Available online at www.sciencedirect.com

Computer Physics Communications 164 (2004) 209–213

Computer Physics Communications

www.elsevier.com/locate/cpc

The particle-continuum method: an algorithmic unification of particle-in-cell and continuum methods [☆]

Srinath Vadlamani [*], Scott E. Parker, Yang Chen, Charlson Kim [1]

Center for Integrated Plasma Studies, University of Colorado, Boulder, CO 80309, USA

Available online 24 July 2004

Abstract

A new numerical algorithm that encompasses both the δf particle-in-cell (PIC) method and a continuum method has been developed, which is an extension to Denavit's [J. Comput. Phys. 9 (1972) 75] original "hybrid" method. In this article we describe this new Particle-Continuum algorithm in general, and we note our methods of interpolation. The issue of phase space convergence of this algorithm is discussed. We analyze the induced numerical diffusion of such an algorithm and compare theory with results. We also created a simple problem that demonstrates this algorithm solves the "growing weight" problem.
© 2004 Elsevier B.V. All rights reserved.

PACS: 07.05.T; 52.65; 52.35; 52.55

Keywords: Plasma; Kinetic; Vlasov; Particle; Simulation

1. Introduction

Expanding upon Denavit's hybrid method [1] and Batischev's PIC–Vlasov method [2–4] we have created a new algorithm that can vary between a continuum and δf PIC method. A primary distinction of the method presented here is the use of the $\delta f = f - f_0$ to describe the distribution function [11]. This study is of value for two reasons: (1) It makes a connection between the particle-based δf and continuum methods for solving kinetic equations thereby allowing detailed comparisons of the two methods. (2) The method solves the "growing weight problem" in δf particle simulations of plasma turbulence as detailed by Krommes and Hu [5]. The basic algorithm is essentially a variant of the δf method. A simple description of the algorithm is as follows: (1) load particles (or characteristics) on a uniform lattice in phase space. The loading need not be uniform, but it greatly simplifies the algorithm, (2) advance the characteristics M time steps, using the usual δf PIC algorithm which involves a grid interpolation, deposition, then field solve all on a spatial grid, (3) every M time steps, deposit δf on the phase space grid, then reset the particle phase space coordinates back to their initial value on the phase space lattice. Also, reset the particle value of δf to the phase space grid value.

☆ This research is supported by the VIGRE Grant #DMS-9810751.

* Corresponding author.
E-mail address: srinath.vadlamani@colorado.edu (S. Vadlamani).
URL: http://srinath.colorado.edu (S. Vadlamani).

1 Currently at UW Madison.

0010-4655/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2004.06.031

In this article we will discuss issues of algorithm convergence, induced diffusion and the "growing weight problem" using a simple nonlinear two-dimensional slab model, bounded in x and periodic in y where the phase space is $(x, y, v_\parallel)$. $v_\parallel$ is also bounded, the ions are treated as gyrokinetic and the electrons are assumed to be adiabatic. This is a standard ion-temperature-gradient (ITG) instability model, which has been previously described in Refs. [10,12,13]. In our study, we implement a quadratic weighting interpolation scheme [7], for small $(k_\perp \rho_i)^2$, where $k_\perp$ is the wave number of the waves perpendicular to the magnetic field lines, and ρ_i is the ion gyroradius.

2. The particle-continuum method

We begin with the perturbed Vlasov (conventional or gyrokinetic) equation

$$\frac{\partial \delta f}{\partial t} + \dot{\mathbf{z}} \cdot \frac{\partial \delta f}{\partial \mathbf{z}} = -\dot{\mathbf{z}}^1 \cdot \frac{\partial f_0}{\partial \mathbf{z}}, \tag{1}$$

where $\mathbf{z} = (\mathbf{R}, v_\parallel, \mu)$ and $\dot{\mathbf{z}} = \dot{\mathbf{z}}^0 + \dot{\mathbf{z}}^1$. $\mathbf{R}$ denotes real space coordinates, $v_\parallel$ is the velocity parallel to magnetic field lines, and μ is the magnetic moment, which is conserved along particle orbits. The "0" superscript is for equilibrium characteristic and the "1" superscript is for the perturbed part of the characteristic (characteristic $\leftrightarrow$ orbit).

We solve this equation in characteristic form

$$\dot{\delta f} = -\dot{\mathbf{z}}^1 \cdot \frac{\partial f_0}{\partial \mathbf{z}}. \tag{2}$$

In a fashion similar to conventional δf PIC methods we approximate δf as

$$\delta f(\mathbf{z}, t) = \sum_i \delta f_i(t) \Delta V_i S^N\big[\mathbf{z} - \mathbf{z}_i(t)\big], \tag{3}$$

where i is the particle index, ΔV_i is the specified amount of phase space volume associated with the ith marker particle [4], $S^N(\mathbf{u})$ is an N-dimensional interpolation function and N is the dimension of the phase space ($N = 2$–6).

We now choose a N-dimensional lattice, with lattice points $\mathbf{z}_\mathbf{l}$, where $\mathbf{l}$ is a vector of length N used to identify each lattice point. The simplest thing to do is make the lattice uniform, however, a more general lattice, even an unstructured lattice, can be used in principle. Initially we load marker particles at the lattice points. δf on the lattice is given by

$$\delta f(\mathbf{z}_\mathbf{l}, t) = \sum_i \delta f_i(t) \Delta V_i S^N\big[\mathbf{z}_\mathbf{l} - \mathbf{z}_i(t)\big]. \tag{4}$$

We can now describe the algorithm in its most general form. Geometry, specific field equations, and dimensionality have little impact on the basic algorithm. The algorithm is the following:

(1) Initially load the marker particles (or characteristics) on the lattice: $\mathbf{z}_i = \mathbf{z}_\mathbf{l}$. Load the same number of particles as lattice points.
(2) Advance the marker particles M time steps.
   (a) Integrate the equation of motion $\dot{\mathbf{z}}$. Field quantities are obtained by spatial interpolation to particle locations.
   (b) Integrate Eq. (2).
   (c) Deposit all particles quantities on the spatial grid. For example, the density is

$$\delta n(\mathbf{x}_{j,k,l}, t) = \sum_i \delta f_i(t) \Delta V_i S^3\big[\mathbf{x}_{j,k,l} - \mathbf{x}_i(t)\big] \tag{5}$$

   in three dimensions where the j, k, l correspond to the 3 grid indexes in x, y, z. Higher order moments (e.g., current or pressure) could be deposited as well. The details of the geometry hide in the interpolation function S^3.
   (d) Solve for field quantities.
   (e) Repeat (a)–(d) M time steps.
(3) Every Mth time step
   (a) Deposit δf on the phase space grid[2]

$$\delta f(\mathbf{z}_\mathbf{l}, t) = \sum_i \delta f_i(t) \Delta V_i S^N\big[\mathbf{z}_\mathbf{l} - \mathbf{z}(t)_i\big]. \tag{6}$$

   (b) Reset the marker particle phase space coordinates back to their initial value on the phase space lattice $\mathbf{z}_i = \mathbf{z}_\mathbf{l}$.

[2] We will distinguish between lattice and grid the following way. When we refer to the marker particle initial (or reset) phase space positions we call this the lattice. When we refer to the locations of where we have δf on a grid, we call this the grid. However, in our implementation, the grid points and lattice points are the same locations.

(c) Reset the marker particle value of δf to the grid value $\delta f_i = \delta f(\mathbf{z}_i = \mathbf{z}_\mathbf{l})$.

(4) Repeat 2–3.

For $M \to \infty$ one recovers the usual δf PIC algorithm with, in our study, a uniform loading of particles. For $M = 1$ the algorithm is similar to the continuum method of Cheng and Knorr [8]. One subtle difference between typical continuum methods such as Cheng and Knorr, and the method presented here for $M = 1$, is that conventional continuum methods involve a classic spline fit problem, whereas our algorithm requires a weighting scheme or the inverse of a classic spline fit. Any value of $M = 1, 2, 3, \ldots$ is permissible. Any loading (e.g., random or bit-reversed) is possible, but then requires a more sophisticated interpolation scheme for a nonuniform and unstructured phase space grid (possibly using a finite element method).

3. Simulation results

3.1. Algorithm convergence

In considering convergence of such an algorithm, we compared the value of the potential at the time of nonlinear saturation and linear growth rate associated with the ion temperature gradient instability [10,12,13], between many simulations with varying values of small, finite M, to the usual δf PIC algorithm, with $M \to \infty$. For the $M \to \infty$ case, and for a fixed phase–space grid, the above mentioned values agreed with establish δf PIC methods [12]. The issue of conservation of particle number, energy, and momentum has been addressed and confirmed in Ref. [12]. The simulations presented here do not conserve energy because we did not retain the $E_\parallel$ nonlinearity for these ITG simulations. Fig. 1 shows the nonlinear saturation levels of the (1, 1) mode of the electrostatic potential. The $v_\parallel$ grid size was fixed at 64 grid cells between $(-4v_t, 4v_t)$, where v_t is the ion thermal velocity, and we varied the spatial grid size. This result shows the necessity of larger spatial grid number for convergence of the smaller M limit. In Fig. 1, we see that for the case $M = 1$, there is no data for spatial grid number equal to 16. The simulation at this resolution and M setting, never achieved a nonlinear saturation. The coarse spatial grid

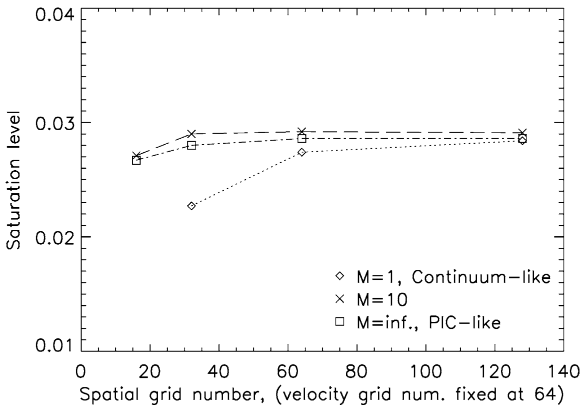

Fig. 1. The small M continuum limit requires greater spatial resolution.

may result in destroying the $\mathbf{E} \times \mathbf{B}$ trapping saturation mechanism. For all cases of M, we do see convergence of the algorithm with finer phase space resolution.

3.2. Induced numerical diffusion

The resetting of marker particles can be shown to be a numerical diffusion process, as was first studied by Denavit [1]. We will refer to this numerical diffusion as *induced* diffusion since it is due to the primary aspect of the algorithm, i.e. the resetting of the marker particles.

In order to obtain a better understanding of the induced diffusion in our algorithm, a simple model was created with only motion in $\hat{\mathbf{y}}$ direction, with all particles having the same velocity, and no external forces. Hence, only interpolation in the $\hat{\mathbf{y}}$ direction will result in induced diffusion.

The derivation by Denavit shows that the Fourier transform of an interpolation of the distribution function $\delta f(y)$ can be written as

$$\widetilde{H}(k_y) = W(k_y)H(k_y), \tag{7}$$

where $H(k_y)$ is the Fourier transform of $\delta f(y)$, $W(k_y)$ is the Fourier transform of the interpolation function (or particle shape), and $\widetilde{H}(k_y)$ is the resulting interpolation. We define $D(k_y) = 1 - W(k_y)$, and for $\frac{k_y \cdot \Delta y}{\pi} \ll 1$, we interpolate m-times. Then *by induction*,

$$\widetilde{H}(k_y; m) = H(k_y; m = 0) \cdot \bigl(1 - D(k_y)\bigr)^m. \tag{8}$$

Table 1
D is the diffusion coefficient due to interpolation. Wave number $k_y \cdot \Delta y = \frac{2\pi}{l_y} \cdot \Delta y = \frac{\pi}{16}$, at a fixed velocity $v_0 = 2$. M is the number of time steps before interpolation

	Linear $D_1\{\times 10^{-3}\}$	Quadratic $D_2\{\times 10^{-3}\}$
Theory	3.208	4.809
$M = 2$	3.079	4.807
$M = 3$	3.079	4.807
$M = 12$	3.079	4.807

For $\frac{k_y \cdot \Delta y}{\pi} \ll 1$, we obtain

$$\widetilde{H}(k_y; m) \approxeq H(k_y; m = 0) \cdot \mathrm{e}^{-m \cdot D(k_y)}. \tag{9}$$

In Fourier space, the repeated interpolation of a discretized particle distribution appears as a diffusion of the initial distribution. Thus, given k_y, $H(k_y; m = 0)$ and after m interpolations one can measure $\widetilde{H}(k_y; m)$, then it follows that one can calculate $D(k_y)$, the induced diffusion coefficient.

In Table 1, we have calculated diffusion coefficients for the 1-d case for the linear and quadratic interpolation functions as implemented in our algorithm for the deposition of δf on the phase space [7]. The theoretical value of $D(k_y)$, calculated via evaluating the Fourier transform of these interpolating functions at the known k_y, corresponds with the measured value of $D(k_y)$, for various M, the number of time steps before interpolation. Each measurement of $D(k_y)$, is obtained after one deposition, for the same initial distribution of particles loaded uniformly in the (x, y) plane, and with a fixed parallel velocity of $v_\parallel = 2$. The single deposition measurement alleviates the propagation of round-off error when comparing with theoretical values. Also, these measurements were conducted with out the influence of forces. Quadratic relative to linear interpolation distributes particle weights over a larger number of grid cells (4 versus 2 in one dimension), resulting in larger diffusion. Table 1 also shows no M dependence, where M is the number of time steps before resetting. This suggests that the effect of diffusion is a result of the phase space interpolation or resetting and not the time integration along trajectories.

3.3. Solving the growing weights problem

The "growing weight problem" in δf particle simulations of plasma turbulence [5] is one limiting feature of the δf algorithm for long-time simulations. Beginning with the gyrokinetic Vlasov equation for a shear free slab, multiplying by $\delta f/f_M$, and using $\nabla \cdot \mathbf{V_E} = 0$, one obtains [5],

$$\frac{1}{2}\frac{\partial}{\partial t}\left(\frac{\delta f^2}{f_M}\right) + \frac{1}{2}\frac{\partial}{\partial z}\left[v_\parallel\left(\frac{\delta f^2}{f_M}\right)\right] + \nabla \cdot \left[\mathbf{V_E}\left(\frac{\delta f^2}{f_M}\right)\right] = \delta f\left[V_{E_x}\kappa + \left(\frac{q}{T}\right)v_\parallel E_\parallel\right] - \delta f \widehat{C}[\delta f], \tag{10}$$

where the distribution function $f(z, v_\parallel, t) = f_M(v_\parallel) + \delta f(z, v_\parallel, t)$, f_M denotes a Maxwellian equilibrium distribution, and δf is the perturbation. $\mathbf{V_E}$ is the velocity due to the $\mathbf{E} \times \mathbf{B}$ drift, q is ion charge, $T = T(z)$ is temperature, $\kappa = \kappa_n + \frac{1}{2}(v_\parallel^2/v_t^2 - 1)\kappa_T$, and $\widehat{C}$ is a linear collisional operator, where v_t.

From this, we can average Eq. (10) over all phase space, use periodic boundary conditions, and neglect finite gyro-radius effects, we obtain

$$\partial_t \overline{\mathcal{F}} = \kappa_n \overline{\Gamma} + \frac{1}{2}\kappa_T \overline{Q} - \overline{\mathcal{D}}, \tag{11}$$

where $\kappa_n = |\nabla n/n|$, $\kappa_T = |\nabla T/T|$, $\overline{(\cdots)}$ is an averaging over all phase space, $\mathcal{F} = \delta f^2/f_M$, Γ and Q are ion particle and ion heat flux, respectively. $\mathcal{D} = \delta f \widehat{C}[\delta f]$, which is the dissipation due to collisions. Arguing the effects of the last two terms are small [9, 10], we obtain

$$\overline{\mathcal{F}}(t) \propto \kappa^2 D t, \tag{12}$$

where D is the turbulent diffusion coefficient of a test particle. It can be shown [6] that $\overline{\mathcal{F}}(t) \propto \frac{1}{2}\sum_i w_i^2(t)$, where the sum is over all marker particles w_i, which are the weights in δf particle simulations used to sample phase space [11,12]. Eq. (12) establishes an explanation of the "growing weight" problem [5], the sum of the weights squared grow linearly in time due to turbulent diffusion.

The Particle-Continuum algorithm resets the locations of the weight after M time steps, via interpolation of δf on the phase space grid. Thereby, course-graining δf and limiting the growth in $(\delta f)^2$. In order to mock-up turbulent diffusion and the associated growing weights, we tested a simplified problem in which we randomly adjust the weights' values by a known small amount. Thus, for the usual δf PIC algorithm, with $M \to \infty$, we see the growing weight phenomena, exhibited by the linear growth of $\sum_i w_i^2$ (which is analogous to $\sum_i (\delta f)^2/f_M$), as

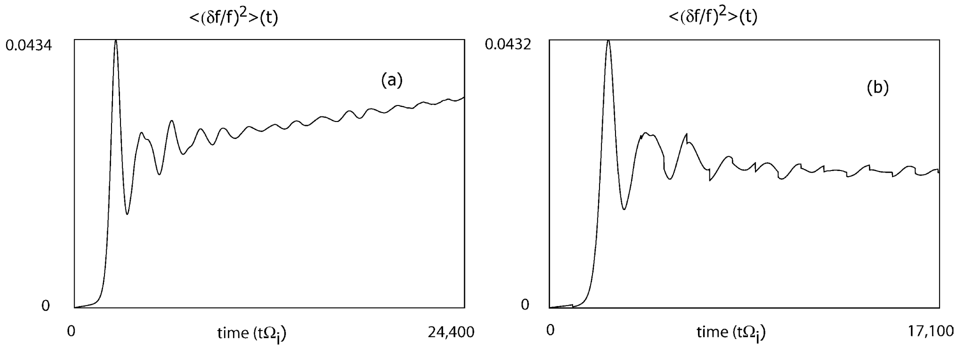

Fig. 2. (a) (δf-PIC) case shows growing $\sum(\text{weights})^2$, (b) $M = 100$, the $\sum(\text{weights})^2$ growth is suppressed.

shown in Fig. 2(a). Setting M to a finite value, we were able to suppress the growth of the $\sum_i w_i^2$, as shown in Fig. 2(b). Also, evident in the $\sum_i w_i^2$ diagnostic shown in Fig. 2(b) is a sawtooth feature due to the re-distribution of the weights every M time steps.

4. Conclusions

We have created an algorithm that has the ability to behave both like a δf PIC method and a variant of a continuum method. This is accomplished by performing a typical δf PIC algorithm up until M time steps with standard field solves, after which δf is interpolated to the grid and the particles are reset to their original phase space coordinates with the new δf values. It is demonstrated that for $M \to \infty$, we recover the standard δf PIC algorithm. We noticed a convergence for lower M values as the velocity resolution is increased. Also, we observed the necessity of spatial refinement for the small M limit case to achieve convergence.

We also studied the induced numerical diffusion of this scheme. Numerical and theoretical values were compared, giving validity of such a concept as *resetting* being analogous to diffusion. The method was created with the intention of using $M > 1$, such that resetting of the weights of the δf PIC algorithm has helped solve the growing weights problem.

References

[1] J. Denavit, J. Comput. Phys. 9 (1972) 75.
[2] O. Batishchev, in: Proc. 17th. Conf. on Num. Sim. Plasmas, Banff, Canada, May 2000, p. 24;
A. Batishcheva, O. Batishchev, in: Proc. 17th. Conf. on Num. Sim. Plasmas, Banff, Canada, May 2000, p. 29;
O. Batishchev, et al., Bull. APS 45 (7) (October 2000) 72, APPS DPP meeting, Quebec City, Canada;
O. Batishchev, APS DcomP-2001 meeting, R1.025, Cambridge, June 2001.
[3] O. Batishchev, A.A. Batishcheva, J. Zhang, in: Proc. 18th. Conf. on Num. Sim. Plasmas, Cape Cod, USA, Sept. 2003, p. 375.
[4] O. Batishchev, M. Martinez-Sanchez, Charged Particle Transport in the Hall Effect Thruster, paper IEPC03-188, -10p, 2003.
[5] J. Krommes, G. Hu, Phys. Plasmas 1 (1994) 3211.
[6] G. Hu, J. Krommes, Phys. Plasmas 1 (1994) 863.
[7] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, A. Hilger, Philadelphia, 1988.
[8] Cheng, Knorr, J. Comput. Phys. 22 (1976) 330.
[9] W.W. Lee, Phys. Fluids 26 (1983) 556.
[10] W.W. Lee, W.M. Tang, Phys. Fluids 31 (1988) 612.
[11] S.E. Parker, W.W. Lee, Phys. Fluids B 5 (1993) 77.
[12] S.E. Parker, W. Dorland, R.A. Santoro, M.A. Beer, Q.P. Liu, W.W. Lee, G.W. Hammet, Phys. Plasmas 1 (1994) 1461.
[13] S.E. Parker, JCP 178 (2002) 2.