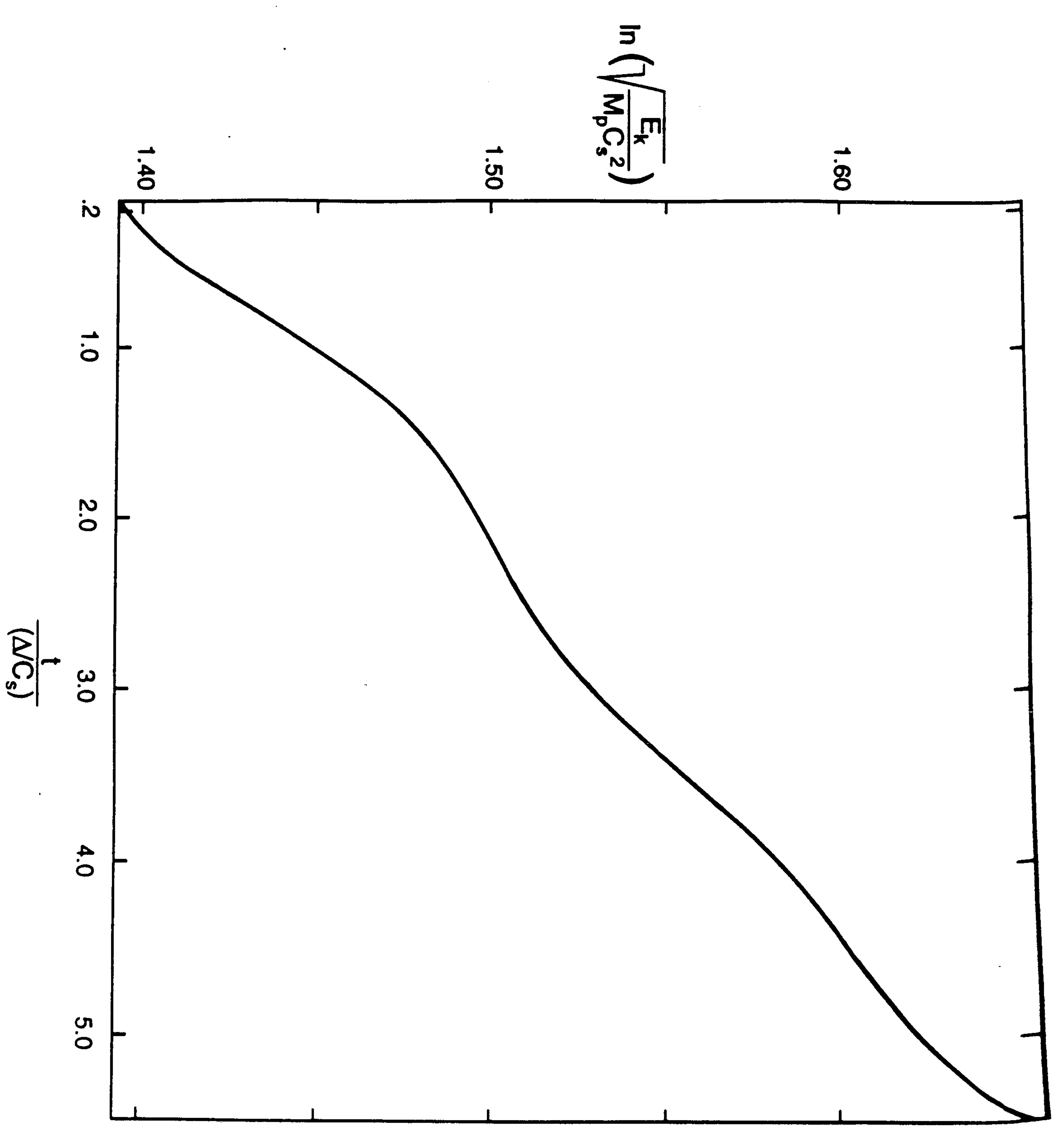
$\ln\left(\sqrt{\frac{E_k}{M_p C_s^2}}\right)$
1.40
1.50
1.60
.2
1.0
2.0
3.0
4.0
5.0
$\frac{t}{(\Delta/C_s)}$

Fig. 2

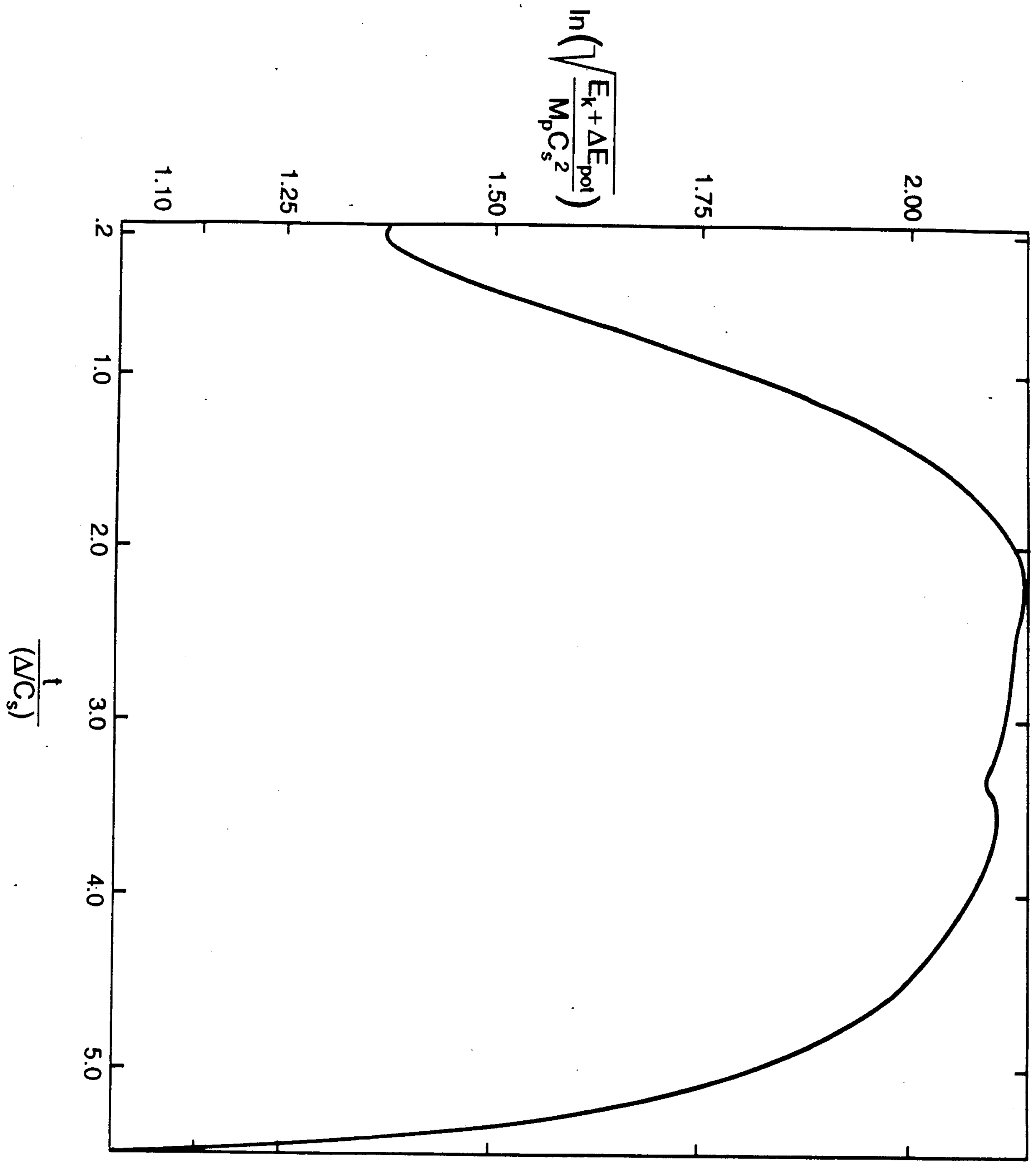
$\ln\left(\sqrt{\frac{E_k+\Delta E_{pot}}{M_p C_s^2}}\right)$
1.10
1.25
1.50
1.75
2.00
.2
1.0
2.0
3.0
4.0
5.0
$\frac{t}{(\Delta/C_s)}$

Fig. 3