JOURNAL OF COMPUTATIONAL PHYSICS **96**, 54–70 (1991)

Multi-scale Particle-in-Cell Plasma Simulation

A. FRIEDMAN

Lawrence Livermore National Laboratory, University of California, Livermore, California 94550

S. E. PARKER

Electronics Research Laboratory, University of California, Berkeley, California 94720

S. L. RAY

Lawrence Livermore National Laboratory, University of California, Livermore, California 94550

AND

C. K. BIRDSALL

Electronics Research Laboratory, University of California, Berkeley, California 94720

Received September 29, 1989; revised March 23, 1990

We describe a form of self-consistent particle-in-cell plasma simulation which is applicable to strongly inhomogeneous systems involving a wide range of space and time scales. In this *multi-scale* method, the plasma particles in each region of phase space are advanced using a step size appropriate to that region, as determined by accuracy considerations. While the necessity of a self-consistent field may seem to require processing of all particles in synchrony, the method overcomes that difficulty. This is accomplished by means of implicit PIC techniques, interpolating grid quantities in time to obtain the source contributions from groups of particles not advanced during the current step. For suitable problems (those in which fine space-time resolution is needed only in isolated spatial regions), most of the particles are not processed on any given step. Thus, major gains in efficiency over conventional simulations may be realized. In this paper we describe the method, and the beginnings of our investigations into its feasibility. © 1991 Academic Press, Inc.

1. INTRODUCTION

A long-standing goal in plasma simulation has been a method which could treat both detailed kinetic and smooth large-scale physics in an efficient and natural way. Until recently, particle simulations were applied almost exclusively to problems of "microscopic" physics, where only a small part of the plasma was modeled. With the advent of implicit particle simulation techniques [1–5], one can now treat long systems of many thousands of Debye lengths and follow them for many thousands of plasma oscillation times. However, the price paid for this capability is a restriction on the allowed spatial resolution (an accuracy constraint). Thus, in current particle codes (e.g., TESS [6] or AVANTI [7]) a small timestep is still necessary whenever the system incorporates a physically important small spatial or temporal scale anywhere within its domain.

We have developed a new *multi-scale* particle-in-cell plasma simulation technique which relaxes these restrictions and is suitable for strongly inhomogeneous problems involving a wide range of space and time scales. The plasma in any part of phase space is advanced on its own natural scales. Of course, it seems only natural to advance each particle with its own, independent, series of timesteps. The major difficulty in doing this has been the necessity of processing the particles in synchrony due to the requirement of a self-consistent field. We have developed and are testing an algorithm [8, 9] which overcomes this difficulty. The code advances the particles in blocks, each with an associated timestep Δt which equals the base (smallest) timestep times a power of two. Direct-implicit techniques are used to produce charge density data many base-steps ahead of the current one, for blocks which are infrequently processed. Interpolations in time yield source arrays a single step in advance. For each region of phase space, the nominal timestep (and possibly the mesh spacing) is independently specified; it can either be fixed, or chosen in an adaptive manner. A major improvement in economy comes about because the majority of the particles are not processed during any given step; for suitable problems this gain may be an order of magnitude or more.

There are many areas of plasma physics where such a capability would be highly desirable. In particular, bounded plasmas are in need of and well suited to multi-scale techniques (indeed, they motivated this invention). In many cases, details of both sheaths (a few Debye lengths) and the bulk plasmas they bound (many thousands of Debye lengths) are desired. Other potential applications include collisionless shocks, double layers, and a wide variety of astrophysical problems. Even in a relatively tractable problem such as the bump-on-tail instability, significant gains might be realized by pushing only those particles in or near the bump with a small timestep; the more numerous bulk particles would be advanced less frequently. Similar methods might prove applicable to problems of gravitating systems and to particle-in-cell fluid modeling.

We have implemented the algorithm in a testbed code (MIST). Our first tests, described below, exercise the method without allowing particle step sizes to change with time; such "redistribution" of particles among blocks has been implemented and tested successfully, but will be described in a future paper.

In Section 2 below, we describe the method; in Section 3 we present the results of tests which serve to illustrate some of its properties; and in Section 4 we describe how the method would apply to the problem of a macroscopic plasma bounded by an electrostatic sheath. Concluding remarks appear in Section 5.

2. Multi-scale Method

We motivate the method by considering briefly the sheath problem described more thoroughly in Section 4 below. In that problem, particles in the sheath are necessarily advanced with small timesteps—they are thus kept in the smallest timestep "group" which is pushed every time the field solution is performed. In contrast, particles in the bulk plasma need be advanced only infrequently. However, accuracy and smoothness of the simulation is enhanced by not processing all particles in this least-frequently-advanced group at the same time—hence, the group is divided into subgroups ("blocks") each of which is actively processed on different timesteps. The multi-scale scheme offers real advantages over conventional explicit and implicit methods for inhomogeneous problems such as this one. When the bulk region is large enough it is completely impractical to use an explicit method; an implicit method would fail to capture the internal sheath dynamics. The reader is referred to Section 4 for further discussion. We now describe the algorithm in detail.

Each group of particles (call them G_m) is pushed every 2^m timesteps using the time increment $\Delta t_m = 2^m\,\delta t$, where $m = 0, 1, 2, 3, ..., m_{\max}$, and δt is the smallest time increment. To avoid processing all the particles of a given group at once, we define subgroups of G_m called blocks, B^l_m. Given a group G_m, there are 2^m blocks B^l_m, where $l = 0, 1, 2, ..., (2^m - 1)$. Each block in a group is moved on a different timestep. We move a block B^l_m on timestep n (with $t = n\,\delta t$) when:

$$(n) \bmod (2^m) = l. \tag{1}$$

The model is shown in Fig. 1. Each species of particles has its set of associated blocks.

Timestep sizes which differ from the smallest step size by powers of two are used because, as can be seen in the algorithm of Section 2.2, it is particularly convenient to double or halve the step size of a particle; see especially steps (1h) and (1i). Furthermore, an "exponential" dependence of the step size upon the group number

Group	Step size	Block	When pushed
G_0	$\Delta t_0 = \delta t$	B^0_0	every step
G_1	$\Delta t_1 = 2\delta t$	B^0_1	even steps
		B^1_1	odd steps
G_2	$\Delta t_2 = 4\delta t$	B^0_2	if n mod 4 = 0
		B^1_2	if n mod 4 = 1
		B^2_2	if n mod 4 = 2
		B^3_2	if n mod 4 = 3

Fig. 1. Example of when blocks are moved for 3 Δt groups.

allows a wide range of step sizes to be achieved using a small number of groups. Other schemes are likely to be possible, and perhaps will be preferable in some cases, but we have not explored them. It is probably unwise to alter a particle's step size by too large a factor, since the small truncation errors associated with a small step might easily be rendered meaningless by large errors from an adjoining step.

Particles are advanced only every 2^m $(m \geqslant 1)$ timesteps when they reside in a region of phase space where there are no large amplitude short wavelength fields ($kv\,\Delta t_m$ and $\omega_{\text{trap}}\,\Delta t_m$ are moderate); here k is a characteristic inverse scale length, v a characteristic particle velocity, and ω_{trap} the oscillation frequency $(ka)^{1/2}$, where a is the electric acceleration. In such a region we may also employ a coarse mesh. The large timestep used in such a region may help damp out the effects of "finite grid instability." This is a numerical instability which can arise as a result of the finite mesh spacing when the cell size is significantly larger than the Debye length [10].

The direct-implicit field equation is

$$\nabla \cdot (1 + \chi(x))\,\nabla\phi = \tilde{\rho}(x) \tag{2}$$

with the "effective susceptibility"

$$\chi(x) = \frac{1}{2} \sum_s \frac{q_s}{m_s}\, \tilde{\rho}_s(x)\, \Delta t^2 \tag{3}$$

and $\tilde{\rho}(x)$ the "free streaming" charge density obtained by moving particles to the advanced time level but omitting any effect of the electric field at that level. The sum is over all species. The field equation is solved over the entire domain every step. In this way, the deposition of charge in a direct-implicit code occurs implicitly, one step earlier than in an explicit code.

In the multiscale algorithm, we allow a block to deposit its information 2^m time levels ahead of the current step; this information is then interpolated backward in time to yield the data needed to produce the field a single time level ahead. "Future" $\tilde{\rho}$ data for a block (extrapolated 2^m levels ahead) are generated on steps during which the block is active; before this is done, the block's "old" (that is, existing) $\tilde{\rho}$ data (which is by now associated with the current time level) are first saved in a separate array. Thus, each block's $\tilde{\rho}$ is described by two mesh-sized arrays; these data are always available for interpolation, even on steps during which the block is inactive. This procedure makes it unnecessary to calculate $\tilde{\rho}(x)$ from the particles every step.

As particles are moved about in phase space, it is necessary to change their

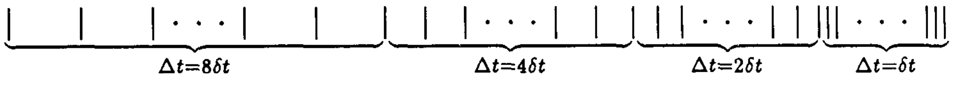

FIG. 2. Example of what a typical spatial grid might look like. Approximate spatial location of Δt groups are shown for the electrons.

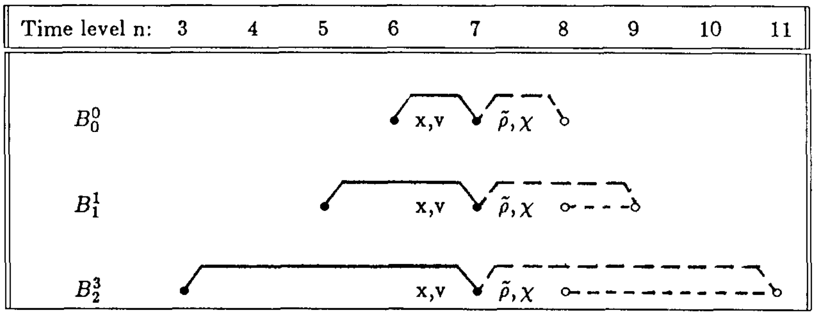

FIG. 3. Example of particle advance to $n = 7$ and interpolation of $\tilde{\rho}$ and χ at $n = 8$. Only blocks that are advanced at $n = 7$ are shown.

timesteps (move them from block to block). We allow changes in Δt by no more than a factor of 2, no matter where the particle is; we can change Δt again on the next step if necessary. The programming logic is simpler this way. In order to facilitate doubling or halving the step size "between steps," we employ a variant of the "d1" implicit particle advance [11] with all key quantities defined at integral (not staggered) time levels. This allows us to preserve second-order accuracy in time. A typical spatial grid might be that shown in Fig. 2.

Figure 3 shows the active blocks that are processed at time level $n = 7$. The dashed "arches" at the right represent the "pre-push" needed to obtain $\tilde{\rho}$ and χ, as described above. The lower dashed lines (for blocks B_1^1 and B_2^3) denote interpolation in time of $\tilde{\rho}$ and χ. The other blocks were advanced on earlier steps, and we need only interpolate their contributions to $\tilde{\rho}$ and χ back to $n = 8$ before the field solve, as shown in Fig. 4.

After the active particles have been pushed to $n = 7$ (and before their contribu-

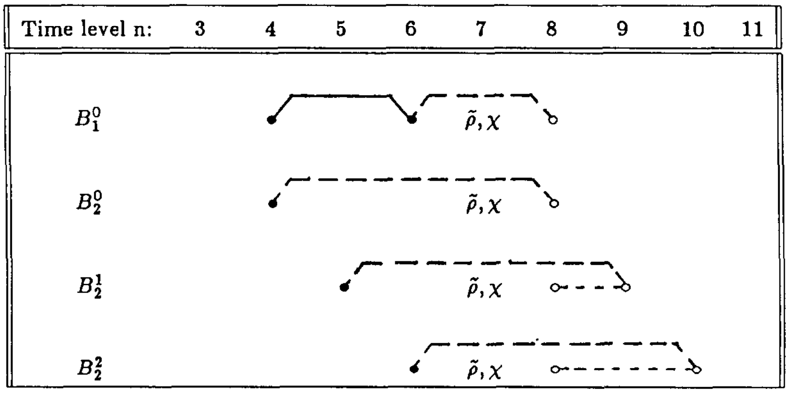

FIG. 4. Example of interpolation of $\tilde{\rho}$ and χ for blocks not moved at $n = 7$.

tions to $\tilde{\rho}$ have been accumulated), they are moved into new blocks if their (x, v) so dictate. The redistribution moves particles only into blocks that will be "pre-pushed" on the current step. Then the pre-push to $n=8$ (or beyond) is performed. Finally, $\tilde{\rho}$ and χ are interpolated to $n=8$ (for all blocks, both active and inactive, pre-pushed beyond 8), and the field equation is solved for ϕ at $n=8$.

It is unnecessary to save a χ for each block, because the total $\chi(x)$ is constructed from all of the (old and new) $\tilde{\rho}$'s at each step. A factor of Δt^2 enters into the calculation of the contribution to χ from a block B^l_m; assuming linear interpolation of $\tilde{\rho}$, it is correct to use the Δt_m associated with that block (roughly, it is also associated with a region of phase space or a grid spacing). Hence, the Δt^2's that go into the expression for χ are not the same for all blocks. This can be understood physically in the following way: the future field acts on infrequently-advanced particles over a longer time, and χ takes this into account, since it represents the action of that future field back upon itself via the particles.

The algorithm subsumes explicit electron subcycling schemes; ions would normally be processed only at the longest interval. However, initialization is simplified if all ions and electrons are placed in their B^0_0 blocks at the start, to avoid referencing a negative time-level. During the first few steps some delay in "promoting" particles into higher-number blocks must be imposed, so that the blocks within a group are uniformly populated. After a few steps, the small Δt_m ion groups will be empty.

2.1. *The Revised d1 Particle-Advance Scheme*

We seek a variant of the d1 scheme with x, v defined at the same (integer) time level, to facilitate changing the timestep size. We start with the scheme as it is usually written. The "final push" is:

$$\bar{a}_{n-1}=\frac{1}{2}\left\{\bar{a}_{n-2}+\frac{q}{m}E_n(\tilde{x}_n)\right\} \tag{4}$$

$$v_{n-1/2}=\tilde{v}_{n-1/2}+\frac{1}{2}\Delta t\frac{q}{m}E_n(\tilde{x}_n) \tag{5}$$

$$x_n=\tilde{x}_n+\frac{1}{2}\Delta t^2\frac{q}{m}E_n(\tilde{x}_n). \tag{6}$$

Then, the "pre-push" is:

$$\tilde{v}_{n+1/2}=v_{n-1/2}+\tfrac{1}{2}\Delta t\bar{a}_{n-1} \tag{7}$$

$$\tilde{x}_{n+1}=x_n+\Delta t\tilde{v}_{n+1/2}. \tag{8}$$

We move the computation of $\tilde{x}_{n+1}$ to the beginning of the "final push," where it becomes $\tilde{x}_n = x_{n-1} + \Delta t \tilde{v}_{n-1/2}$. We then relabel $\tilde{v}_{n+1/2}$, calling it v_n (it is formally centered at time level n, so this is a notational improvement). Then, we write

$$\begin{aligned} v_n &= v_{n-1/2} + \tfrac{1}{2}\Delta t \bar{a}_{n-1} \\ &= v_{n-1} + \tfrac{1}{2}\Delta t\{\bar{a}_{n-1} + E_n(\tilde{x}_n)\} \end{aligned} \tag{9}$$

to yield an advance from one integer level to the next.

2.2. *The Algorithm, in Detail*

We enter a timestep with the particle data x_{n-1}, v_{n-1}, and $\bar{a}_{n-2}$, and with E_n on the mesh. Strictly speaking, we should write a trivial generalization of the following, with incoming positions x defined at time level $n - 2^m$, etc., but we write the algorithm as if m were zero for clarity, so $\Delta t = \delta t$. In the following, time-subscripted quantities are stored in the particle arrays, while unsubscripted quantities are used only as scratch within the particle loops. The algorithm for one timestep is:

1. First we begin the "final push" loop over species, blocks and particles (Δt is really that of the current block; $\Delta t = \Delta t_m$):

   (a) $\tilde{x} = x_{n-1} + \Delta t v_{n-1}$

   (b) $\bar{a}_{\text{old}} = \bar{a}_{n-2}$

   (c) $a = (q/m)\, E_n(\tilde{x})$ (interpolation of field from mesh)

   (d) $\bar{a}_{n-1} = \frac{1}{2}\{a + \bar{a}_{n-2}\}$

   (e) $v_n = v_{n-1} + \frac{1}{2}\Delta t\{\bar{a}_{n-1} + a\}$

   (f) $x_n = \tilde{x} + \frac{1}{2}\Delta t^2 a$

   (g) Enforce particle boundary conditions; reflect or absorb, or shift by one period if using a periodic model.

   (h) If the particle has moved to a point in phase space where $\Delta t(x_n, v_n) \leqslant \Delta t/r$, set $\bar{a}_{n-1} = \frac{1}{2}\{a + \bar{a}_{n-1}\}$ and set the "new block" flag to "true." Here we have used either $r = \sqrt{2}$ or $r = 2$; the latter choice provides a useful "hysteresis" and prevents particles from changing step size back and forth unnecessarily in certain cases.

   (i) If the particle has moved to a point in phase space where $\Delta t(x_n, v_n) \geqslant r\, \Delta t$, set $\bar{a}_{n-1} = \bar{a}_{\text{old}}$ and set the "new block" flag to "true."

   (j) If the particle has been flagged to change blocks then, using $\tilde{x}$, subtract the particle's contribution from the $\tilde{\rho}$ array associated with the current block at time level n, and add the particle's contribution to $\tilde{\rho}$ of the new block.

2. At this point we exit the "final push" loop. For each active block, copy $\tilde{\rho}$ into $\tilde{\rho}_{\text{old}}$, then set $\tilde{\rho}$ to zero.

3. We now treat special cases.

   (a) Sort flagged particles into new blocks.

   (b) Inject any new particles by adding them to the appropriate blocks.

4. At this point we begin the "pre-push loop" over species, blocks and particles:

   (a) $\tilde{x} = x_n + \Delta t v_n$.

   (b) Using $\tilde{x}$, compute $\tilde{\rho}$ array associated with the current block at time level $n+1$.

5. At this point we exit the "pre-push" loop.

6. We then calculate the field quantities:

   (a) Interpolate $\tilde{\rho}$ and χ from all necessary blocks to time level $n+1$.

   (b) Perform the field-solve to obtain E_{n+1}.

Steps 1 through 6 are performed at each timestep. The particle advance scheme is formally second-order accurate and would be "time centered" (reversible) were it not for the damping built into the d1 mover. This is described in somewhat more detail in the context of a generalization [13] of the d1 advance; the reference discusses varying both the timestep size and a damping parameter. Our latest multi-scale work actually employs the generalized d1 scheme of the reference, but typically remains in the d1 limit of that algorithm. Centering of the electrostatic field is a result of the interpolations by which the source term is computed.

3. Test Results

Here we describe several simple tests using MIST. The first test (run #1) involves a translating slab of test particle electrons advanced every eighth step. There were 100 grid cells and 8192 infinitesimally charged electrons, all given initial velocity 0.005. Figure 5 shows the free streaming charge density of the slab at times $t = 32.0$ and $t = 33.0$. At the latter time, a "ledge" effect appears because this density is obtained by interpolation of quantities known at times 32.0 and 40.0, and the slab has moved a distance of 0.04 (four cells) during that longer interval; the charge density at $t = 32.0$ is 5×10^{-21} in the cell at $x = 42$. For this test particle case, the free streaming density is the same as the self consistent density, since there are no forces on the particles. Figure 6 shows the electrostatic potential in one selected cell, and the field energy, as functions of time. The potential varies linearly in time between those time levels where it is calculated using true (not interpolated) particle data. A characteristic "scalloping" of the field energy is evident, due to the smoothing of the charge density inherent in the interpolation.

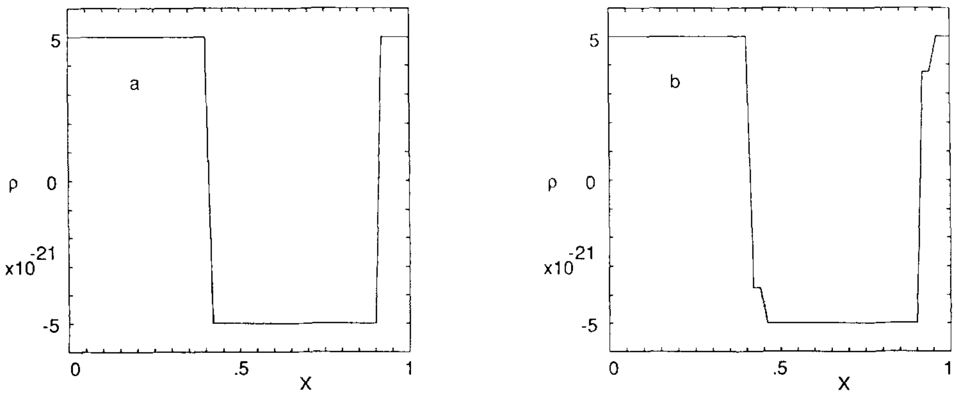

FIG. 5. Test particle electron charge density for translating slab run #1 at (a) time $t = 32.0$ and (b) $t = 33.0$.

In one class of tests, we considered the free expansion of a plasma into vacuum [2]. In run #2, we set Δt to 8.0, $t_{\max}$ to 4000, and loaded all particles into the base group so that they were advanced every step. We then made run #3 with a base Δt of 2.0, but with all particles in blocks B_2^0 so that they were advanced every fourth step; in this run, the interpolated values of $\tilde{\rho}$, χ, and E were not used to advance the particles at all. The object of this pair of tests was to verify that the two runs behaved identically. It was found that the answers agreed to all bits, as expected because the algebraic steps were exactly the same.

A similar but shorter run (#4) was done as a test of interpolation smoothness. There were 512 grid cells, and 8192 particles of each species. The electron plasma

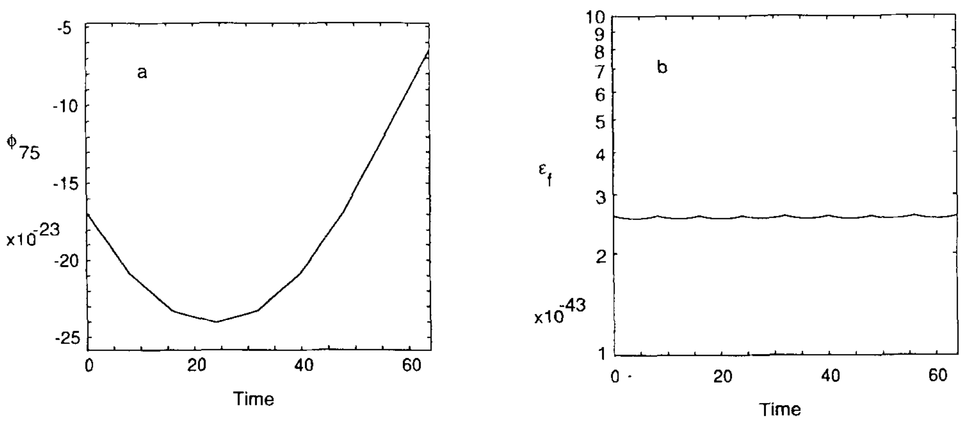

FIG. 6. Time histories of (a) the electrostatic potential in cell number 75 and (b) the field energy for translating slab test run #1.

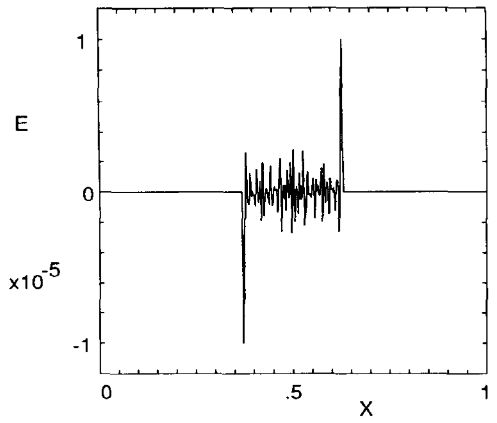

FIG. 7. A snapshot of the instantaneous electric field at the end of free expansion run #4.

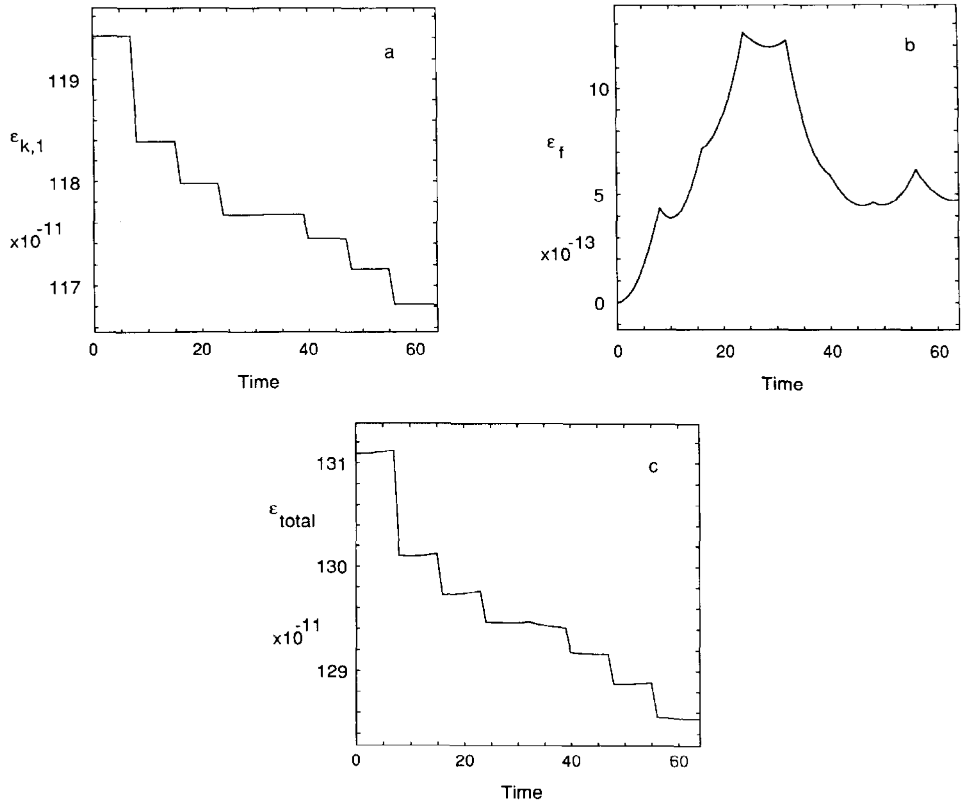

FIG. 8. Time histories of (a) electron kinetic, (b) field, and (c) total energies for free expansion run #4.

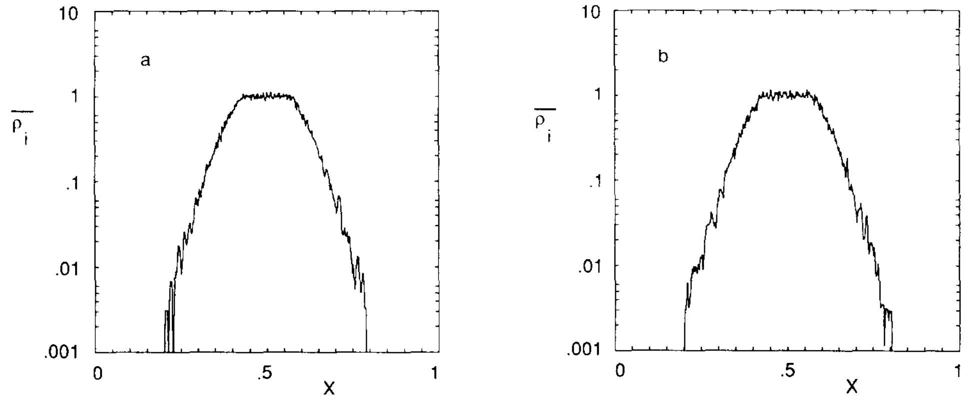

FIG. 9. Snapshots of the ion charge density at time $t = 15{,}000$ for (a) free expansion run #5, without subcycling, and (b) run #6, with ion advance every eighth step.

frequency was 1.0, and the mass ratio was 900:1.The system length was 1.0, the electron thermal velocity was 9.8×10^{-5}, and the ion thermal velocity was 1.03×10^{-6}. Again, the interpolated quantities are not used to move the particles, but they can be used as a diagnostic. We set $\Delta t = 1.0$, $t_{\max} = 64$, and advanced each particle on every eighth step (there were eight "particle steps"). The instantaneous electric field at the end of the run is shown in Fig. 7; the ambipolar effect is clearly evident at the slab edge. Figure 8 shows time histories of the electron kinetic, field, and total energies. The kinetic energy clearly changes only every eighth step, since particles are untouched during the interim. The field (and, hence, the total) energies do change on intermediate steps, because they involve interpolated quantities. The field energy exhibits scalloping.

As a test of electron subcycling, we made free expansion run #5 with $\Delta t = 8$, $t_{\max} = 20{,}000$, and compared it with run #6 where the ions were advanced every eighth step ($\Delta t_i = 64$). Figure 9 shows the ion charge density three quarters of the

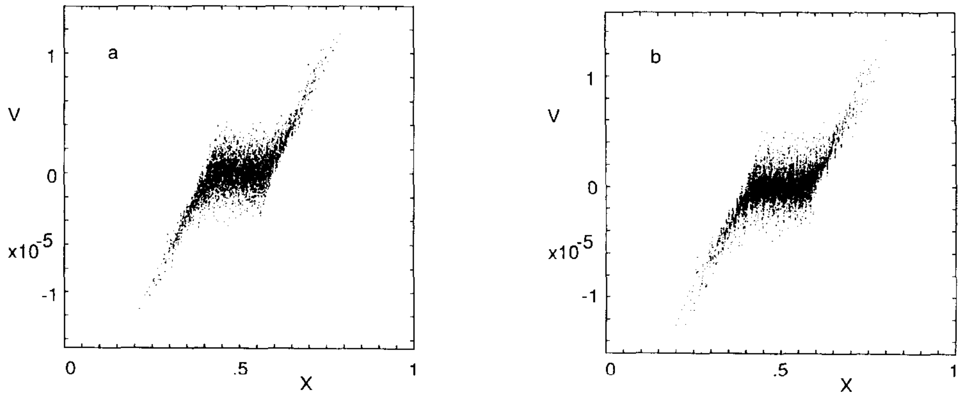

FIG. 10. Ion phase space at time $t = 15{,}000$ for (a) run #5 and (b) run #6.

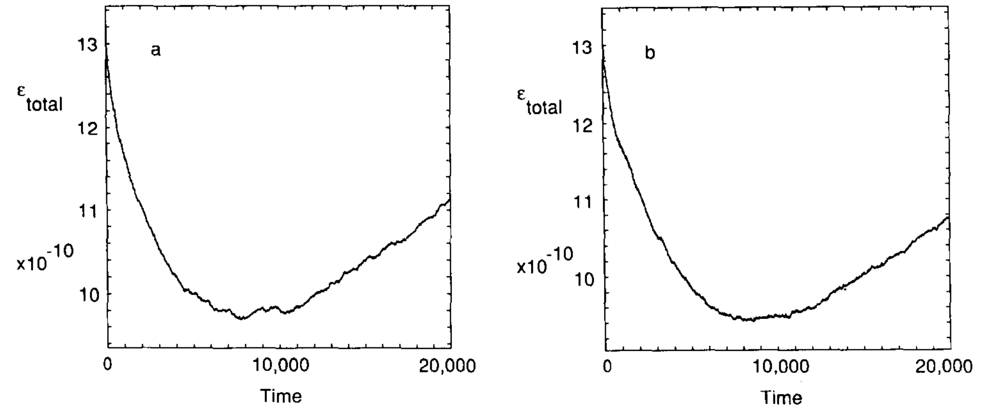

FIG. 11. Time histories of the total energy for (a) run #5 and (b) run #6.

way through these runs (it has been smoothed slightly by time averaging over 100 steps). Figure 10 shows snapshots of the ion phase space at this same time. Figure 11 shows time histories of the total energy. The good agreement between these two runs demonstrates that stable electron subcycling is possible. Note, however, the poor energy conservation for these parameters, which is most severe at the beginning of the run when the fields are large and the gradient lengths short. These implicit (not fully multi-scale) simulations discard the energy associated with motions which occur on short timescales. The free expansion problem may thus be a good test case for the full multi-scale algorithm (with particle redistribution).

4. Applicability

A problem of current interest is the simulation of an electrostatic sheath. It was found desirable to employ on the order of 500 particles per Debye length in order to keep the noise associated with the finite particle number to a reasonable level [12]. If one hopes to model a macroscopic plasma system (overall lengths centimeters to meters) which also includes a sheath region (scale lengths microns to millimeters), this requirement on N_p is clearly computationally prohibitive. However, we note two simplifying aspects of the problem, namely: (1) the very few electrons with speeds greater than about three times the thermal speed v_t charge the wall and are the ones which need be treated accurately; (2) far away (a few λ_D) from the wall, fine resolution in space and time may not be necessary, so we may use "smoother" physics in the central plasma region. We should note that conventional sheath simulations have typically treated the fast particles poorly, with $v_t \Delta t > \Delta x$.

A simple calculation shows the potential efficacy of our scheme. Assume the system is 10,000 Debye lengths in extent, roughly uniform in density, and contains a "sheath" region of 25 Debye lengths which is to be resolved in space and time. For simplicity let us base our particle step-size solely upon location and not at all upon velocity. If we require 500 particles of each species per Debye length, the total number of electrons is $N_e = 5{,}000{,}000$. Now assume that we want a small timestep in the sheath region, $\omega_{pe}\,\Delta t = 0.1$, but can tolerate $\omega_{pe}\,\Delta t \approx 100$ in the bulk plasma. Hence, we choose $m_{\max} = 10$, so that the ratio $\Delta t_{\max}/\Delta t_{\min} = 1024$. The block which must be pushed every step contains $25N_e/10{,}000 = N_e/400$ electrons. Assume that the transition timestep sizes are equally populated (to make a gradual transition region that will not spuriously reflect any plasma waves generated by the sheath); then $N_e/400$ particles are pushed every second step, $N_e/400$ every fourth, etc. The blocks which are pushed every 512 steps will contain about 24 particles each, enough for vectorization purposes. Thus, neglecting the particles which are pushed every 1024 steps, the number pushed on a step is

$$\left\{\tfrac{1}{2} + \tfrac{1}{4} + \cdots\right\} N_e/400 \approx N_e/200. \tag{10}$$

At the largest interval, we have $N_e - 10N_e/400 \approx N_e$ electrons being pushed, but only every 1024th step, so the number pushed on one step is $N_e/1024$. Thus, even with 10,000 Debye lengths, the region near the sheath dominates the cost. If we push ions only every 1024 steps, the net number of particles (electrons plus ions) pushed on a step is approximately $N_e/200 + 2N_e/1000 = 35{,}000$. Comparing this number with the total 10,000,000 particles in the system we obtain a *raw speedup* of a factor of 286. Finally, estimating that our computer can push 3×10^5 particles/s in 1d, the particle-pushing part of a step should take 0.11 s. Thus, neglecting field solving and particle sorting, in a one hour run we might take 33,000 steps so that $\omega_{pe} t_{\text{run}} = 3300$.

Of course, overhead associated with moving particles among blocks will reduce this speedup, as will the extra complexity of the implicit particle and field calculations. The realities of sorting and field interpolation, and the extra difficulty of optimizing a multi-scale code, will lead to smaller speed enhancements than that calculated above. Nonetheless, it is to be expected that significant real gains will be achieved on truly inhomogeneous problems.

There is considerable storage and computation overhead associated with the multiple source arrays, old and new, associated with the various blocks. Storage is less of a problem than it might seem, because the nonuniform meshes that are most appropriate for multiscale simulations contain far fewer grid points than corresponding uniform meshes capable of resolving small scale lengths. Further storage relief is possible because it is not really necessary to employ 2^m blocks of particles which are advanced every 2^m timesteps; one might use fewer, and simply delay "demotion" to larger timestep-size until the appropriate time were reached. In the extreme case of one block per value of m, this time level would be an integer multiple of 2^m; however, the presence of such extremely "special" time levels might lead to difficulties.

Research is needed into the computer science aspects of the technique. The re-distribution might be done by moving actual particle data, or by resetting pointers to that data. Dynamic allocation of storage might be desirable. Alternatively, one might not use blocks of particles at all, but merely associate a value of m, and an offset with each particle. Deposition would occur into an array indexed by both cell and this new information. A simple test at the top of the loop would inhibit processing of particles which were inactive on the current step. This approach would be simple to implement and would presere the identities of individual particles in a natural way for diagnostic purposes; it would probably not be efficient on a vector processor such as a CRAY computer, however, unless special care were taken.

5. Concluding Remarks

Particle-in-cell plasma simulation is very close to a first-principles description of a physical system. However, since it is necessary to employ far fewer particles than exist in a real plasma, the discrete-particle noise would be unacceptably large, were it not for the smoothing of short-range forces due to use of a field mesh, the smooth particle shape functions, and additional spatial filtering. Fortunately, in many real systems being modeled, long wavelength collective interactions generally dominate the physics; hence well-established techniques can be used with a high degree of confidence. Care is needed in choosing the simulation parameters.

However, many systems of interest, in particular, entire plasma systems with large space- and time-scales, cannot be treated by explicit methods because the costs would be prohibitive. Implicit methods have been developed which extend the domain of applicability of particle-in-cell simulations to such problems. Unfortunately, implicit simulations embody far less "fundamental" descriptions than do explicit ones; the damped equations of motion employed are qualitatively different from the reversible ones generally employed in explicit particle codes. Such damping is often necessary in order to discard under-resolved modes from the simulation, so that noise associated with them does not corrupt the low-frequency physics being studied. Even in the absence of large timesteps, the extra "quietness" of damped simulations can be a real advantage. It is possible to obtain this benefit in explicit simulations, without the additional computational costs of an implicit simulation, by using an explicit damped particle mover and/or EM field propagator [13]. Furthermore, the amount of damping can be adjusted to the needs of the problem at hand, in both implicit and explicit simulations.

As yet, implicit simulations are not so thoroughly understood as explicit ones. In particular, our current measures of goodness are incomplete. One criterion, energy conservation, provides a valuable clue to code behavior, but is not sufficient. Reasonable energy conservation is obtained for the expanding slab in the vicinity of a line in the 2D parameter space of cell size and timestep size [14]. However, the physics captured at different points along that line is very different. At small Δx and Δt, the behavior is similar to that of an explicit code; at the other extreme of large steps, almost all physics will be under-resolved. While it is possible to have overall energy conservation in that regime by carefully balancing Δx and Δt, one must be careful not to misinterpret this as proof that one is describing the physics of the problem, even at long wavelengths and low frequencies. For example, when the cell size is significantly larger than a Debye length, finite grid instability (a mesh aliasing effect) can set in. There is no reason to expect aliasing problems to disappear as the timestep size is increased, because they are a result of the spatial discretization. It could well be that one obtains good energy conservation at large cell size by balancing imposed damping and finite-grid instability. In many problems this balance may be acceptable; often implicit methods are used to avoid short timestep constraints imposed by a region one does not care about (e.g., a small high-density region). However, in other problems one will be damping away coherent flows in phase space and replacing them with (quasi-random) ones induced by the grid instability, and this may be unacceptable. Conventional implicit simulations have been, and will continue to be, very useful, but they are relatively harder to use and understand than explicit simulations.

The above considerations further motivate our efforts in developing multi-scale simulations. In addition to solving problems of direct physical interest, a multi-scale code can serve as a research tool. One can choose whether or not to resolve selected parts of the problem, to see whether or not such resolution is important to the phenomena in which one is interested. Of course, the dimensionality of the parameter space in a multi-scale simulation is quite large, and it is likely to be some time before our understanding is mature. The point here is that it is easy to do the experiments; the multi-scale method is in some ways less of an all-or-noting proposition than is a conventional implicit simulation.

We do not yet know if an irregular mesh (or spatially-varying spatial filtering) is in fact a necessity for efficient multi-scale simulations. There is an underlying assumption that the infrequently-pushed particles see a smoothly varying field; if this assumption were violated, then such orbits would be inaccurate. To some degree the smoothly varying field will happen naturally, since the effective susceptibility χ will in general be larger in regions where Δt is large, and, hence, the implicit field will be smoothed somewhat. However, it may be possible to enhance this smoothness by filtering in time rather than in space, or by using a spatially-dependent amount of damping in the equations of motion. The every-nth-step particles might be advanced using a field which has been smoothed (on the mesh) over the preceding n steps, using a lag-average or some other filter; the details of this are unclear. In any event, it may be desirable to use either a coarse mesh or filtering to make it easier to ensure that infrequently-pushed particles see only smooth fields.

Multi-scale modeling is quite open-ended. At its simplest the method can serve as a stable technique for electron sub-cycling, or as a technique for varying the timestep globally (with all particles using the same step size at any given step) as the configuration evolves. At the other extreme multi-scale algorithms may provide a natural way to advance particles on a nonuniform mesh, especially on a highly irregular mesh that is fitted to boundaries, is time-varying, etc.

An early explicit method which presages the present work has in fact employed a nonuniform mesh in 2D [15]. While unable to employ a very large timestep in the large-cell smooth-flow region, that author was apparently successful in using computer resources more efficiently than could have been done using a fixed-step-size code. Further discussion of spatially varying meshes can be found in [16].

It may be possible to employ a *locally implicit* description, wherein the field equation is solved implicitly only over each of a set of small overlapping subdomains. One does not need global implicitness, since it is not desired to use an infinite timestep size, and even in an electrostatic simulation "old" data at the boundary of a subdomain might well be good enough to serve as a boundary condition. This method has been successfully tested by the first author on the heat equation in 1D; it is indeed possible to exceed the explicit stability limiting timestep by a factor of several using such methods. Such methods may be necessary for implicit (or explicit electrostatic) simulations on loosely-coupled multiprocessing computers.

Other ideas (some of which have been employed in the past) aim at reducing the noise level and increasing the efficiency of particle simulations. One such approach involves the introduction of a fluid component along with the particles [17]; the fluid may create particles or absorb them depending upon how important kinetic effects are at each point in space. Another approach involves using particles of different "weight" (e.g., associates a greater weight to slow moving "bulk" particles but uses fewer of them). Here too, "coalescing" or "splitting" of particles is a possible element (though complicated by issues such as accounting for the "energy of constitution" of a particle that is to be split). These ideas are in no way in conflict with the multi-scale concept presented here.

The multi-scale method has recently been applied to a bounded plasma with encouraging results; these will be described in an upcoming paper [18].

Acknowledgments

We thank Bruce Cohen, Bruce Langdon, Rod Mason, and Richard Procassini for helpful discussions on multiple time scale and bounded plasma simulations. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory underr Contract W-7405-ENG-48, and by the University of California at Berkely under D.O.E. Contract DE-FG03-86ER53220 and the U.S. Office of Naval Research Contract N00014-85-K-0809.

References

1. R. J. Mason, *J. Comput. Phys.* **41**, 233 (1981).
2. J. Denavit, *J. Comput. Phys.* **42**, 337 (1981).
3. A. Friedman, A. B. Langdon, and B. I. Cohen, *Comments Plasma Phys. Controlled Fusion* **6**, 225 (1981).
4. D. C. Barnes, T. Kamimura, J.-N. LeBœuf, and T. Tajima, *J. Comput. Phys.* **52**, 480 (1983).
5. A. B. Langdon and D. C. Barnes, "Direct Implicit Plasma Simulation," in *Multiple Time Scales*, edited by J. U. Brackbill and B. I. Cohen (Academic Press, New York, 1985).

6. B. I. COHEN, M. E. STEWART, AND C. K. BIRDSALL, "Direct Implicit Particle Simulation of Tandem Mirrors," 11th International Conference on Numerical Simulation of Plasmas, Montreal, Canada, June 25–27, 1985 (unpublished).
7. D. W. HEWETT AND A. B. LANGDON, *J. Comput. Phys.* **72**, 121 (1987).
8. A. FRIEDMAN, C. K. BIRDSALL, S. E. PARKER, AND S. L. RAY, "Multi-Scale Particle Simulations," US-Japan Workshop on Plasma Modeling with MHD and Particle Simulation, Napa, CA, September 25–26, 1987 (unpublished).
9. A. Friedman, S. L. Ray, C. K. Birdsall, and S. E. Parker, "Particle-in-Cell Plasma Simulation with a Wide Range of Space and Time Scales," 12th Conference on Numerical Simulation of Plasmas, San Fransisco, CA, September 20–23, 1987 (unpublished).
10. C. K. BIRDSALL AND A. B. LANGDON, *Plasma Physics Via Computer Simulation* (McGraw–Hill, New York, 1985).
11. B. I. COHEN, A. B. LANGDON, AND A. FRIEDMAN, *J. Comput. Phys.* **46**, 15 (1982).
12. L. A. SCHWAGER, Ph. D. thesis, University of California, Berkeley, 1987 (unpublished).
13. A. FRIEDMAN, "A Second Order Implicit Particle Mover with Adjustable Damping," *J. Comput. Phys.* **90**, 292 (1990).
14. B. I. COHEN, A. B. LANGDON, D. W. HEWETT, AND R. J. PROCASSINI, *J. Comput. Phys.* **81**, 151 (1989).
15. R. C. CHAKY, Ph. D. thesis, University of Kansas, 1981 (unpublished).
16. T. TAJIMA, *Computational Plasma Physics* (Addison–Wesley, New York, 1989), Chap. 10; M. LEBRUN AND T. TAJIMA, *J. Comput. Phys.*, submitted.
17. J. DENAVIT, *Space Sci. Rev.* **42**, 85 (1985); reprinted in *Space Plasma Simulations*, edited by M. Ashour-Abdalla and D. A. Dutton (Reidel, Boston, 1985).
18. S. E. PARKER, A. FRIEDMAN, S. L. RAY, AND C. K. BIRDSALL, Bounded multi-scale plasma simulation: Application to sheath problems, *J. Comput. Phys.*, submitted.